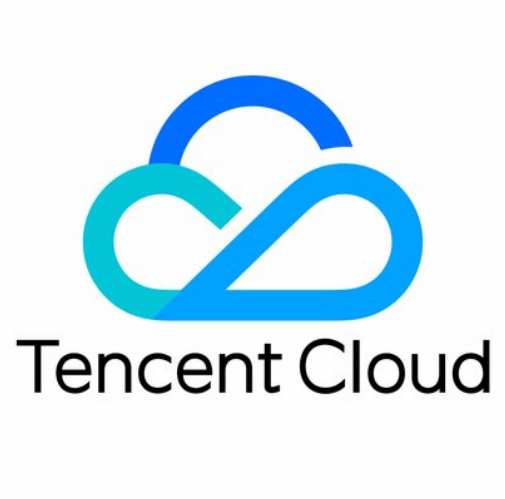
Serverless Framework

Framework Support

Product Documentation



Contents

Framework Support
- List of Supported Frameworks
- Deploying Python Django
- Deploying PHP Laravel
- Deploying ThinkPHP Framework
- Fast Construction of RESTful API
- Deploying Python Flask
- Deploying Koa Framework
- Deploying Egg.js Framework
- Deploying Next.js Application
- Deploying Express.js Application
- Deploying Nuxt.js Application

Framework Support
List of Supported Frameworks

Last updated : 2021-01-15 15:30:53

In addition to basic SCF applications, Serverless Framework also provides solutions to migrate applications in traditional frameworks to the serverless platform based on serverless components. Currently, Serverless Framework supports the following components:

Component	Use Instructions	Full Configuration
Express component	Use Instructions	serverless.yml Configuration
Koa component	Use Instructions	serverless.yml Configuration
Egg component	Use Instructions	serverless.yml Configuration
Next.js component	Use Instructions	serverless.yml Configuration
Nuxt.js component	Use Instructions	serverless.yml Configuration
Flask component	Use Instructions	serverless.yml Configuration
Django component	Use Instructions	serverless.yml Configuration
Laravel component	Use Instructions	serverless.yml Configuration
ThinkPHP component	Use Instructions	serverless.yml Configuration

Deploying Python Django

Last updated : 2021-01-08 16:13:04

Overview

Tencent Cloud Django component uses Tencent Serverless Framework. Based on serverless services (such as COS) in the cloud, it can implement "zero" configuration, convenient development, and rapid deployment of your Django webpage applications.

Django features:

- **"Zero" configuration**: you only need to write project code and then deploy it, and the Serverless Framework will take care of all the configuration work.
- **Pay-as-you-go billing**: fees are charged based on the request usage, and you don't need to pay anything if there is no request.
- **Fast deployment**: you can deploy your entire webpage application in just a few seconds.
- **Convenient collaboration**: the development mode and in-cloud debugging are supported to facilitate team collaboration.
- **Rich extensions**: RESTful API services can be deployed.

Directions

1. Install Serverless CLI

Install the latest version of Serverless CLI through npm:

```
$ npm install -g serverless
```

2. Initialize the Django template project (optional)

If you don't have a local Django project, you can initialize a Django project with the following commands (if you already have one, you can ignore this step):

```
serverless init django-starter --name example
cd example
```

3. Install project dependencies

If you want to create a project by yourself, please install the dependencies required by Python in the project directory. For example, Django is needed in this example, and you can run `pip` to install it:

```
pip install Django -t ./
```

4. Configure the .yml file

Create a `serverless.yml` file in the project root directory:

```
touch serverless.yml
```

Paste the following configuration template into the file to implement basic project configuration.

> **Note :**
>
> You can add more configuration items in `serverless.yml` based on your actual deployment needs. For more information on the configuration of the .yml file, please see Django Component Configuration.

```
#serverless.yml
component: django
name: djangoDemo
org: orgDemo
app: appDemo
stage: dev

inputs:
region: ap-guangzhou
djangoProjectName: mydjangocomponent
src: ./src
functionConf:
timeout: 10
memorySize: 256
apigatewayConf:
protocols:
- https
environment: release
```

5. Deploy the application

Deploy by running the `sls deploy` command, and you can add the `--debug` parameter to view the information during the deployment process:

```
sls deploy --debug
```

After the deployment is completed, access the application by accessing the output API Gateway link.

