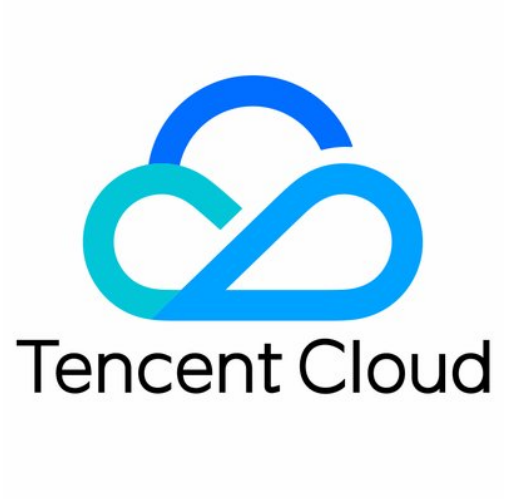
Serverless Application Center

Practical Tutorial

Product Documentation



Contents

Practical Tutorial
Developing and Launching Serverless Application
Project Development

Last updated：2024-12-02 11:16:20

Overview

This document uses deploying an Express website with the `tencent-express` component as an example to describe how to use Serverless Framework to develop, manage, deploy, and publish a project. You can see the demo here.

Project development may involve the following branches:

Branch Type	Description
master	It is used to deploy a production environment
testing	It is used for testing in a testing environment
dev	It is used for daily development
feature-xxx	It is used to add a new feature; for example, different developers can pull different feature branches from `dev` for development
hotfix-xxx	It is used to fix an urgent bug

Directions

Project initialization

1. Create an Express project as instructed in Deploying Express.js Application and modify the YML file content as follows:

```
#serverless.yml
app: expressDemoApp # Application name, which is the component instance name by def
stage: ${env:STAGE} # Parameter used to isolate the development environment, which
```

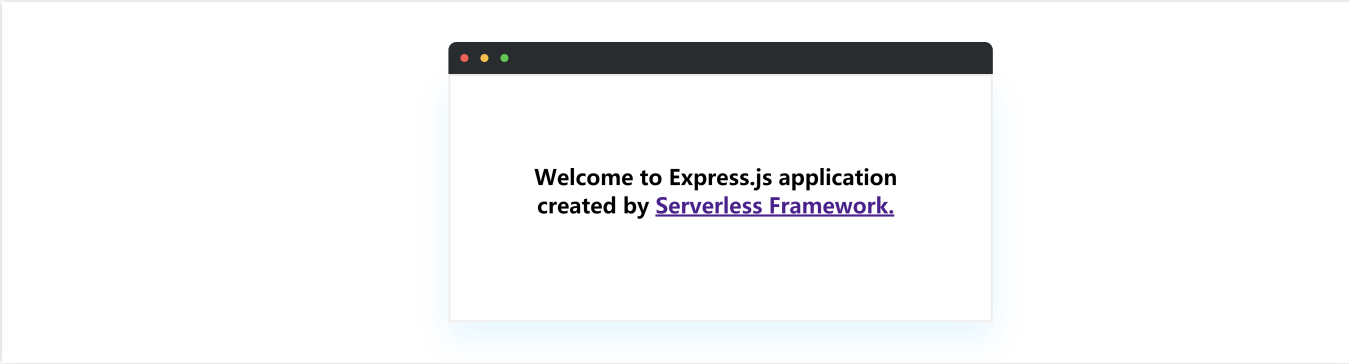
```
component: express # Name of the imported component, which is required. The `expres
name: expressDemo # Name of the instance created by the component, which is require

inputs:
  src:
    src: ./
    exclude:
      - .env
  region: ap-guangzhou
  runtime: Nodejs10.15
  functionName: ${name}-${stage}-${app} # Function name
  apigatewayConf:
    protocols:
      - http
      - https
    environment: release
```

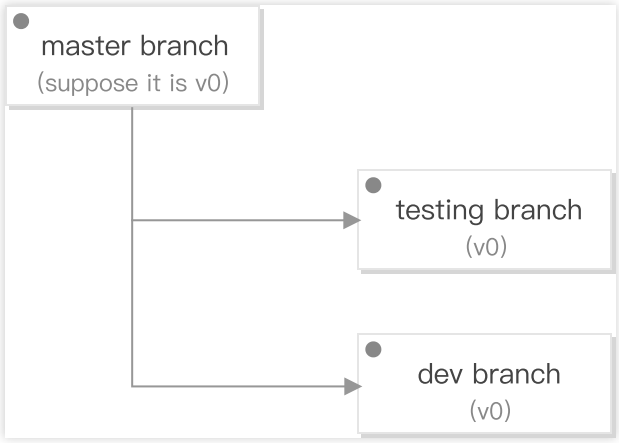
2. Configure the following content in the .env file in the project root directory:

```
TENCENT_SECRET_ID=xxxxxxxxxx # `SecretId` of your account
TENCENT_SECRET_KEY=xxxxxxxx # `SecretKey` of your account
STAGE=prod # `STAGE` is the `prod` environment. You can also run `sls deploy --stag
```

3. After the deployment by running `sls deploy` succeeds, access the generated URL as shown below:



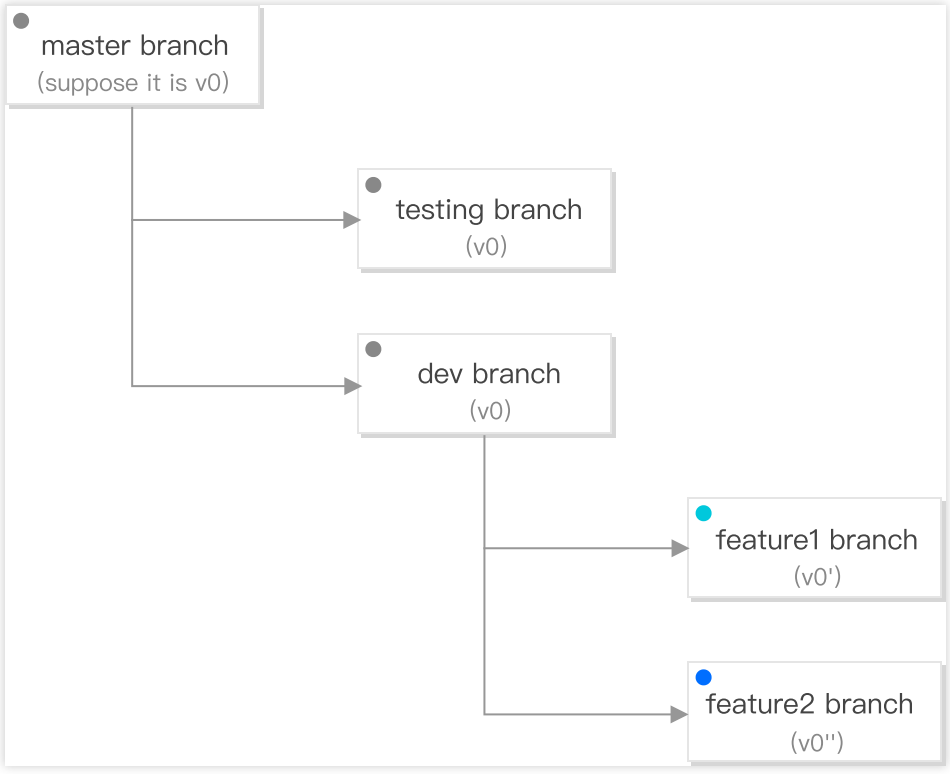
4. Create a remote repository (sample), submit the project code to the remote `master` branch, and create `testing` and `dev` branches. In this way, the code of the three branches is on the same version (suppose it is v0).



Development and testing

Background

In this stage, a feature module needs to be developed. Suppose two developers are needed: Tom and Jorge, who create `feature1` and `feature2` feature branches from `dev` (v0) for development, respectively.



Tom starts developing `feature1`. In this example, a `feature.html` file is added, and "This is a new feature 1." is written in it.

Development

1. Add router configuration in the `sls.js` file:

```
// Routes
app.get(`/feature`, (req, res) => {
 res.sendFile(path.join(__dirname, 'feature.html'))
})
```

2. Add `feature.html` :

```
<!DOCTYPE html>
<html lang="en">
 <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Serverless Component - Express.js</title>
 </head>
 <body>
  <h1>
   This is a new feature 1.
  </h1>
 </body>
</html>
```

3. Set a stage in the .env file so as to get an independent runtime and debugging environment during the development. For example, Tom configures the .env file in the project directory of `serverless.yml` as follows:

```
TENCENT_SECRET_ID=xxxxxxxxxx
TENCENT_SECRET_KEY=xxxxxxxx
STAGE=feature1
```

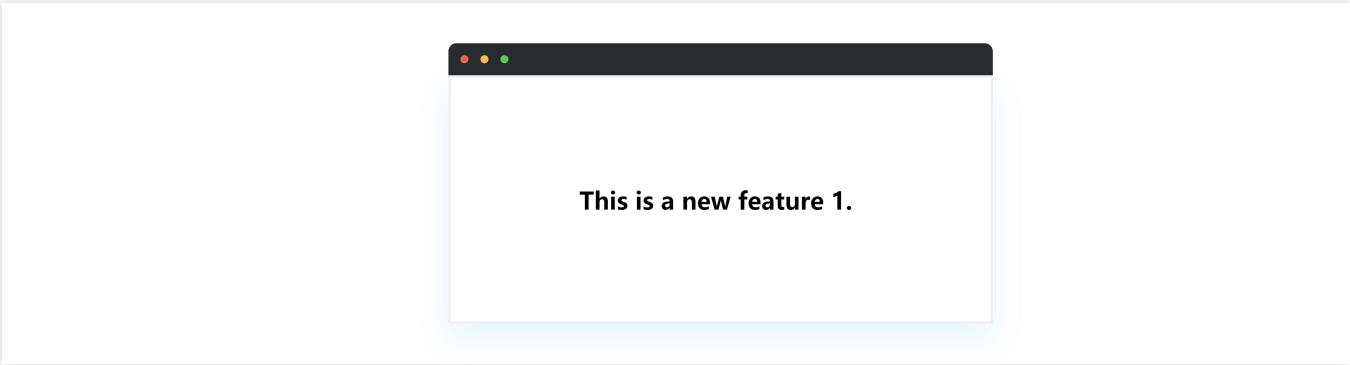
4. After the deployment by running `sls deploy` succeeds, the following content will be returned:

```
region: ap-guangzhou
apigw:
  serviceId:   service-xxxxxx
  subDomain:   service-xxxxxx-123456789.gz.apigw.tencentcs.com
  environment: release
  url:         https://service-xxxxxx-123456789.gz.apigw.tencentcs.com/release/
scf:
  functionName: express-demo-feature1
  runtime:      Nodejs10.15
  namespace:    default
  lastVersion:  $LATEST
  traffic:      1

Full details: https://serverless.cloud.tencent.com/instances/expressDemoApp%3Afeatu

10s » expressDemo » Success
```