Account configuration (optional)

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
$ touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

Note :

- If you don't have a Tencent Cloud account yet, please sign up first.
- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

Deploying PHP Laravel

Last updated : 2021-01-15 15:47:54

Overview

Tencent Cloud Laravel Serverless Component supports Laravel v6.0 and above.

Directions

1. Install Serverless CLI

Use npm to install Serverless CLI globally:

```
npm install -g serverless
```

Note :

The following steps are mainly for deployment on the command line. For deployment in the console, please see Console Deployment Guide.

2. Initialize the Laravel template project (optional)

If you don't have a local Laravel project, you can initialize a Laravel project with the following commands (if you already have one, you can ignore this step):

```
serverless init laravel-starter --name example
cd example
```

3. Configure the .yml file

Create a `serverless.yml` file in the project root directory:

```
touch serverless.yml
```

Paste the following configuration template into the file to implement basic project configuration.

Note :

> You can add more configuration items in `serverless.yml` based on your actual deployment needs. For more information on the configuration of the .yml file, please see Laravel Component Configuration.

```
# serverless.yml

component: laravel
name: laravelDemo
org: orgDemo
app: appDemo
stage: dev

inputs:
src: ./
region: ap-guangzhou
runtime: Php7
apigatewayConf:
protocols:
- http
- https
environment: release
```

4. Deploy the application

Deploy by running the `sls deploy` command, and you can add the `--debug` parameter to view the information during the deployment process:

```
sls deploy --debug
```

After the deployment is completed, access the application by accessing the output API Gateway link.

5. Monitor the OPS

After the deployment is completed, you can log in to the Serverless Framework console to view the basic information of the application and monitor logs.

Account configuration (optional)

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

Note :

- If you don't have a Tencent Cloud account yet, please sign up first.
- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

Deploying ThinkPHP Framework

Last updated : 2020-09-04 15:55:38

Overview

Tencent Cloud ThinkPHP Serverless Component supports deploying RESTful API services based on ThinkPHP 6.x or above.

Prerequisites

Initializing ThinkPHP project

1. The PHP environment has been set up locally. The recommended version is PHP 7.1 or above (ThinkPHP 6 is recommended).
2. Install Composer: ThinkPHP uses Composer to manage dependencies. For the installation method, please see here.

After the above two steps are completed, use Composer to initialize a `ThinkPHP` project:

```
composer create-project topthink/think serverless-thinkphp
```

Operation Guide

1. Install

Use npm to install Serverless CLI globally:

```
npm install -g serverless
```

2. Configure

Create a `serverless.yml` file in the project root directory:

```
$ touch serverless.yml
```

Configure `serverless.yml` as follows:

```
# serverless.yml
org: orgDemo
app: appDemo
stage: dev
component: thinkphp
name: thinkphpDemo

inputs:
src:
src: ./
exclude:
- .env
region: ap-guangzhou
runtime: Php7
apigatewayConf:
protocols:
- http
- https
environment: release
```

More configuration items >>

3. Deploy

> ⚠ **Note :**
>
> Before deployment, you need to run `php think clear` to clear the locally running configuration cache.

Deploy by running the `sls` command, and you can add the `--debug` parameter to view the information during the deployment process:

```
$ sls deploy --debug
```

4. Remove

You can run the following command to remove the deployed ThinkPHP project:

```
$ sls remove --debug
```

Account configuration (optional)

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

Note :

- If you don't have a Tencent Cloud account yet, please sign up first.
- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

More components

You can view more component information in the repository of Serverless Components.

Fast Construction of RESTful API

Last updated : 2020-08-28 09:41:29

Operation Scenarios

The RESTful API template uses the Tencent SCF component and its trigger capabilities for easy creation, configuration, and management of a RESTful API application in Tencent Cloud. You can use the Serverless SCF component to quickly construct a RESTful API application and perform `GET` and `PUT` operations.