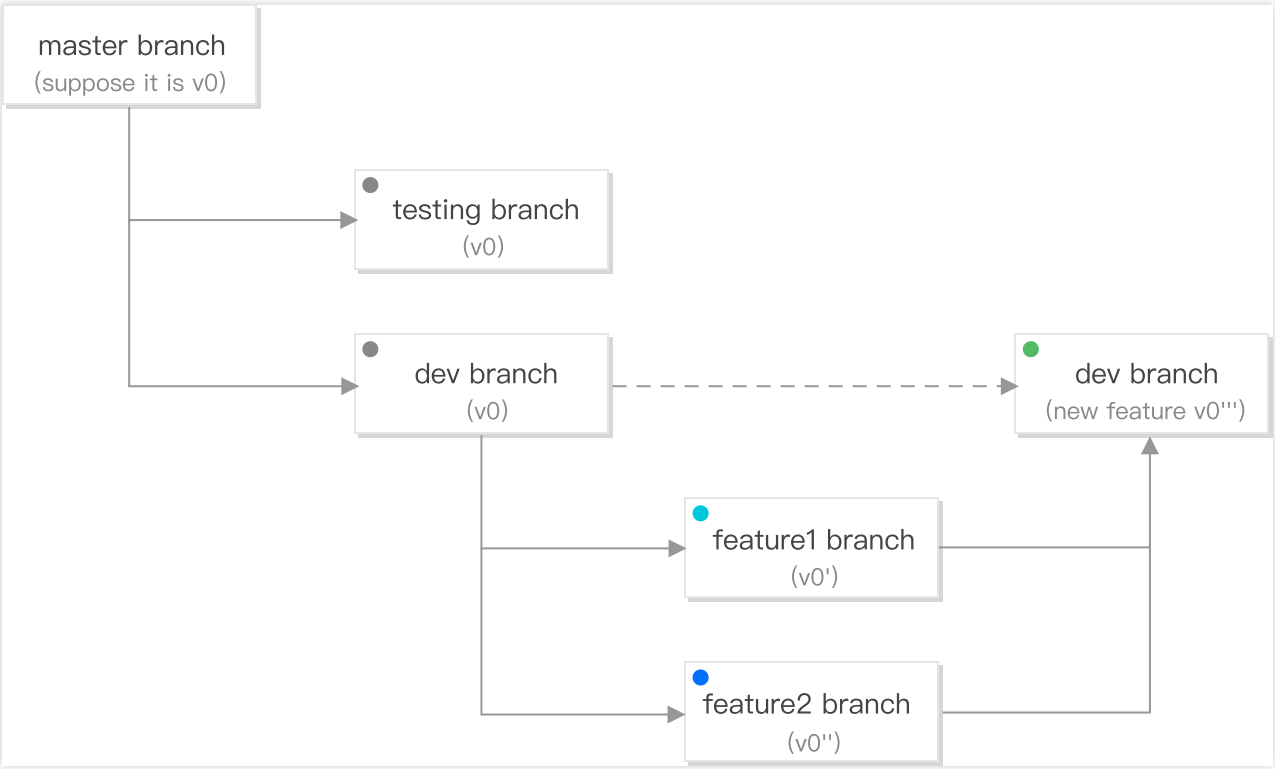
5. Access the generated URL (https://service-xxxxxx-123456789.gz.apigw.tencentcs.com/release/feature) as shown below:



At this point, Tom has completed feature development and successfully tested the feature.
Suppose Jorge has also completed feature development and successfully tested the feature. In this example, a `feature.html` file is added, and "This is a new feature 2." is written in it.

Joint testing

1. The two developers merge their feature branch code into the `dev` branch (conflicts may occur and need to be solved manually).

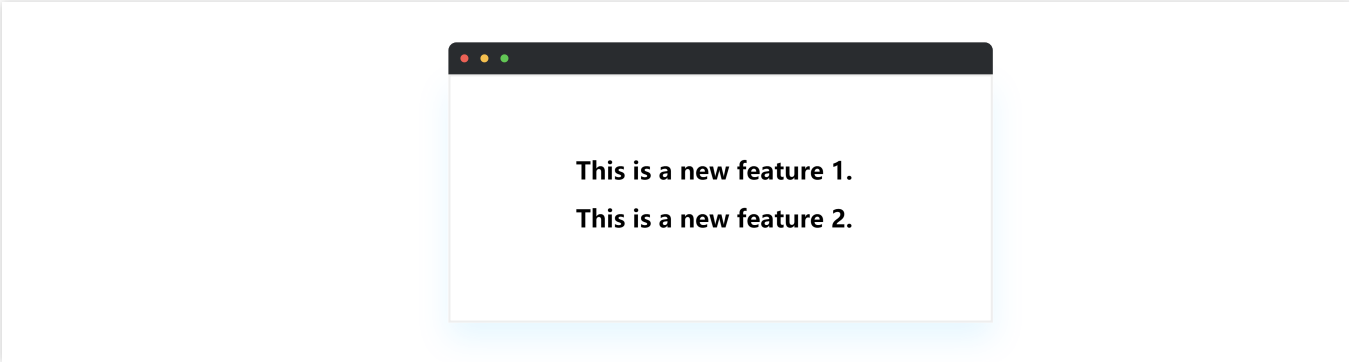


2. Carry out joint testing in `dev` . The .env file is configured as follows in the joint testing environment:

```
TENCENT_SECRET_ID=xxxxxxxxxx
```

```
TENCENT_SECRET_KEY=xxxxxxxx
STAGE=dev
```

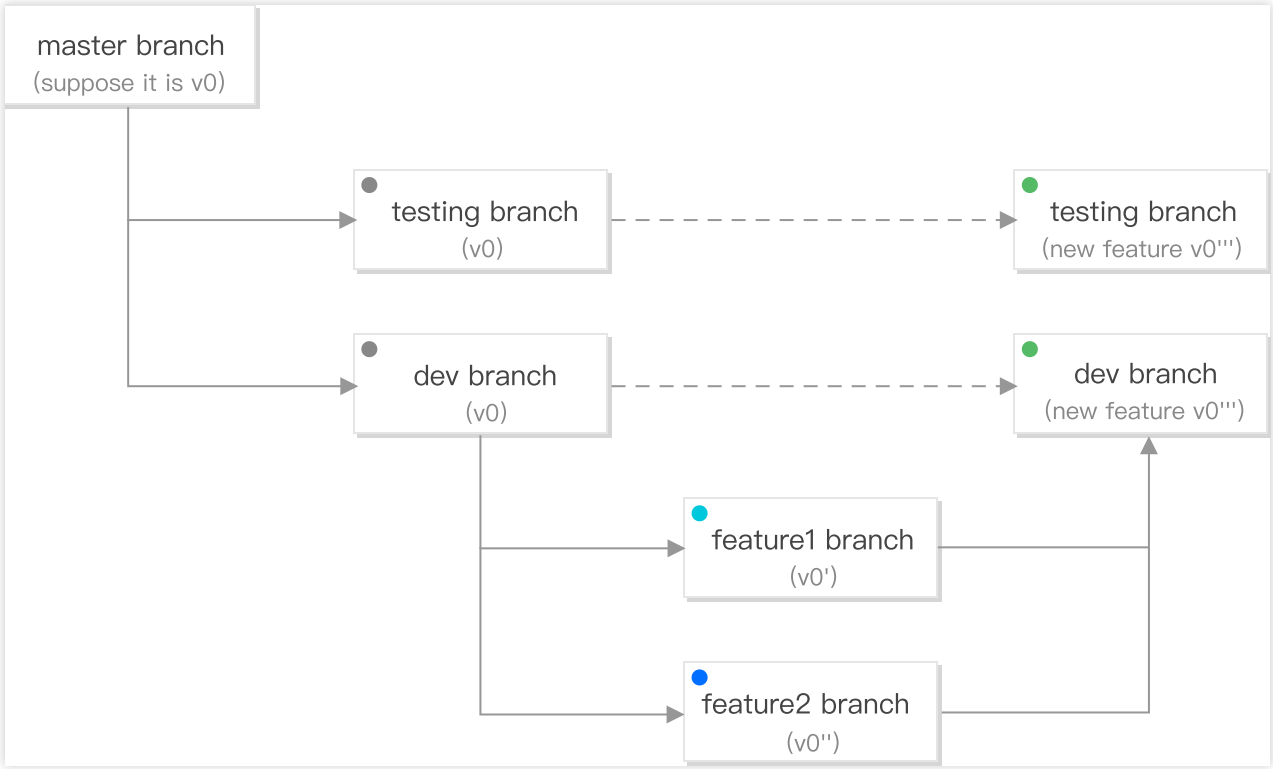
3. After running `sls deploy` to deploy the joint testing environment, access the URL (https://service-xxxxxx-123456789.gz.apigw.tencentcs.com/release/feature) as shown below:



At this point, the joint testing is completed, and the development of the entire functionality is also completed.

Testing

1. Merge the jointly tested `dev` branch into `testing` code to enter the testing stage.



2. Configure the .env file in the testing environment as follows:

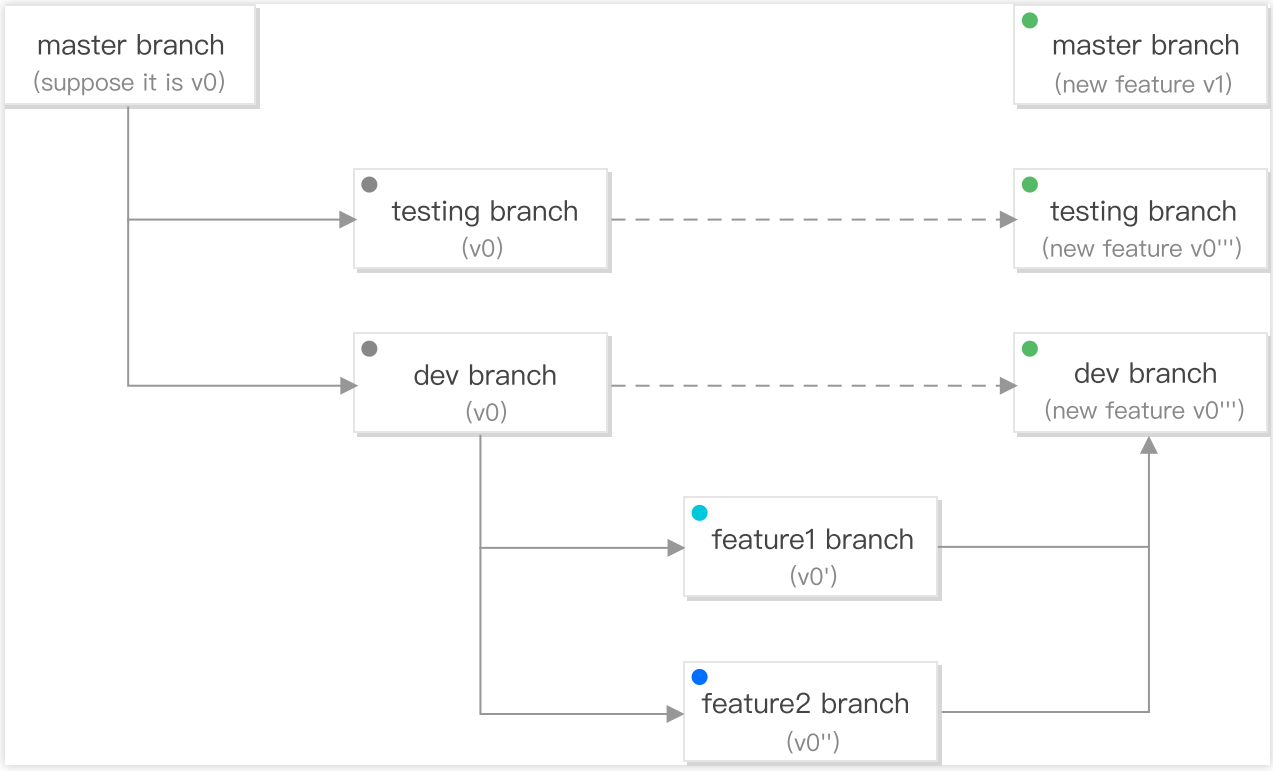
```
TENCENT_SECRET_ID=xxxxxxxxxx
TENCENT_SECRET_KEY=xxxxxxxx
```

```
STAGE=testing
```

3. After the deployment by running `sls deploy` succeeds, the testing personnel will carry out relevant tests until the features are stable in the testing.

Release and launch

After the test is passed, merge the testing code into the `master` branch and prepare for release and launch.



Set the .env file in the production environment as follows:

```
TENCENT_SECRET_ID=xxxxxxxxxx
TENCENT_SECRET_KEY=xxxxxxxx
STAGE=prod
```

Run the deployment command:

```
sls deploy
```

At this point, a `serverless-express` project has been developed and published.

Grayscale Release

Last updated：2024-12-02 11:16:20

Overview

During update and switch of your business version, to ensure that the business in the production environment is stable, we recommend you use grayscale release:
This document uses a deployed Express project as an example to describe how to perform grayscale release.

Prerequisites

You have developed a project.

Directions

1. Set the .env file in the production environment:

```
TENCENT_SECRET_ID=xxxxxxxxxx
TENCENT_SECRET_KEY=xxxxxxxx
STAGE=prod
```

2. Deploy the `$latest` version in the production environment and switch 10% traffic to it (90% traffic will be switched to the last published function version N):

```
sls deploy --inputs.traffic=0.1
```

3. Monitor the `$latest` version and switch 100% traffic to this version after it becomes stable:

```
sls deploy --inputs.traffic=1.0
```

4. After all traffic is successfully switched, the stable version needs to be marked, so that you can easily and quickly roll back to this version if a problem occurs in the production environment when a new feature is published. Deploy and publish the function version N+1 and switch all traffic to it:

```
sls deploy --inputs.publish --inputs.traffic=0
```

Automated Deployment

Last updated：2024-12-02 11:16:20

Overview

During serverless application development, we need to manually run deployment commands to deploy development projects to the cloud and automate the serverless application deployment by introducing some CI capabilities.

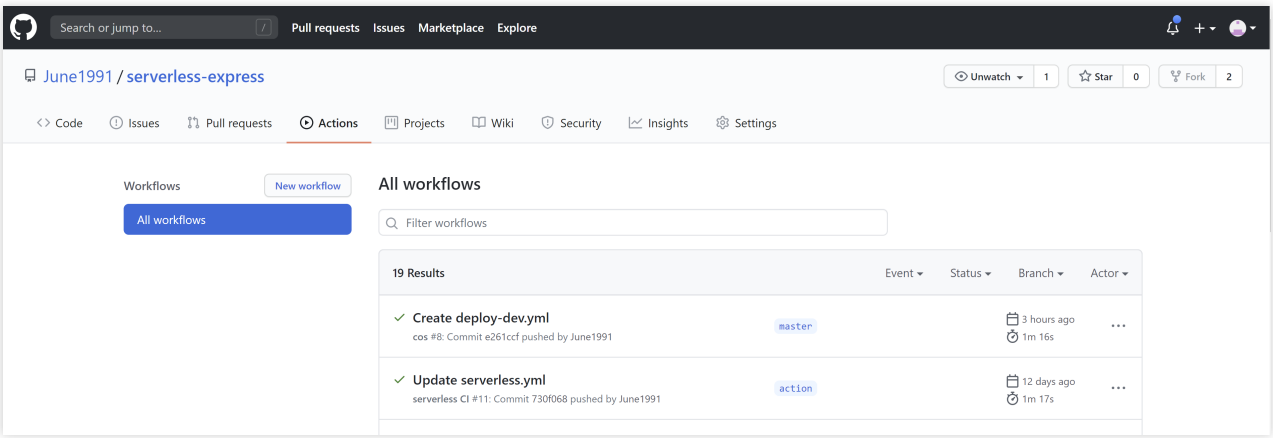
Automated Deployment Based on GitHub

Prerequisites

You have created a serverless application project. For how to create a serverless application project and create environments and branches, see Project Development.
You have hosted your serverless application project in GitHub.

Directions

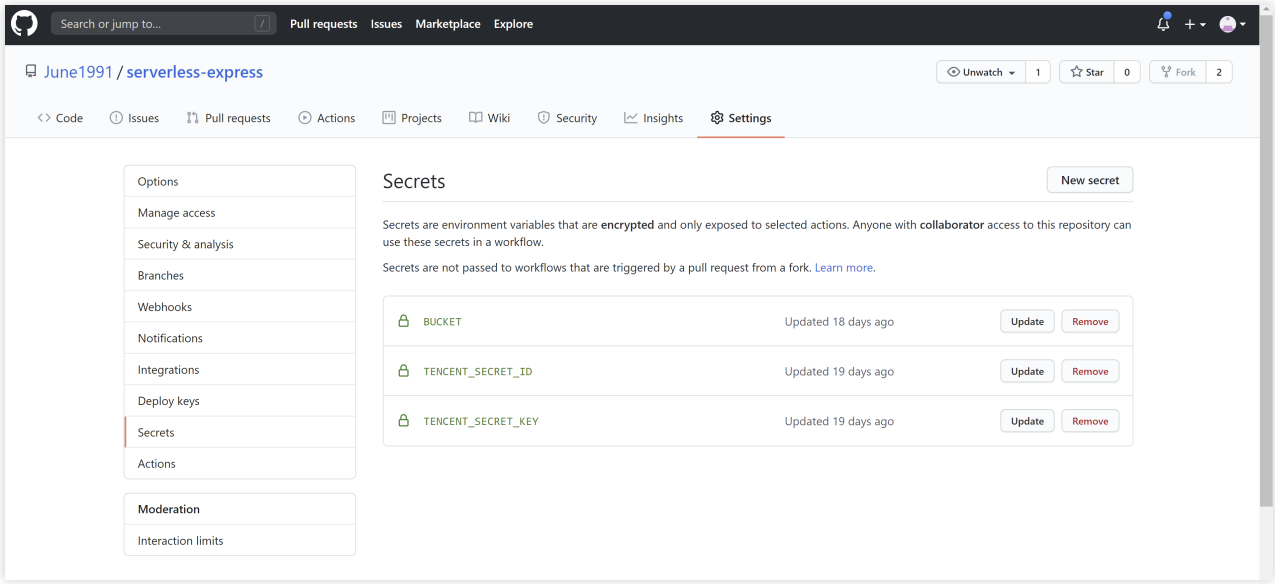
During the development and testing phase, if you want your code to be deployed automatically after each commit for ease of development, testing, and debugging, perform the following operations:
1. Select a branch for which you want to perform automated deployment (the `dev` branch is selected in this example).
2. Create your actions under the branch.



Note:

GitHub specifies that if an event occurs on a particular repository branch, the workflow file must be placed in that branch's repository.