Directions

1. Install

Install Serverless Framework:

```
$ npm install -g serverless
```

2. Configure

Directly download the sample by running the following command:

```
serverless create --template-url https://github.com/serverless/components/tree/v1/templates/tencent-python-rest-api
```

The directory structure is as follows:

```
.
├─── code
|  └─── index.py
└─── serverless.yml
```

Currently, the SCF component supports v2; therefore, the `serverless.yml` file has been modified to the following:

```
# serverless.yml

component: scf
name: apidemo
org: test
```

```
app: scfApp
stage: dev

inputs:
name: myRestAPI
enableRoleAuth: true
src: ./code
handler: index.main_handler
runtime: Python3.6
region: ap-guangzhou
description: My Serverless Function
memorySize: 128
timeout: 20
events:
- apigw:
name: serverless
parameters:
protocols:
- http
serviceName: serverless
description: the serverless service
environment: release
endpoints:
- path: /users/{user_type}/{action}
method: GET
description: Serverless REST API
enableCORS: TRUE
serviceTimeout: 10
param:
- name: user_type
position: PATH
required: 'TRUE'
type: string
defaultValue: teacher
desc: mytest
- name: action
position: PATH
required: 'TRUE'
type: string
defaultValue: go
desc: mytest
```

View the `code/index.py` code, and you can see the API's parameter input and return logic:

```
# -*- coding: utf8 -*-

def teacher_go():
```

```
# todo: teacher_go action
return {
"result": "it is student_get action"
}

def student_go():
# todo: student_go action
return {
"result": "it is teacher_put action"
}

def student_come():
# todo: student_come action
return {
"result": "it is teacher_put action"
}

def main_handler(event, context):
print(str(event))
if event["pathParameters"]["user_type"] == "teacher":
if event["pathParameters"]["action"] == "go":
return teacher_go()
if event["pathParameters"]["user_type"] == "student":
if event["pathParameters"]["action"] == "go":
return student_go()
if event["pathParameters"]["action"] == "come":
return student_come()
```

3. Deploy

Deploy by running the `sls deploy` command, and you can add the `--debug` parameter to view the information during the deployment process:

If you have not logged in to or signed up for a Tencent Cloud account, you can directly log in or sign up:

```
$ sls deploy

serverless ⚡ framework
Action: "deploy" - Stage: "dev" - App: "scfApp" - Instance: "myRestAPI"

FunctionName: myRestAPI
Description: My Serverless Function
Namespace: default
Runtime: Python3.6
Handler: index.main_handler
MemorySize: 128
```

```
Triggers:
apigw:
- http://service-jyl9i6mc-1258834142.gz.apigw.tencentcs.com/release/users/{user_type}/{action}

31s › myRestAPI › Success
```

4. Test

You can run the following commands to test the returned result of the RESTful API:

> If you have not installed `curl` on Windows, you can directly open the corresponding link in a browser to view the returned result.

```
$ curl -XGET http://service-9t28e0tg-1250000000.gz.apigw.tencentcs.com/release/users/teacher/go

{"result": "it is student_get action"}
```

```
$ curl -PUT http://service-9t28e0tg-1250000000.gz.apigw.tencentcs.com/release/users/student/go

{"result": "it is teacher_put action"}
```

5. Remove

You can run the following command to remove the RESTful API application:

```
$ sls remove --debug

DEBUG — Flushing template state and removing all components.
DEBUG — Removing any previously deployed API. api-37gk3l8q
DEBUG — Removing any previously deployed service. service-9t28e0tg
DEBUG — Removing function
DEBUG — Request id
DEBUG — Removed function myRestAPI successful

7s » myRestAPI » done
```

Account configuration (optional)

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
$ touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

- If you don't have a Tencent Cloud account yet, please sign up first.
- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

Deploying Python Flask

Last updated : 2021-01-11 10:57:37

Overview

Tencent Cloud Flask Serverless Component supports deploying RESTful API services but does not support Flask Command.

> ⚠ **Note :**
>
> Any Python server framework that supports Web Server Gateway Interface (WSGI) can be deployed through this component, such as Falcon.

Prerequisites

1. Before using this component, please make sure that you have installed the Python environment locally.
2. Initialize a Flask project first and then add `Flask` and `werkzeug` to the dependent file `requirements.txt` as follows:

```
Flask==1.0.2
werkzeug==0.16.0
```

Meanwhile, add the API service `app.py`. The following code is for reference only:

```
from flask import Flask, jsonify
app = Flask(__name__)
@app.route("/")
def index():
return "Hello Flask"
@app.route("/users")
def users():
users = [{'name': 'test1'}, {'name': 'test2'}]
return jsonify(data=users)
@app.route("/users/<id>")
def user(id):
return jsonify(data={'name': 'test1'})
```

Directions

> **Note :**
>
> The following steps are mainly for deployment on the command line. For deployment in the console, please see Console Deployment Guide.