3. Configure Tencent Cloud secrets (keys).



4. Configure action deployment.

```
# When the code is pushed to the `dev` branch, the current workflow will be execute
# For more information on configuration, visit https://docs.github.com/cn/actions/g
name: deploy serverless
on: # Configuration of the event and branch listened on
   push:
     branches:
       - dev
jobs:
   test: # Configure unit testing
     name: test
     runs-on: ubuntu-latest
     steps:
       - name: unit test
         run: ''
   deploy:
     name: deploy serverless
     runs-on: ubuntu-latest
     needs: [test]
     steps:
       - name: clone local repository
         uses: actions/checkout@v2
       - name: install serverless
         run: npm install -g serverless
```

```
      - name: install dependency
        run: npm install
      - name: build
        run: npm build
      - name: deploy serverless
        run: sls deploy --debug
        env: # Environment variable
          STAGE: dev # Your deployment environment
          SERVERLESS_PLATFORM_VENDOR: tencent # The serverless platform is `aws` b
          TENCENT_SECRET_ID: ${{ secrets.TENCENT_SECRET_ID }} # `secret ID` of you
          TENCENT_SECRET_KEY: ${{ secrets.TENCENT_SECRET_KEY }} # `secret key` of
```

After the above configuration, the code that you commit to the `dev` branch every time will be automatically deployed.

Automated Deployment Based on CODING

Prerequisites

You have signed up for a CODING account. If you are a Tencent Cloud user, you can use CODING DevOps to quickly register a CODING account.
You have created a serverless application project. For how to create a serverless application project and create environments and branches, see Project Development.
You have hosted your serverless application project on platforms such as CODING, GitHub, GitLab, and Gitee.

Overview

During the development and testing phase, if you want your code to be deployed automatically after each commit for ease of development, testing, and debugging, perform the following operations:
1. Create your CODING DevOps project.
2. Create a build plan and choose to customize a build process.
3. Configure the build plan.
3.1 Configure basic information. This example shows how to configure the GitHub repository `June1991/express-demo` .
3.2 Configure trigger rules. This example shows how to configure a rule to trigger build when code is pushed to the `dev` branch.
3.3 Configure environment variables. This example shows how to set the `STAGE` variable to the deployment environment `dev` and set `TENCENT_CLOUD_API_CRED` to the Tencent Cloud account key (key configuration path: **Project Settings** > **Developer Options** > **Credential Management** > **Enter Credential** > **Tencent Cloud API key**).
3.4 Configure the pipeline (process).

```
pipeline {
    agent any
    stages {
      stage('Check out') {
        steps {
          checkout([$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
              userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.CREDENT
        }
      }
      stage('Install dependencies') {
        steps {
          echo 'Installing dependencies...'
          sh 'npm i -g serverless'
          sh 'npm install'
          echo 'Dependencies installed.'
        }
      }
      stage('Deploy') {
        steps {
          echo 'Deploying...'

          withCredentials([
            cloudApi(
              credentialsId: "${env.TENCENT_CLOUD_API_CRED}",
              secretIdVariable: 'TENCENT_SECRET_ID',
              secretKeyVariable: 'TENCENT_SECRET_KEY'
            ),
          ]) {

               // Generate the credential file
               sh 'echo "TENCENT_SECRET_ID=${TENCENT_SECRET_ID}\\nTENCENT_SECRET_KEY
               // Deploy
               sh 'sls deploy --debug'
               // Remove the credential
               sh 'rm .env'
          }

          echo 'deployment complete'
        }
      }
    }
  }
```

After the above configuration, the code that you commit to the `dev` branch every time will be automatically deployed.

Deploying Hexo Blog

Last updated：2024-12-02 11:16:20

Overview

This document describes how to use the Serverless Website component to quickly construct a serverless Hexo website.

Prerequisites

Node.js has been installed (**starting from September 1, 2020, Serverless components no longer support Node.js versions below 10.0. Please upgrade if needed**).
You have installed Git.
If the applications above are not installed, please install them as instructed in the Hexo documentation.

Directions

1. Install

Install Serverless Framework through npm:

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

Install Hexo through npm:

```
$ npm install -g hexo-cli
```

After installing Hexo, please run the following command to make Hexo create required files in the specified folder:

```
$ hexo init hexo   # Generate a Hexo directory
$ cd hexo
$ npm install
```

After the specified folder is created, its structure is as follows:

```
.
├── _config.yml
├── package.json
├── scaffolds
├── source
```

```
|   ├── _drafts
|   └── _posts
└── themes
```

After the installation is completed, run the `hexo g` command to generate a static webpage:

```
$ hexo g   # generate
```

Note:

If you want to view the result locally, you can run the following command to access `localhost:4000` and view the webpage in a browser:

```
$ hexo s   # server
```

2. Configure

Create a `serverless.yml` file in the `hexo` directory:

```
$ touch serverless.yml
```

Configure the `serverless.yml` file as follows:

```
# serverless.yml

component: website # Name of the imported component, which is required. The `tencen
name: hexodemo # Name of the instance created by this `website` component, which is

app: websiteApp # Website application name, which is optional
stage: dev # Information for identifying environment, which is optional. The defaul

inputs:
  src:
    src: ./public # Upload static files generated by HEXO
    index: index.html
    # dist: ./dist
    # hook: npm run build
    # websitePath: ./
  region: ap-guangzhou
  bucketName: my-bucket
  protocol: https
```

After the configuration is completed, the directory structure is as follows:

```
.
├── .serverless
└── hexo
```

```
    ├── public
    ├── ...
    ├── serverless.yml
    ├── ...
    └── source
```

3. Deploy

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

```
$ sls deploy

serverless ⚡ framework
Action: "deploy" - Stage: "dev" - App: "websiteApp" - Instance: "hexodemo"

region:  ap-guangzhou
website: https://my-bucket-1258834142.cos-website.ap-guangzhou.myqcloud.com


25s › hexodemo › Success
```

You can view your serverless Hexo website by accessing the website URL output by the command line.

Note:

If you want to update an article in your Hexo website, you need to run `hexo g` locally again to generate a static webpage and run `serverless` to update the webpage.

4. Remove

You can run the following command to remove the Hexo website:

```
$ sls remove --debug

  DEBUG — Flushing template state and removing all components.
  DEBUG — Starting Website Removal.
  DEBUG — Removing Website bucket.
  DEBUG — Removing files from the "my-bucket-1250000000" bucket.
  DEBUG — Removing "my-bucket-1250000000" bucket from the "ap-guangzhou" region.
  DEBUG — "my-bucket-1250000000" bucket was successfully removed from the "ap-guang
  DEBUG — Finished Website Removal.

  6s » myWebsite » 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.

Deploying Full-Stack Website with Vue + Express + PostgreSQL

Last updated：2024-12-02 11:16:20

Overview

This document introduces how to use a template to quickly deploy a full-stack serverless application based on Vue, Express, and PostgreSQL. The template mainly contains the following components:
Serverless RESTful API: You can use the Express framework constructed by **SCF** and **API Gateway** to implement RESTful APIs.
Serverless static website: The frontend hosts the Vue.js static pages in **COS**.
PostgreSQL Serverless: You can create a **PostgreSQL database** to provide database services for the full-stack website.
VPC: You can create a **VPC** and **subnet** to enable the interconnection between the SCF and database networks.

Prerequisites

Node.js has been installed (**starting from September 1, 2020, SLS no longer supports Node.js versions below 10.0. Please upgrade if needed**).
The **QcloudPostgreSQLFullAccess** policy has been configured for your account. For the configuration method, see Account and Permission Configuration.

Directions

Installation

Use npm to install Serverless Framework globally:

```
npm install -g serverless
```

If you have already installed Serverless Framework, you can run the following command to upgrade it to the latest version:

```
npm update -g serverless
```

After the installation is completed, run the `serverless -v` command to view the version information of Serverless Framework. Versions earlier than those listed below are not allowed.

```
$ serverless -v
Framework Core: 1.74.1 (standalone)
Plugin: 3.6.14
SDK: 2.3.1
Components: 2.31.6
```

Configuration

1. Create a local folder and run the `serverless init` command to download the related template.

```
serverless init fullstack
```

2. In the root directory of the project, create an .env file and configure the corresponding Tencent Cloud SecretId, SecretKey, region, and availability zone (AZ) information in the file.

```
# .env
TENCENT_SECRET_ID=xxx  // `SecretId` of your account
TENCENT_SECRET_KEY=xxx // `SecretKey` of your account

# Region and AZ configuration
REGION=ap-guangzhou // Resource deployment region, that is, the region for deployin
ZONE=ap-guangzhou-2 // Resource deployment AZ, that is, the AZ for deploying DBs in
```

Note:

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

If you already have a Tencent Cloud account, ensure that your account is authorized with the "AdministratorAccess" permission. You can get `SecretId` and `SecretKey` in API Key Management.

`ZONE` currently can only be `ap-beijing-3` , `ap-guangzhou-2` , or `ap-shanghai-2` .

3. Run the following command to install required dependencies:

```
npm run bootstrap
```

Deployment

1. Run the `sls deploy --all` command to start deployment. The returned information is as follows:

```
$ sls deploy --all

serverless ⚡ framework

serverlessVpc:
  region:     ap-guangzhou
```

```
  zone:       ap-guangzhou-2
  vpcId:      vpc-xxx
  vpcName:    serverless
  subnetId:   subnet-xxx
  subnetName: serverless

fullstackDB:
  region:         ap-guangzhou
  zone:           ap-guangzhou-2
  vpcConfig:
    subnetId: subnet-100000
    vpcId:    vpc-1000000
  dBInstanceName: fullstackDB
  dBInstanceId:   postgres-100000
  private:
    connectionString: postgresql://tencentdb_100000xxxxxxxxxxxxx@172.16.250.15:5432
    host:             172.16.250.15
    port:             5432
    user:             tencentdb_100000
    password:         xxxxxxxx
    dbname:           tencentdb_100000

fullstack-api:
  region: ap-guangzhou
  apigw:
    serviceId:   service-100000
    subDomain:   service-100000-123456789.gz.apigw.tencentcs.com
    environment: release
    url:         https://service-100000-123456789.gz.apigw.tencentcs.com/release/
  scf:
    functionName: fullstack-api
    runtime:      Nodejs10.15
    namespace:    default

fullstack-frontend:
  website: https://fullstack-serverless-db-123456789.cos-website.ap-guangzhou.myqcl