1. Install Serverless CLI

Use npm to install Serverless CLI globally:

```
npm install -g serverless
```

2. Initialize the Flask template project (optional)

If you don't have a local Flask project, you can initialize a Flask project with the following commands (if you already have one, you can ignore this step):

```
serverless init flask-starter --name example
cd example
```

3. Configure the .yml file

Create a `serverless.yml` file in the project root directory and paste the following configuration template into it to implement basic project configuration.

> **Note :**
>
> You can add more configuration items in `serverless.yml` based on your actual deployment needs. For more information on the configuration of the .yml file, please see Flask Component Configuration.

```
touch serverless.yml
```

```
#serverless.yml
component: flask
name: flashDemo
stage: dev

inputs:
src:
```

```
hook: 'pip install -r requirements.txt -t ./'
dist: ./
exclude:
- .env
region: ap-guangzhou
runtime: Python3.6
apigatewayConf:
protocols:
- http
- https
environment: release
```

4. Deploy the application

Deploy by running the `sls deploy` command, and you can add the `--debug` parameter to view the information during the deployment process:

```
sls deploy --debug
```

After the deployment is completed, access the application by accessing the output API Gateway link.

5. Monitor the OPS

After the deployment is completed, you can log in to the Serverless Framework console to view the basic information of the application and monitor logs.

Account configuration (optional)

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

Note :

- If you don't have a Tencent Cloud account yet, please sign up first.
- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

Deploying Koa Framework

Last updated : 2021-01-15 15:54:11

Overview

The Koa component can help you quickly and conveniently create, configure, and manage a Koa framework in Tencent Cloud with the aid of the serverless-tencent basic components (such as API Gateway component and SCF component).

> ⚠ **Note :**
>
> We recommend you use Node.js 10.0 or above; otherwise, Component v2 may report errors during deployment.

Migration Prerequisites

- Serverless Framework 1.67.2 or above has been installed.
- You have registered a Tencent Cloud account and completed identity verification.

> ⓘ **Note :**
>
> If your account is a **Tencent Cloud sub-account**, please get the authorization from the root account first as instructed in Account and Permission Configuration.

Directions

> ⓘ **Note :**
>
> The following steps are mainly for deployment on the command line. For deployment in the console, please see Console Deployment Guide.

1. Initialize the Koa template project (optional)

If you don't have a local Koa project, you can quickly create a Koa project template with the following command (if you already have one, you can ignore this step):

```
serverless init koa-starter --name example
cd example
```

2. Modify the project code

Open the entry file `sls.js` (or `app.js`) of the Koa project, comment out the local listening port, and export the default Koa application:

```
// sls.js

const koa = require('koa');
const app = koa();

// *****

// Comment out the local listening port
// app.listen(3000);

// Export the Koa application
module.exports = app;
```

3. Generate a .yml file and deploy

After modifying the code, you can run the `sls deploy` command, and Serverless Framework will automatically generate a basic `serverless.yml` file and complete the deployment to quickly migrate the Koa framework application.

The generated default configuration file is as follows:

```
component: koa
name: koaDemo
app: appDemo

inputs:
entryFile: sls.js # Use the actual entry file name
src: ./
region: ap-guangzhou
runtime: Nodejs10.15
apigatewayConf:
protocols:
- http
- https
environment: release
```

After the deployment is completed, access the application by accessing the output API Gateway link.

4. Modify the .yml file

You can add more configuration items in `serverless.yml` based on your actual deployment needs and run `sls deploy` for redeployment.

For more information on the configuration of the .yml file, please see Koa Component Configuration.

5. Monitor the OPS

After the deployment is completed, you can log in to the SSR console to view the basic information of the application and monitor logs.

Account configuration (optional)

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

Note :

- If you don't have a Tencent Cloud account yet, please sign up first.
- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

More Components

You can view more component information in the repository of Serverless Components.

Deploying Egg.js Framework

Last updated : 2021-01-08 16:46:44

Overview

Tencent Cloud Egg.js Serverless Component supports deploying RESTful API services.

Prerequisites

Initialize Egg.js project

```
$ mkdir serverless-egg && cd serverless-egg
$ npm init egg src --type=simple
$ cd src && npm install
```

Directions

1. Install

Use npm to install Serverless CLI globally:

```
$ npm install -g serverless
```

2. Configure

Create a `serverless.yml` file in the project root directory:

```
$ touch serverless.yml
```

Configure the `serverless.yml` file as follows (for more information on the configuration of the .yml file, please see Configuration Information):

```
component: egg
name: eggDemo
app: appDemo

inputs:
src: ./
```

```
region: ap-guangzhou
runtime: Nodejs10.15
apigatewayConf:
protocols:
- http
- https
environment: release
```

3. Deploy

If you have not logged in to or signed up for a Tencent Cloud account, you can directly log in or sign up by scanning the QR code on the command line with **WeChat**.

Deploy by running the `sls deploy` command, and you can add the `--debug` parameter to view the information during the deployment process:

> **Note :**
>
> `sls` is short for the `serverless` command.

```
$ sls deploy

MyComponent:
region: ap-beijing
functionName: egg-function
apiGatewayServiceId: service-n5m5e8x3
url: https://service-n5m5e8x3-1251971143.bj.apigw.tencentcs.com/release/

32s › MyComponent › done
```

4. Remove

You can run the following commands to remove the deployed API Gateway and SCF services:

```
$ sls remove --debug

DEBUG — Flushing template state and removing all components.
DEBUG — Removing function
DEBUG — Request id
DEBUG — Removed function egg-function successful
DEBUG — Removing any previously deployed API. api-cmkhknda
DEBUG — Removing any previously deployed service. service-n5m5e8x3

8s › MyComponent › done
```

Account configuration (optional)

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
$ touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

Note :

- If you don't have a Tencent Cloud account yet, please sign up first.
- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

More Resources

FAQs

1. **What should I do if the `app/public` deployment directory does not exist?**

   Generally, when you initialize an Egg.js project, the `app/public` directory will be created automatically. If this directory is empty during packaging and compressing, it will not exist after the deployment; instead, it will automatically be created when the Egg.js project is started; however, SCF will not have the permission to manipulate it. Therefore, we recommend you create an empty folder `.gitkeep` in the `app/public` directory to solve this problem.

2. **What should I do if a failure to find the dependent module is reported after deployment with Layer?**

   For more information, please see here.

More components

You can view more component information in the repository of Serverless Components.

Deploying Next.js Application

Last updated : 2021-01-15 15:43:46

Overview

Tencent Cloud Next.js component uses Tencent Serverless Framework to implement "zero" configuration, convenient development, and rapid deployment of your webpage application in the Next.js framework based on Tencent Cloud Serverless services (API Gateway, SCF, etc.). The Next.js component supports a rich set of configuration extensions and provides easy-to-use, practical, and cost-effective development/hosting capabilities for webpage application projects.

Next.js features:

- **Pay-as-you-go billing**: fees are charged based on the request usage, and you don't need to pay anything if there is no request.
- **"Zero" configuration**: you only need to write project code and then directly deploy it, and Serverless Framework will take care of all the configuration work.
- **Fast deployment**: you can deploy your entire application in just a few seconds.
- **Real-time log**: you can view the business status through the output of the real-time log, which makes it easy for you to develop applications directly in the cloud.
- **Cloud debugging**: you can directly debug your project in the cloud to avoid problems caused by differences from the local environment.
- **Convenient collaboration**: the status information and deployment logs in the cloud-based console make multi-person collaborative development easier.

Migration Prerequisites

- Serverless Framework 1.67.2 or above has been installed.
- You have registered a Tencent Cloud account and completed identity verification.

Note :

If your account is a **Tencent Cloud sub-account**, please get the authorization from the root account first as instructed in Account and Permission Configuration.

Architecture

The Next.js component will use the following Serverless services in your Tencent Cloud account:

- **API Gateway**: it will receive external requests and forward them to the SCF function.
- **SCF**: it will carry the Next.js application.
- **CAM**: this component will create a default CAM role for authorizing access to associated resources.
- **COS**: to ensure the upload speed and quality, when the function is compressed and the code is uploaded, the code package will be stored in a specifically named COS bucket by default.

Directions

> **Note :**
>
> The following steps are mainly for deployment on the command line. For deployment in the console, please see Console Deployment Guide.