50s › tencent-fullstack › Success
```

After successful deployment, you can use a browser to access the website link generated by the project and see the generated website.

Note:

In this project, SCF is in a VPC and therefore cannot directly access a public network. To enable it to access a public network, configure the SCF network.

2. You can run `sls remove --all` to remove the project. The returned information is as follows:

```
$  sls remove --all

serverless ⚡ framework

38s › tencent-fullstack › Success
```

Deploying ASR

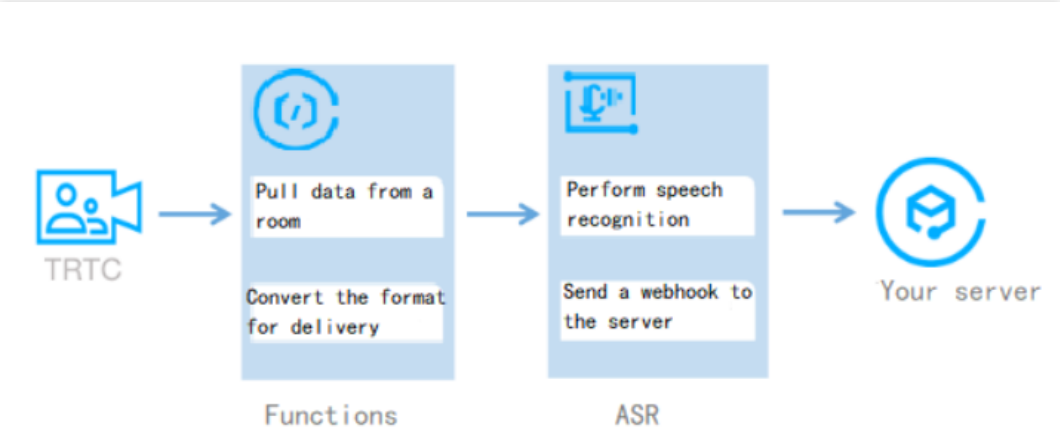
Last updated：2024-12-02 11:16:20

Use cases

Draw and guess: The audio of a user in the room is pulled through this API for real-time recognition and then converted into text, which is called back to the customer's business server for business logic judgment.
Audio audit: This API is closely related to business. It delivers the data streams to the speech recognition API for speech recognition and keyword-based filtering.
Real-time subtitles: The audio data in a room is recognized through this API in real time and converted into text, which is displayed at the frontend.

Architecture

The following figure shows the detailed process:



Application strengths

Real-time return: The audio data in a Tencent Real-Time Communication (TRTC) room is recognized and returned in real time, which is fast and efficient.
Simple process: TRTC and Automatic Speech Recognition (ASR) are deeply integrated, so that the data streams are fully pipelined without complex operations.

Flexible use: The data can be associated with the business logic in real time after it is returned to the business server.

Must-knows

In general, it takes a long time to process speech recognition because Async Execution is enabled during function deployment.
The recognition results are sent to the business server. WebSocket connections are not supported. Therefore, the recognition results cannot be sent to clients.
The default authentication type is **App Authentication**. For more information, see Application Management. You can change the authentication type to **Authentication-Free** during a test. For more information, see [Step 3. Create an API with Mock as the Backend Type](https://www.tencentcloud.com/document/product/628/44318!6af3e7433c19e0c5b321cde7fed199bd).

Directions

1. Activate the service

You must activate Tencent Cloud ASR. For more information, see [Activate the service](https://intl.cloud.tencent.com/document/product/1118/43344#3.-.E6.96.B0.E6.89.8B.E5.85.A5.E9.97.A8!673aaf6cd84c2d22e15eb0eeef6866e8).

2. Deploying function

1. Log in to the SLS console.
2. Click **Create application** to go to the "Create application" page.
3. Select **Live Stream Real-time ASR** and set parameters on the **Basic Configuration** page.
Application name: Specify a custom name for the application.
Region: Select the region based on the actual business.
Key information: You can view the key's information of the Tencent Cloud account on the Manage API Key page.
4. Click **Complete**.
5. Click the function name in the **Function name** column in the **Cloud function** section to go to the details page of the function, and click "Trigger management" on the left to view the access path.

3. Configuring the ASR startup API

```
proto: HTTPS
Method: POST
URL: https://service-xxx-xxxx.sh.apigw.tencentcs.com/release/asr_speech
```

Request parameters:

Parameter	Type	Required	Description
SdkAppId	Int	Yes	Application ID. Each TRTC application has a unique application ID.
RoomId	Int	No	Room ID of the integer type. Each room of a TRTC application has a unique room ID.
StrRoomId	String	No	Room ID of the string type. Either `RoomId` or `StrRoomId` must be configured. If both are configured, `RoomId` is used.
UserId	String	Yes	ID of the user who uses the recording service. Each user of a TRTC application has a unique user ID.
UserSig	String	Yes	Signature of the user who uses the recording service. The signature is used for login authentication of the user.
Callback	String	No	The address to which a webhook is sent by using the POST method when the recording ends.

Sample request:

```
{
    "SdkAppId": 1400000000,
    "RoomId": 43474,
    "UserId": "user_55952145",
    "UserSig": "eJwtzNEKgkAUBNBxxxxxxx",
    "Callback": "https:xxxxxxxx.com/post/xxx"
}
```

Recognition result webhook API

Webhook parameters:

Parameters	Type	Required	Description
SdkAppId	Int	Yes	Application ID.
RoomId	int	Yes	Room ID of the integer type.
UserId	String	Yes	ID of the recognized user.
StrRoomId	String	Yes	Room ID of the string type.

Result	Array	Yes	Results of audio recognition in the format of [{},{},{},{}].
Status	String	Yes	Recognition status of the current user. Valid values: `normal` and `finished`.

The value of `Result` is a JSON array that contains the following objects:

Parameter	Type	Required	Description
Voice	String	Yes	Text of the current sentence in UTF8.
Index	Integer	Yes	Sequence number of the current sentence in the entire audio stream. The sequence number starts from 0.
StartTime	Integer	Yes	Start time of the current sentence in the entire audio stream.
EndTime	Integer	Yes	End time of the current sentence in the entire audio stream.
Message	String	Yes	Execution result of the recognition task, such as recognition finished, recognition in progress, and recognition failed.

Sample result:

```
{
    "RequestID": "95941e2c85898384a95b81c2a5******",
    "SdkAppId": 1400000000,
    "RoomId": 43474,
    "UserId": "user_55952145",
    "Status": "recognizing/finished",
    "Result": [{
        "Voice": "Real-time voice recognition",
        "Index": 0,
        "StartTime": 0,
        "EndTime": 1024,
        "Message": "success"
    }]
}
```

Deploying Static Website

Last updated：2024-12-02 11:16:20

Overview

Tencent Cloud static website 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 static website. The static website component supports a rich set of configuration extensions such as custom domain name and CDN acceleration and provides the easiest-to-use, low-cost, and elastically scalable cloud-based static website development and hosting capabilities.
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 static website 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.
Convenient collaboration: the status information and deployment logs in the cloud make multi-person collaborative development easier.
CDN acceleration, SSL certificate configuration, and custom domain name: you can configure CDN acceleration, custom domain names, and HTTPS access.

Directions

1. Install

Install the latest version of Serverless Framework through npm:

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

2. Create

Create a directory and enter it:

```
$ mkdir tencent-website && cd tencent-website
```

Use the following command and template link to quickly create a static website hosting application:

```
$ serverless init website-demo
$ cd website-demo
```

After download, the directory structure is as follows:

```
|- src
|    └── index.html
└──  serverless.yml
```

In the `src` directory, you can host both simple HTML files and complete React/Vue applications.

3. Deploy

Run the following command in the directory under the `serverless.yml` file to deploy the static website. After the deployment is completed, you can view the URL address of your static website in the output on the command line. Then, you can click the address to visit the hosted website.

```
$ serverless 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**.
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. Configure

The static website component supports "zero" configuration deployment, that is, it can be deployed directly through the default values in the configuration file. Nonetheless, you can also modify more optional configuration items to further customize your project.
The following describes certain configuration items in `serverless.yml` of the static website component:

```
# serverless.yml

component: website # Name of the imported component, which is required. The `tencen
name: websitedemo # Name of the instance created by this `website` component, which

app: websiteApp # Website application name, which is optional
stage: dev # Information for identifying environment, which is optional. The defaul

inputs:
    src:
      src: ./src  # Directory path of the deployed project
      # dist: ./dist # Output directory after build completion. If a hook is config
      # hook: npm run build # Hook script
      index: index.html
```

```
    websitePath: ./
  region: ap-guangzhou
  bucketName: my-bucket
  protocol: http
  hosts:
    - host: abc.com
      https:
        switch: on
        http2: on
        certInfo:
          certId: 'abc'
```

View the complete configuration and configuration description >>

After you update the configuration fields according to the configuration file, run `serverless deploy` or `serverless` again to update the configuration to the cloud.

5. Debug

After the static website 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.

6. Check status

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

```
$ serverless info
```

7. Remove

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

```
$ serverless 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).

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.

Connecting to Serverless DB
Connecting to PostgreSQL

Last updated：2024-12-02 11:16:20

Operation Scenarios

You can use the Serverless Framework component to create, deploy, and manage a Serverless DB instance easily and connect to and access the database through the SCF SDK. Based on the serverless services in the cloud, you can conveniently develop and rapidly deploy your business with "zero" configuration so as to implement it more easily. Currently, Serverless Framework supports connecting to and deploying two types of databases: **PostgreSQL** and **NoSQL**. This document describes how to use an SCF function to connect to a PostgreSQL database.

Prerequisites

You have installed Serverless Framework on at least the following versions; otherwise, please install it as instructed in Installing Serverless Framework.