1. Initialize the Next.js template project (optional)

If you don't have a local Next.js project, you can quickly create a Next.js project template with the following command (if you already have one, you can ignore this step):

```
serverless init nextjs-starter --name example
cd example
```

2. Modify the entry function code (optional)

If the project uses a custom Node.js service, such as the Express or Koa framework, you need to modify the entry file `sls.js` (or `app.js`) and export the corresponding framework application (otherwise, you can skip this step). Click here to view the modification template.

3. Build the project

```
npm run build
```

4. Generate a .yml file and deploy

After modifying the code, you can run the `sls deploy` command, and Serverless Framework will automatically generate a basic `serverless.yml` file and complete the deployment to quickly migrate the Next.js framework application.

The generated default configuration file is as follows:

```
component: nextjs
name: nextjsDemo
app: appDemo

inputs:
src: ./
exclude:
- .env
region: ap-guangzhou
runtime: Nodejs10.15
apigatewayConf:
protocols:
- http
- https
environment: release
```

After the deployment is completed, access the application by accessing the output API Gateway link.

5. View deployment status

In the directory where the `serverless.yml` file is located, run the following command to check the deployment status:

```
$ serverless info
```

Modification Template for Project Migration

- Express template

```
const express = require('express')
const next = require('next')
// not report route for custom monitor
const noReportRoutes = ['/_next', '/static', '/favicon.ico']
async function createServer() {
const app = next({ dev: false })
const handle = app.getRequestHandler()
await app.prepare()
const server = express()
server.all('*', (req, res) => {
noReportRoutes.forEach((route) => {
if (req.path.indexOf(route) !== -1) {
req.__SLS_NO_REPORT__ = true
}
```

```
})
return handle(req, res)
})
// define binary type for response
// if includes, will return base64 encoded, very useful for images
server.binaryTypes = ['*/*']
return server
}
module.exports = createServer
```

- Koa template

```
const Koa = require('koa')
const next = require('next')
async function createServer() {
const app = next({ dev: false })
const handle = app.getRequestHandler()
const server = new Koa()
server.use((ctx) => {
ctx.status = 200
ctx.respond = false
ctx.req.ctx = ctx
return handle(ctx.req, ctx.res)
})
// define binary type for response
// if includes, will return base64 encoded, very useful for images
server.binaryTypes = ['*/*']
return server
}
module.exports = createServer
```

Custom monitoring

When you deploy the Next.js application, if you don't specify the `role` in `serverless.yml`, the system will try to bind `QCS_SCFExcuteRole` by default and enable custom monitoring to help you collect the statistics of application metrics. For projects with custom entry files, routes except those with `/_next` and `/static` will be reported by default.

If you want to report the performance of custom routes, you can customize the `sls.js` entry file. For routes that do not need to be reported, set the `__SLS_NO_REPORT__` attribute value to `true` in the `req` object of the Express service; for example:

```
server.get('/no-report', (req, res) => {
req.__SLS_NO_REPORT__ = true
return handle(req, res)
})
```

After configuration, custom monitoring metrics will not be reported when users access the `GET /no-report` route.

More Resources

Account configuration

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
$ touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

Note :

- If you don't have a Tencent Cloud account yet, please sign up first.
- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

More components

You can view more component information in the repository of Serverless Components.

FAQs

Why is an entry file no longer needed?

In the previous version, to use this component, you need to add an `sls.js` file in the project root directory. In the current version, this is taken care of by the component, so you do not need to deal with it separately. For more information, please see the GitHub documentation.

Deploying Express.js Application

Last updated : 2021-01-15 15:56:22

Overview

Tencent Cloud Express component uses Tencent Serverless Framework. Based on serverless services (gateways, functions, etc.) in the cloud, it can implement "zero" configuration, convenient development, and rapid deployment of your Express application. The Express component supports a rich set of configuration extensions and provides the easiest-to-use, low-cost, and elastically scalable cloud-based Express project development and hosting capabilities.

Express.js features:

- **Pay-as-you-go billing**: fees are charged based on the request usage, and you don't need to pay anything if there is no request.
- **"Zero" configuration**: you only need to write project code and then deploy it, and the Serverless Framework will take care of all the configuration work.
- **Fast deployment**: you can deploy your entire Express application in just a few seconds.
- **Real-time log**: you can view the business status through the output of the real-time log, which makes it easy for you to develop applications directly in the cloud.
- **Cloud debugging**: it supports quick cloud debugging for the Node.js framework, which shields differences in the local environment.
- **Convenient collaboration**: the status information and deployment logs in the cloud make multi-person collaborative development easier.
- **Custom domain name**: it supports configuration of custom domain names and HTTPS access.

Through the Serverless Framework Express component, you can quickly migrate traditional local Express applications to the serverless function platform.

Migration Prerequisites

- Serverless Framework 1.67.2 or above has been installed.
- You have registered a Tencent Cloud account and completed identity verification.

Note :

> If your account is a **Tencent Cloud sub-account**, please get the authorization from the root account first as instructed in Account and Permission Configuration.

Architecture

The Express component will use the following Serverless services in your Tencent Cloud account:

- **API Gateway**: it will receive external requests and forward them to the SCF function.
- **SCF**: it will carry the Express application.
- **CAM**: this component will create a default CAM role for authorizing access to associated resources.
- **COS**: to ensure the upload speed and quality, when the function is compressed and the code is uploaded, the code package will be stored in a specifically named COS bucket by default.

Directions

> **Note :**
>
> The following steps are mainly for deployment on the command line. For deployment in the console, please see Console Deployment Guide.

1. Initialize the Express template project (optional)

If you don't have a local Express project, you can quickly create an Express project template with the following command (if you already have one, you can ignore this step):

```
serverless init express-starter --name example
cd example
```

2. Modify the project code

Open the entry file `sls.js` (or `app.js`) of the Express project, comment out the local listening port, and export the default Express application:

```
// sls.js

const express = require('express');
const app = express();

// *****
```

```
// Comment out the local listening port
// app.listen(3000);

// Export the Express application
module.exports = app;
```

3. Generate a .yml file and deploy

After modifying the code, you can run the `sls deploy` command, and Serverless Framework will automatically generate a basic `serverless.yml` file and complete the deployment to quickly migrate the Express framework application.

The generated default configuration file is as follows:

```
component: express
name: expressDemo
app: appDemo

inputs:
entryFile: sls.js # Use the actual entry file name
src: ./
region: ap-guangzhou
runtime: Nodejs10.15
apigatewayConf:
protocols:
- http
- https
environment: release
```

After the deployment is completed, access the application by accessing the output API Gateway link.

4. Modify the .yml configuration file

You can add more configuration items in `serverless.yml` based on your actual deployment needs and run `sls deploy` for redeployment.

For more information on the configuration of the .yml file, please see Express Component Configuration.

5. Monitor the OPS

After the deployment is completed, you can log in to the Serverless Framework console to view the basic information of the application and monitor logs.

Account Configuration

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
$ touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

Note：

- If you don't have a Tencent Cloud account yet, please sign up first.
- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

Deploying Nuxt.js Application

Last updated : 2020-07-21 18:38:22

Operation Scenarios

Tencent Cloud Nuxt.js component uses **Tencent Serverless Framework** to implement "zero" configuration, convenient development, and rapid deployment of your webpage application in the Nuxt.js framework based on Tencent Cloud Serverless services (API Gateway, SCF, etc.). The Nuxt.js component supports a rich set of configuration extensions and provides easy-to-use, practical, and cost-effective development/hosting capabilities for webpage application projects.

Features:

- **Pay-as-you-go billing**: fees are charged based on the request usage, and you don't need to pay anything if there is no request.
- **"Zero" configuration**: you only need to write project code and then directly deploy it, and Serverless Framework will take care of all the configuration work.
- **Fast deployment**: you can deploy your application in just a few seconds.
- **Real-time log**: you can view the business status through the output of the real-time log, which makes it easy for you to develop applications directly in the cloud.
- **Cloud debugging**: you can directly debug your project in the cloud to avoid problems caused by differences from the local environment.
- **Convenient collaboration**: the status information and deployment logs in the cloud-based console make multi-person collaborative development easier.

Prerequisites

Initialize Nuxt.js project

Create a root directory and initialize a Nuxt.js project locally:

```
$ mkdir serverless-nuxtjs && cd serverless-nuxtjs
$ npx create-nuxt-app src
```

⚠ Note :

The Nuxt project in this tutorial is built with JavaScript and the npm installation package. Please select the appropriate options when initializing the project.