```
Framework Core: 1.67.3
Plugin: 3.6.6
SDK: 2.3.0
Components: 2.30.1
```

Make sure that the current account has been associated with the **QcloudPostgreSQLFullAccess** policy. To learn more about the association, see Authorization Management.

Directions

This document uses a function in the Node.js programming language as an example to describes how to use the Serverless Framework component to write and create a function and use it to access a PostgreSQL database. The configuration process is as follows:

1. **Configure identity information**: configure the Tencent Cloud account information locally.
1. **Configure a VPC**: use the Serverless Framework VPC component to create a **VPC** and **subnet** for communications between the function and the database.
2. **Configure Serverless DB**: use the Serverless Framework PostgreSQL component to create a PostgreSQL instance to provide database services for the function project.

3. **Write business code**: use the Serverless DB SDK to call the database. SCF allows you to directly call the Serverless DB SDK to connect to and manage a PostgreSQL database.
4. **Deploy and debug**: use Serverless Framework to deploy the project in the cloud and test it in the SCF Console.
5. **Remove project**: you can use Serverless Framework to remove the project.

Configuring identity information

1. Create a local directory to store code and dependent modules. This document uses the `test-postgreSQL` folder as an example.
2. Create a `.env` file in `test-postgreSQL` and configure your Tencent Cloud `SecretId`, `SecretKey`, region, and AZ information in the following format in the file:

```
# .env
TENCENT_SECRET_ID=xxx  // `SecretId` of your account
TENCENT_SECRET_KEY=xxx // `SecretKey` of your account
# Region and AZ configuration
REGION=ap-guangzhou // Resource deployment region, which refers to the region wher
ZONE=ap-guangzhou-2 // Resource deployment AZ, which refers to the AZ where the da
```

Note:

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

If you already have a Tencent Cloud account, please make sure that your account has been granted the `AdministratorAccess` permission. In addition, you can get `SecretId` and `SecretKey` in API Key Management.

Configuring VPC

1. Create a `VPC` folder in `test-postgreSQL`.
2. Create a `serverless.yml` file in `VPC` and enter the following content to configure the VPC and subnet:

```
org: fullstack
app: fullstack-serverless-db
stage: dev
component: vpc # (required) name of the component. In that case, it's vpc.
name: serverlessVpc # (required) name of your vpc component instance.
inputs:
  region: ${env:REGION}
  zone: ${env:ZONE}
  vpcName: serverless
  subnetName: serverless
```

Configuring Serverless DB

1. Create a `DB` folder in `test-postgreSQL`.

2. Create a `serverless.yml` file in `DB` and enter the following content to create and configure a PostgreSQL database:

```
org: fullstack
app: fullstack-serverless-db
stage: dev
component: postgresql
name: fullstackDB
inputs:
  region: ${env:REGION}
  zone: ${env:ZONE}
  dBInstanceName: ${name}
  vpcConfig:
    vpcId: ${output:${stage}:${app}:serverlessVpc.vpcId}
    subnetId: ${output:${stage}:${app}:serverlessVpc.subnetId}
  extranetAccess: false
```

Writing business code

1. Create an `api` folder in `test-postgreSQL` to store the business logic code and relevant dependencies.
2. Create an `src` folder in the `api` folder, enter it on the command line, and run the following command to install the PostgreSQL dependency package:

```
npm install pg
```

3. Create an `index.js` file in the `src` folder and enter the following sample code, so that you can use the Serverless DB SDK to create a connection pool and call the database through the function:

```
'use strict';
const { Pool } = require('pg');
exports.main_handler = async (event, context, callback) => {
  let pgPool = new Pool({
     connectionString: process.env.PG_CONNECT_STRING,
   });
  await pgPool.query(`CREATE TABLE IF NOT EXISTS users (
     ID serial NOT NULL,
     NAME           TEXT         NOT NULL,
     EMAIL          CHAR(50)     NOT NULL,
     SITE          CHAR(50)     NOT NULL
   );`);
  const client = await pgPool.connect();
  const { rows } = await client.query({
   text: 'select * from users',
 });
  await client.end();
  console.log('pgsql query result:',rows)
```

```
}
```

4. Create a `serverless.yml` file in `api` and enter the following content to configure the `Connection String` of Serverless DB in the environment variables:

```
org: fullstack
app: fullstack-serverless-db
stage: dev
component: scf
name: fullstack-serverless-db
inputs:
  name: ${app}
  src:
    src: ./src
    exclude:
      - .env
  region: ${env:REGION}
  runtime: Nodejs10.15
  handler: index.main_handler
  timeout: 30
  vpcConfig:
    vpcId: ${output:${stage}:${app}:serverlessVpc.vpcId}
    subnetId: ${output:${stage}:${app}:serverlessVpc.subnetId}
  environment:
    variables:
      PG_CONNECT_STRING: ${output:${stage}:${app}:fullstackDB.private.connectionStr
```

Deploying and debugging

1. Run the following command for deployment in the `test-postgreSQL` directory on the command line:

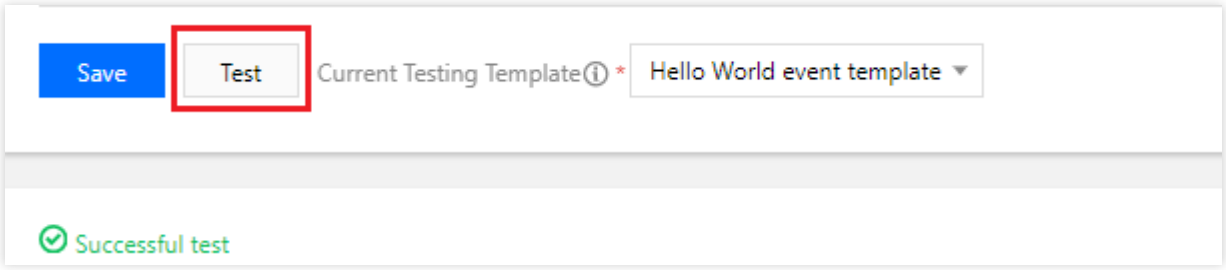
```
sls deploy --all
```

If the following result is returned, the deployment is successful:

```
serverless ⚡ framework
serverlessVpc:
  region:     ap-guangzhou
  zone:       ap-guangzhou-2
  vpcId:      vpc-0ncak84t
  vpcName:    serverless
  subnetId:   subnet-gi085his
  subnetName: serverless
fullstackDB:
  region:         ap-guangzhou
  zone:           ap-guangzhou-2
  vpcConfig:
```

```
  subnetId: subnet-gi085his
  vpcId:    vpc-0ncak84t
   dBInstanceName: fullstackDB
   dBInstanceId:   postgres-0y2x3fd3
   private:
  connectionString: postgresql://tencentdb_0y2x3fd3:GD0U1(q~g7%3D6ySI@10.0.0.10:5432
  host:             10.0.0.10
  port:             5432
  user:             tencentdb_0y2x3fd3
  password:         GD0U1(q~g7=6ySI
  dbname:           tencentdb_0y2x3fd3
fullstack-serverless-db:
   FunctionName: fullstack-serverless-db
   Description:
   Namespace:    default
   Runtime:      Nodejs10.15
   Handler:      index.main_handler
   MemorySize:   128
25s › fullstack-serverless-db › Success
```

2. After the deployment succeeds, you can view and debug the function in the SCF Console. For more information on the test steps, please see Cloud Test. The test success result is as shown below:



Note:

You can also use Serverless Dashboard to monitor the deployed project in real time with ease.

Removing project

Run the following command in the `test-postgreSQL` directory to remove the project:

```
sls remove --all
```

If the following result is returned, the removal is successful:

```
serverless ⚡ framework
38s › tencent-fullstack › Success
```

Connecting to NoSQL

Last updated：2024-12-02 11:16:20

Operation Scenarios

You can use the Serverless Framework component to create, deploy, and manage a Serverless DB instance easily and connect to and access the database through the SCF SDK. Based on the serverless services in the cloud, you can conveniently develop and rapidly deploy your business with "zero" configuration so as to implement it more easily. Currently, Serverless Framework supports connecting to and deploying two types of databases: **PostgreSQL** and **NoSQL**. This document describes how to use an SCF function to connect to a NoSQL database.

Prerequisites

You have installed Serverless Framework on at least the following versions; otherwise, please install it as instructed in Installing Serverless Framework.

```
Framework Core: 1.67.3
Plugin: 3.6.6
SDK: 2.3.0
Components: 2.30.1
```

Notes

Please make sure that the executing role of `SLS_QcsRole` under your account has been granted the `QcloudTCBFullAccess` permission; otherwise, please go to the CAM Console to configure it.
Currently, TCB allows you to create/terminate an environment for up to **4** times per month. Please create an environment with caution, as an error will be reported if the limit is exceeded.

Directions

This document uses a function in the Node.js programming language as an example to describes how to use the Serverless Framework component to write and create a function and use it to create and access a NoSQL database. The configuration process is as follows:
1. **Configure identity information**: configure the Tencent Cloud account information locally.

2. **Create a TCB environment configuration file**: use the Serverless Framework component to create a TCB environment and create and use a NoSQL database in the environment.
3. **Write business code**: use the Serverless DB SDK to call the database. SCF allows you to directly call the Serverless DB SDK to create and manage a NoSQL database.
4. **Deploy and debug**: use Serverless Framework to deploy the project in the cloud and test it in the SCF Console.
5. **Remove project**: you can use Serverless Framework to remove the project.

Configuring identity information

1. Create a local directory to store code and dependent modules. This document uses the `test-NoSQL` folder as an example.
2. Serverless Framework allows you to configure identity information in the following two ways. Please choose one as needed:

Run the following command for authentication:

```
serverless login
```

Create a `.env` file in `test-NoSQL` and configure the corresponding Tencent Cloud `SecretId` and `SecretKey` as follows:

```
# .env
TENCENT_SECRET_ID=xxx  // `SecretId` of your account
TENCENT_SECRET_KEY=xxx // `SecretKey` of your account
```

Note:

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

If you already have a Tencent Cloud account, please make sure that your account has been granted the `AdministratorAccess` permission. In addition, you can get `SecretId` and `SecretKey` in API Key Management.

Creating TCB environment configuration file

1. Create a `DB` folder in `test-NoSQL` .
2. Create a `serverless.yml` file in the `DB` folder and enter the following content to use the Serverless Framework component to configure the TCB environment:

```
# serverless.yml
component: mongodb
name: mongoDBDemoMongo
org: anycodes
app: mongoDBAPP
stage: dev
inputs:
  name: Mydemo
```

Writing business code

1. Create a `function` folder in `test-NoSQL` to store the business logic code and relevant dependencies.
2. Create an `src` folder in the `function` folder, enter it on the command line, and run the following command to install the TCB dependency package:

```
npm install --save tcb-admin-node@latest
```

3. Create an `index.js` file in the `src` folder and enter the following sample code, so that you can use the Serverless TCB SDK to call the TCB environment through the function and call the NoSQL database in the environment:

```
const tcb = require('tcb-admin-node')
const app = tcb.init({
 secretId: process.env.SecretId,
 secretKey: process.env.SecretKey,
 env: process.env.MongoId,
 serviceUrl: 'https://tcb-admin.tencentcloudapi.com/admin'
})
const db = app.database()
const _ = db.command
exports.main = async (event, context) => {
 await db.createCollection('serverless')
 const username = 'serverless'
 const collection = db.collection('serverless')
 if (username) {
         await collection.add({username: username})
             }
     const userList = await collection.get()
     return userList
}
```

4. After writing the business code, create a `serverless.yml` file and enter your **SecretId** and **SecretKey** in the environment variables.

Note:

If you use the following configuration, a TCB environment will be created free of charge. If you already have a free TCB environment, please enter its ID in `MongoId`; otherwise, an error will be reported.

```
component: scf
name: mongoDBDemoSCF
org: anycodes
app: mongoDBAPP
stage: dev
inputs:
```

```
  name: MongoDBDemo
  src: ./src
  runtime: Nodejs8.9
  region: ap-guangzhou
  handler: index.main
  environment:
    variables:
      SecretId: enter your `SecretId`
      SecretKey: enter your `SecretKey`
      MongoId: ${output:${stage}:${app}:mongoDBDemoMongo.EnvId}
```

Deploying and debugging

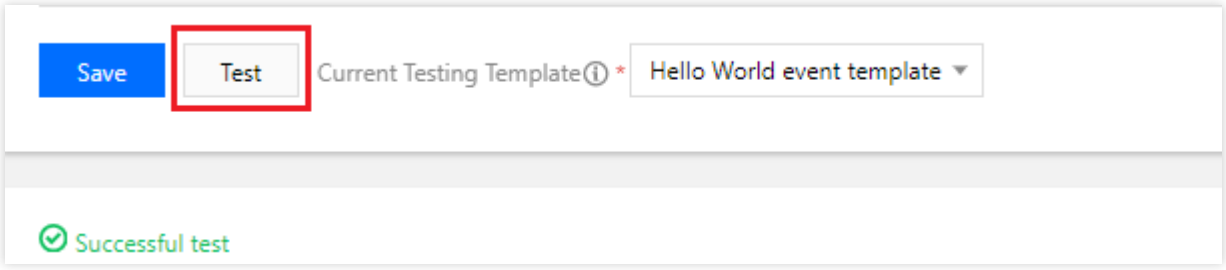
1. Run the following command for deployment in `test-NoSQL` on the command line:

```
sls deploy --all
```

If the following result is returned, the deployment is successful:

```
serverless ⚡ framework
mongoDBDemoMongo:
  Region:    ap-guangzhou
  Name:      Mydemo
  EnvID:     Mydemo-dyxfxv
  FreeQuota: basic
mongoDBDemoSCF:
  FunctionName: MongoDBDemo
  Description:
    Namespace:    default
    Runtime:      Nodejs8.9
    Handler:      index.main
    MemorySize:   128
25s › tcbdemo › Success
```

2. After the deployment succeeds, you can view and debug the function in the SCF Console. For more information on the test steps, please see Cloud Test. The test success result is as shown below:



Note:

You can also use Serverless Dashboard to monitor the deployed project in real time with ease.

Removing project

Run the following command in the `test-NoSQL` directory to remove the project:

```
sls remove --all
```

If the following result is returned, the removal is successful:

```
serverless ⚡ framework
4s › test-NoSQL › Success
```

Deploying OCR-based Text Recognition Application

Last updated：2024-12-02 11:16:20

Based on Tencent Cloud's industry-leading deep learning technology, Tencent Cloud OCR is capable of intelligently recognizing words on images and converting them into editable text. It can recognize printed text and handwriting, meeting the text recognition needs in different scenarios.
By using Serverless Framework Component and OCR SDK, you can quickly deploy a general OCR application based on COS + API + SCF, which contains the following key components:
Serverless RESTful API: you can use the Express framework constructed by SCF and API Gateway to implement RESTful API.
Serverless static website: the frontend hosts React static pages in COS.
COS: you can create a bucket to store target images.

Prerequisites

You have installed Node.js (v8.6 or above; v10.0 or above is recommended).
You have activated OCR.

Directions

Installation

Use npm to install Serverless Framework globally:

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

If you have already installed Serverless Framework, you can run the following command to upgrade it to the latest version:

```
$ npm update -g serverless
```

After the installation is completed, run the `serverless -v` command to view the version information of Serverless Framework and make sure that the version information is as follows or displays higher versions:

```
$ serverless -v
Framework Core: 1.74.1 (standalone)
Plugin: 3.6.14
```

```
SDK: 2.3.1
Components: 2.31.6
```

Configuration

1. Create a local folder and run the `serverless init` command to download the relevant template.

```
$ serverless init ocr-app
```

2. Find the `.env.example` file in the template, rename it to `.env`, and enter your account, key information, and specified bucket (used to store uploaded images) in it.

```
# .env
TENCENT_APP_ID=xxx
TENCENT_SECRET_ID=xxx
TENCENT_SECRET_KEY=xxx

# region of bucket
REGION=ap-guangzhou
```

3. Download all npm dependencies.

```
$ npm run bootstrap
```

Local debugging

1. Run the following command to launch the server:

```
$ cd server && npm run start
```

2. Run the following command to start the frontend:

```
$ cd frontend && npm run start
```

3. Log in to the frontend page at http://localhost:3000 for local debugging.

Deployment

1. Run the following command for deployment:

```
$ sls deploy --all

serverless ⚡ framework

backend:
  region: ap-guangzhou
  apigw:
```

```
    serviceId:   service-4i62q1pg
    subDomain:   service-4i62q1pg-1258834142.gz.apigw.tencentcs.com
    environment: release
    url:         https://service-4i62q1pg-1258834142.gz.apigw.tencentcs.com/release
  scf:
    functionName: serverless-ocr
    runtime:      Nodejs10.15
    namespace:    default

frontend:
  region:  ap-guangzhou
  website: https://serverless-ocr-1258834142.cos-website.ap-guangzhou.myqcloud.com

38s › serverless-ocr › Success
```

After the deployment succeeds, you can use a browser to access the website link generated by the project to view the generated website. Click **Upload Image**, and the project can implement OCR with the OCR SDK.

2. You can run `sls remove --all` to remove the project.

```
$  sls remove --all

serverless ⚡ framework

38s › tencent-fullstack › Success
```

Deploying Streaming Transcoding Application

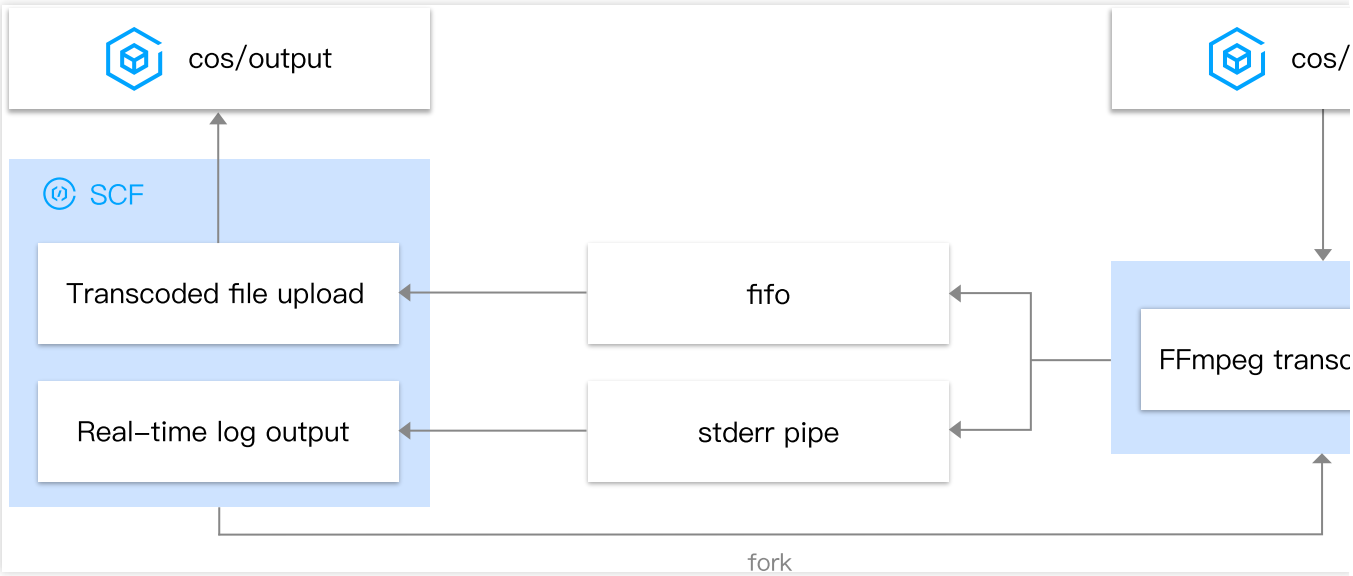
Last updated：2024-12-12 15:51:32

Application Overview

By using COS + SCF + CLS + FFmpeg, you can quickly build a highly available and customizable video transcoding service that features concurrent processing and real-time logging.

Architecture

Create an FFmpeg task process through SCF. The SCF process and FFmpeg task process carry out data transfer through pipe and FIFO. The two task threads in the SCF process respectively receive the FFmpeg log stream output by the FFmpeg task process to the SCF process and the transcoded file stream. The real-time logging thread outputs the log stream, and the upload task is responsible for caching the file stream and uploading it to the user-defined destination COS bucket.



Application advantages

Streaming transcoding

The source video file is pulled and the transcoded file is uploaded in a streaming manner, which breaks through the limitation of local storage and eliminates the need for additional deployment of CFS and other services.

Real-time logging

During video transcoding, you can view the transcoding progress in real time through CLS logs. In addition, this scheme supports outputting the complete logs of the FFmpeg application.

Prolonged execution

By leveraging the prolonged execution mechanism of SCF, this scheme can be run for 12 to 24 hours, making it suitable for transcoding large files that take a long time.

Custom parameters

You can customize the FFmpeg command parameters.

Application resources

After the transcoding application is deployed, the following resources will be created:

SCF function: it reads COS files, outputs files transcoded with FFmpeg to COS, and outputs the real-time log of the transcoding process to CLS in a streaming manner.

CLS log: it stores the real-time log of the transcoding process. CLS logs may incur fees. For more information, please see CLS Billing Rules.

Notes

The transcoding application relies on the prolonged function execution capability. For more information, please see Async Execution.

We recommend you configure the transcoding output bucket and the function in the same region, because cross-region configuration will reduce the stability and efficiency of the transcoding application and incur traffic fees.

You need to create an execution role for the function of the transcoding application and grant it the read/write permissions of COS. For detailed configuration, please see Execution Role.

FFmpeg command parameter configuration varies by transcoding scenario; therefore, you should have basic knowledge of FFmpeg. This document only provides a few samples for your reference. For more FFmpeg commands, please visit FFmpeg official website.

Prerequisites

1. Install Serverlesss Framework.
2. Configure and deploy account permissions as detailed in Account and Permission Configuration.
3. Configure execution role permissions.

Directions

1. Download the transcoding application