> ⚠ **Note :**
>
> You are recommended to use Node.js 10.0 or above; otherwise, Component v2 may report errors during deployment.

Directions

1. Install

Use npm to install Serverless CLI globally:

```
$ npm install -g serverless
```

2. Configure

Create a `serverless.yml` file in the project root directory:

```
$ touch serverless.yml
```

Configure `serverless.yml` as follows:

```
# serverless.yml
component: nuxtjs # Component name, which is required. `nuxtjs` is used in this example
name: nuxtjsDemo # Instance name, which is required
org: orgDemo # Organization information, which is optional. The default value is the `APPID` of y
our Tencent Cloud account.
app: appDemo # Nuxt.js application name, which is optional
stage: dev # Information for identifying environment, which is optional. The default value is `de
v`

inputs:
src:
src: ./src
exclude:
- .env
region: ap-guangzhou
runtime: Nodejs10.15
apigatewayConf:
protocols:
- http
- https
environment: release
```

View more configurations and descriptions >>

3. Deploy

3.1. Construct static resources

Enter the Nuxt.js project directory and construct static resources:

```
$ cd src && npm run build
```

3.2. Deploy into the cloud

Go back to the project root directory where the `serverless.yml` file is and run the following command for deployment:

```
# Enter the project directory `serverless-nuxtjs`
$ sls deploy

serverless ⚡ framework
Action: "deploy" - Stage: "dev" - App: "appDemo" - Instance: "nuxtjsDemo"

region: ap-guangzhou
apigw:
serviceId: service-4v2jx72g
subDomain: service-4v2jx72g-1258834142.gz.apigw.tencentcs.com
environment: release
url: https://xxxxxx.gz.apigw.tencentcs.com/release/
scf:
functionName: nuxtjs_component_mm518kl
runtime: Nodejs10.15
namespace: default

139s › nuxtjsDemo › Success
```

Note :

If you want to view more information on the deployment process, you can run the `sls deploy --debug` command to view the real-time log information during the deployment process (`sls` is an abbreviation for the `serverless` command).

4. Debug

After the Nuxt.js application is deployed, the project can be further developed through the debugging feature to create an application for the production environment. After modifying and

updating the code locally, you don't need to run the `serverless deploy` command every time for repeated deployment. Instead, you can run the `serverless dev` command to directly detect and automatically upload changes in the local code.
You can enable debugging by running the `serverless dev` command in the directory where the `serverless.yml` file is located.
`serverless dev` also supports real-time outputting of cloud logs. After each deployment, you can access the project to output invocation logs in real time on the command line, which makes it easy for you to view business conditions and troubleshoot issues.

5. View deployment status

In the directory where the `serverless.yml` file is located, run the following command to check the deployment status:

```
$ serverless info
```

6. Remove

In the directory where the `serverless.yml` file is located, run the following command to remove the deployed API gateway. After removal, this component will delete all related resources created during deployment in the cloud.

```
$ sls remove
```

Similar to the deployment process, you can run the `sls remove --debug` command to view real-time log information during the removal process (`sls` is an abbreviation for the `serverless` command).

More Resources

Account configuration

Currently, you can scan a QR code to log in to the CLI by default. If you want to configure persistent environment variables/key information, you can also create a local `.env` file:

```
$ touch .env # Tencent Cloud configuration information
```

Configure Tencent Cloud's `SecretId` and `SecretKey` information in the `.env` file and save it:

Note :

- If you don't have a Tencent Cloud account yet, please sign up first.

- If you already have a Tencent Cloud account, you can get `SecretId` and `SecretKey` in API Key Management.

```
# .env
TENCENT_SECRET_ID=123
TENCENT_SECRET_KEY=123
```

⚠ Note :

When logging in with an IP outside Mainland China, you need to add `SERVERLESS_PLATFORM_VENDOR=tencent` in the `.env` file to make `sls` use the `tencent` component by default.

More components

You can view more component information in the repository of Serverless Components.

FAQs

Why is an entry file no longer needed?

In the previous version, to use this component, you need to add an `sls.js` file in the project root directory. In the current version, this is taken care of by the component, so you do not need to deal with it separately. For more information, please see the GitHub documentation.