```
sls init transcode-app
```

Enter the project directory `transcode-app`, and you will see the directory structure as follows:

```
transcode-app
|- .env  # Environment configuration
|- serverless.yml # Application configuration
|- log/ # Log configuration
|  └── serverless.yml
└──transcode/  # Transcoding function configuration
       |- src/
       |   |- ffmpeg   # FFmpeg transcoding tool
       |   └── index.py
       └── serverless.yml
```

`log/serverless.yml` defines a CLS logset and topic, which are used to store the logs output during the transcoding process. Currently, CLS is used for log storage. Each transcoding application will create related resources based on the configured CLS logset and topic. Using CLS will incur fees. For more information, please see CLS Billing Rules.

`transcode/serverless.yml` defines the basic function configuration and transcoding parameter configuration.

`transcode/src/index.py` implements the transcoding feature.

`transcode/src/ffmpeg` is the Ffmpeg transcoding tool.

2. Configure environment variables and application parameters

Application parameters are in the `transcode-app/serverless.yml` file.

```
# Application information
app: transcodeApp # You need to configure it as your application name
stage: dev # Environment name. Default value: dev
```

Environment variables are in the `transcode-app/.env` file.

```
REGION=ap-shanghai  # Application region
TENCENT_SECRET_ID=xxxxxxxxxxxx # Your Tencent Cloud `sercretId`
TENCENT_SECRET_KEY=xxxxxxxxxxxx # Your Tencent Cloud `sercretKey`
```

Note:

You can log in to the Tencent Cloud console and get `SecretId` and `SecretKey` in API Key Management.

If your account is the root account or a sub-account that has the permission to scan QR codes, you can also directly scan the QR code to deploy the application without configuring `SercretId` and `SercretKey`. For more information, please see Account and Permission Configuration.

3. Configure the parameters needed for transcoding

CLS log definition is in the `transcode-app/log/serverless.yml` file:

```
# Component information. For all configuration items, please visit https://github
component: cls # Name of the imported component
```

```
  name: cls-video # Name of the created instance. Replace it with the name of your

      # Component parameters
  inputs:
          name: cls-log  # You need to configure a name as the name of your CLS lo
          topic: video-log # You need to configure a topic as the name of your CLS
          region: ${env:REGION} # Region, which is uniformly defined in environmen
          period: 7 # Log retention period in days
```

SCF function and transcoding configurations are in the `transcode-app/transcode/serverless.yml` file:

```
# Component information. For all configuration items, please visit https://github.c
component: scf # Name of the imported component
name: transcode-video # Name of the created instance. Replace it with the name of y

# Component parameters
inputs:
  name: transcode-video-${app}-${stage}
  src: ./src
  handler: index.main_handler
  role: transcodeRole # Function execution role which has been granted full access
  runtime: Python3.6
  memorySize: 3072 # Memory size in MB
  timeout: 43200 # Function execution timeout period in seconds. This example indic
  region: ${env:REGION} # Function region, which is uniformly defined in environmen
  asyncRunEnable: true # Enable prolonged execution
  cls: # Function log
    logsetId: ${output:${stage}:${app}:cls-video.logsetId}  # CLS logset. `cls-vide
    topicId: ${output:${stage}:${app}:cls-video.topicId}  # CLS log topic
  environment:
    variables:  # Transcoding parameters
      REGION: ${env:REGION} # Output bucket region
      DST_BUCKET: test-123456789 # Output bucket name
      DST_PATH: video/outputs/ # Output bucket path
      DST_FORMAT: avi # Transcoding format
      FFMPEG_CMD: ffmpeg -i {input} -y -f {dst_format} {output}  # Basic transcodin
      FFMPEG_DEBUG: 1 # Whether to output FFmpeg log. 0: no; 1: yes
      TZ: Asia/Shanghai # CLS log output time zone
  events:
    - cos: # COS trigger
        parameters:
          bucket: test-123456789.cos.ap-shanghai.myqcloud.com  # Input bucket
          filter:
            prefix: video/inputs/  # Bucket path
          events: 'cos:ObjectCreated:*'  # Triggering event
        enable: true
```

Note:

We recommend you configure the output bucket and the function in the same region, because cross-region configuration will reduce the stability and efficiency of the application and incur traffic fees.

The upper limit of memory size is 3,072 MB, and the upper limit of execution time is 43,200s. If you need to adjust them, please submit a ticket for application.

The prolonged execution feature of SCF must be enabled for the transcoding application with `asyncRunEnable: true` .

Please create and authorize the execution role according to Execution Role.

The FFmpeg command configured in the example is only applicable to AVI transcoding. For more information, please see FFmpeg commands.

4. Deploy the project

In the `transcode-app` project directory, run `sls deploy` to deploy the project.

```
cd transcode-app && sls deploy
```

5. Upload the video file

Upload the video file to the specified path of the configured COS bucket, and it will be automatically transcoded, which is `/video/inputs/` in the COS bucket `test-123456789.cos.ap-shanghai.myqcloud.com` in this example.

After successful transcoding, the file will be saved in the output bucket path you configured, which is `/video/outputs/` in the COS bucket `test-123456789.cos.ap-shanghai.myqcloud.com` in this example.

6. Redeploy

If you need to adjust the transcoding configuration, you can modify the `transcode/serverless.yml` file and redeploy the function:

```
cd transcode && sls deploy
```

Monitoring and Logging

Batch uploading files to COS will trigger concurrent transcoding executions.

1. Go to the **Function Service** page in the SCF console and click the function name to enter the function management page.
2. Click **Log Query** to view the log monitoring data.

3. Select **Function Management** > **Function configuration** and click the link of the log topic to redirect to the CLS console.
4. On the **Search and Analysis** page in the CLS console, select the logset and log topic to search for and analyze logs.

FFmpeg Tool

FFmpeg commands

`DST_FORMAT` and `FFMPEG_CMD` in the `transcode-app/transcode/serverless.yml` file specify the transcoding commands of the transcoding application. You can customize the configuration according to your actual use case.
For example, to transcode videos to MP4 format, you can configure `FFMPEG_CMD` as follows:

```
DST_FORMAT: mp4
FFMPEG_CMD: ffmpeg -i {input} -vcodec copy -y -f {dst_format} -movflags frag_keyfra
```

Note:
`FFMPEG_CMD` must include the FFmpeg configuration parameters and formatted part; otherwise, the transcoding task will fail.
FFmpeg command parameter configuration varies by transcoding scenario; therefore, you should have basic knowledge of FFmpeg. The above commands are only for the described use cases. For more FFmpeg commands, please visit FFmpeg official website.

Customizing FFmpeg

The default FFmpeg tool is provided in the sample transcoding application. If you want to customize FFmpeg, you can do the following:
1. Replace the FFmpeg in the sample with your customized FFmpeg.
2. Run `sls deploy` in the `transcode-app/transcode` directory again to deploy the update.

```
cd transcode && sls deploy
```

Note:
If your FFmpeg compilation environment is different from the SCF runtime environment, FFmpeg permission issues may occur. We provide an official image of the SCF runtime environment, which can be used to compile your FFmpeg.

Execution Role

When the transcoding function is running, it needs to read resources in COS for transcoding and write the transcoded resources back to COS; therefore, the function should be configured with an execution role that has read/write permissions of COS. For more information, please see Role and Authorization.

1. Log in to the CAM console and create a role with **Tencent Cloud Product Service** as the role entity.
2. In the **Enter role entity info** step, select **Serverless Cloud Function (scf)** and click **Next**.
3. In the "Configure role policy" step, select the policies required by the function and click **Next**.

Note:

You can directly select `QcloudCOSFullAccess` (full access to COS). If you need more fine-grained permission control, please configure and select according to the actual situation.

4. Enter the role name to complete the role creation and authorization. This role will be used as the execution role of the function and configured in the `transcode-app/transcode/serverless.yml` file.

Note:

As the maximum validity period of an execution role key is 12 hours, the timeout period configured for the function should not be greater than 12 hours. If you need a longer function execution time, you can change the COS access method in `transcode-app/transcode/src/index.py` to configure a permanent key for full access to COS. However, this will expose your key in the code and should be used with caution.