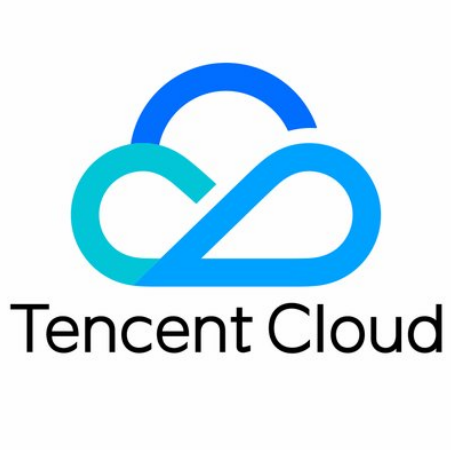
Serverless Framework

Operation Guide

Product Documentation

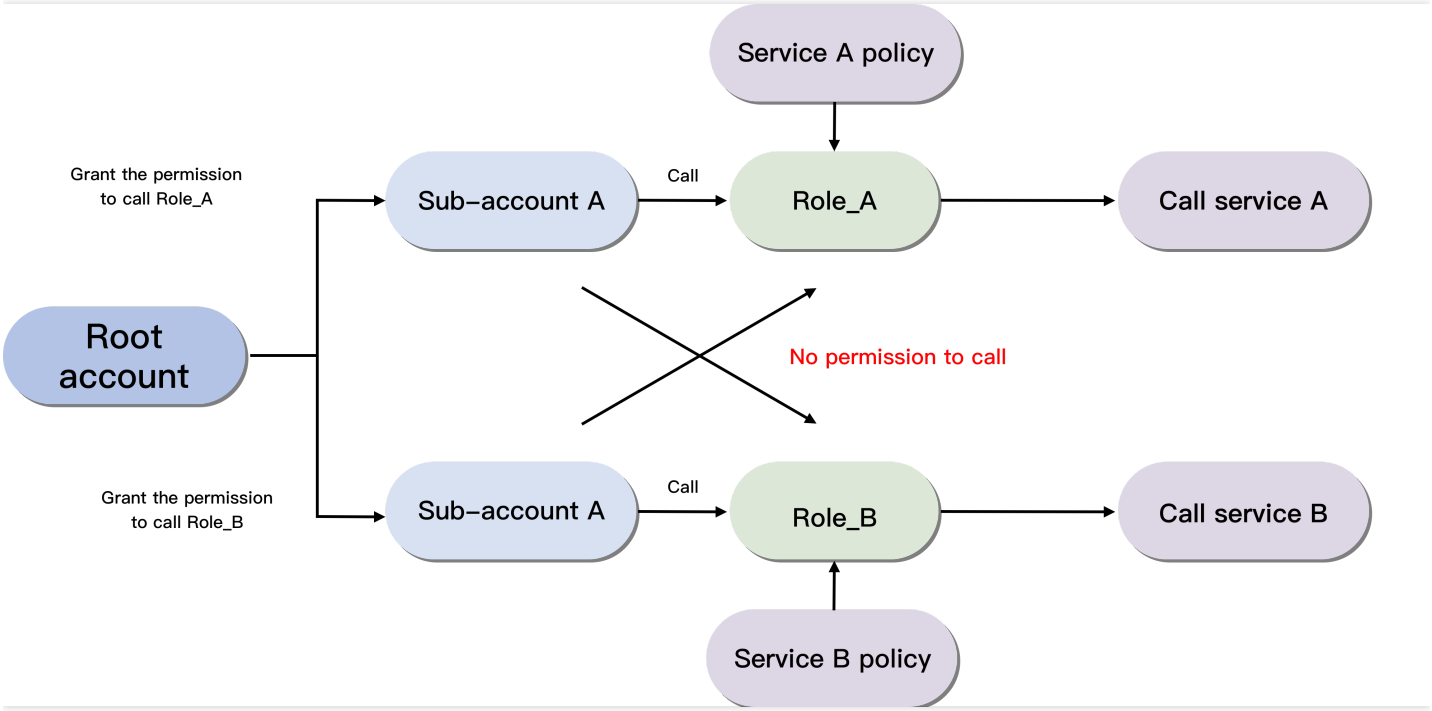


Contents

Operation Guide
Permission Configuration
Configuring Role for Specified Operation

Last updated : 2020-07-14 11:41:58

In addition to the default role `SLS_QcsRole` , a root account can also create multiple custom roles and assign them to sub-users, so that they can have only the policies granted by the corresponding roles as needed, which can implement permission control. Its flowchart is as follows:



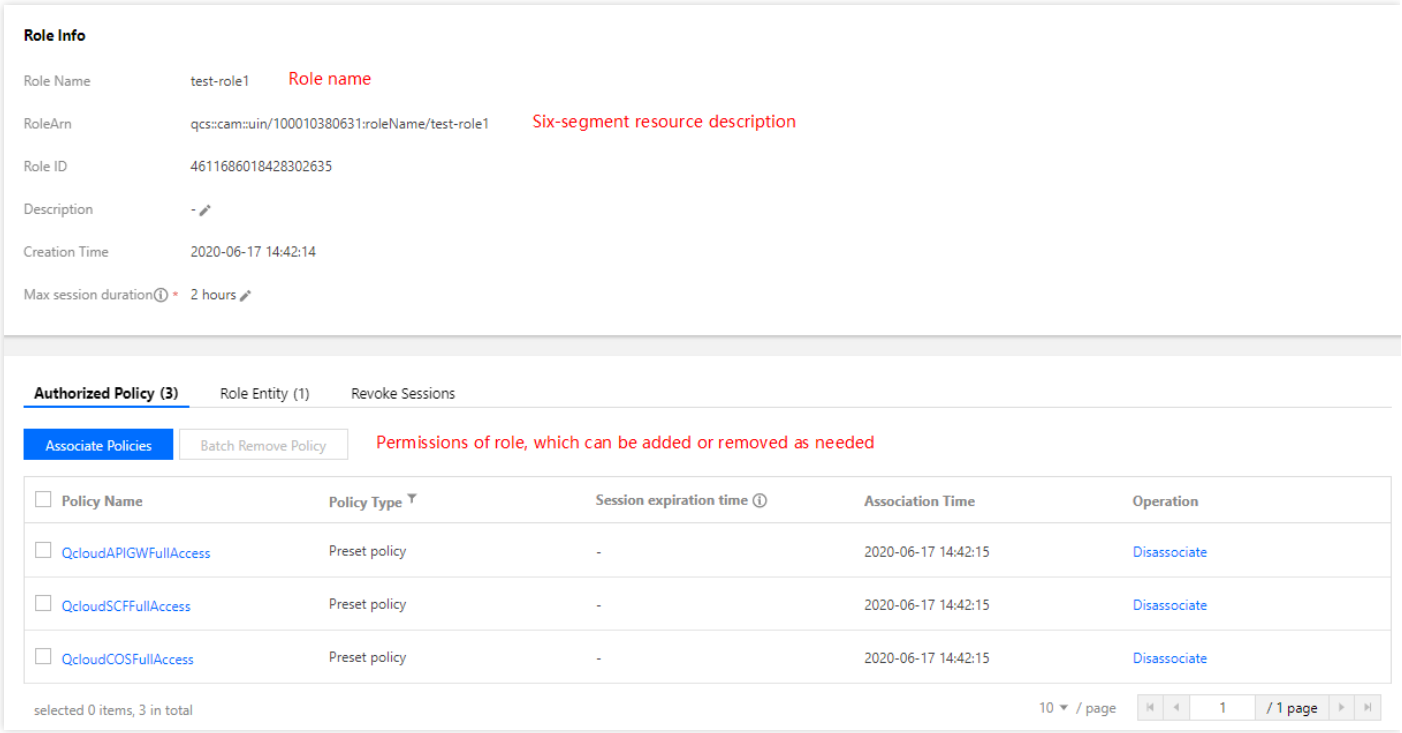
Root Account Configuration Process

You can create a sub-account, configure a role, and grant the role the corresponding policies. The following uses the deployment of an SCF function triggered by API Gateway as an example:

Creating sub-account role

1. Log in to the CAM Console with your root account and click **Role** on the left sidebar.
2. On the role page, click **Create Role** and select **Tencent Cloud Service** as the role entity.
3. Among the services supporting roles, select **Serverless Framework (sls)** and click **Next**.
4. Select presets policies **QcloudCOSFullAccess**, **QcloudAPIGWFullAccess**, and **QcloudSCFFullAccess** and click **Next**.

5. Enter a role name such as `test-role1` and click **Complete**.
   You can click the role name to view the role page after configuration:



Configuring role policy

1. Click **Policy** on the left sidebar to enter the policy management page.
2. On the policy management page, click **Create Custom Policy** and select **Create by Policy Syntax**.
3. Select **Blank Template** as the policy template and click **Next**.
4. Enter the policy name and content and click **Complete**.
   Bind the role policy. Here, you need to populate the `resource` parameter with the six-segment description of the role to be bound to the sub-account:

```
{
"version": "2.0",
"statement": [
{
"action": [
"cam:PassRole"
],
"resource": [
# Six-segment role description (such as `qcs::cam::uin/123456789:roleName/test-role1`)
],
"effect": "allow"
}
]
}
```

> **Note :**
>
> The role resource description can be obtained on the role information page.

Associating sub-user with policy

1. Click **User** > **User List** on the left sidebar to enter the user list page.
2. Select the sub-user to be authorized and click **Authorize** in the "Operation" column.
3. Select the created policy and the preset policy **QcloudSLSFullAccess** in the policy list and click **OK** to associate them with the target sub-account so as to bind the role.
4. **(Optional)** If you think that `QcloudSLSFullAccess` contains excessive permissions, you can create a custom policy to grant a specified resource the SLS call permission with the following policy template:

```
{
"version": "2.0",
"statement": [
{
"action": [
"sls:*"
],
"resource": [
# Enter the project resource name (such as `qcs::sls:ap-guangzhou::appname/*`)
],
"effect": "allow"
}
]
}
```

> **Note :**
>
> The project resource description must strictly follow the CAM specifications. You can also describe the resource more specifically by entering a function name or stage name.

Sub-account Configuration Process

Create a Serverless project locally, add a global configuration item `configRole` in the `serverless.yaml` configuration file, and enter the role name. After the backend successfully checks the permissions, the deployment will be completed.

```
# serverless.yml

component: scf # Name of the imported component, which is required. The `tencent-scf` component is used in this example
name: scfdemo # Name of the instance created by this component, which is required
org: test # Organization information, which is optional. The default value is the `appid` of your Tencent Cloud account
```

```
app: scfApp # SCF application name, which is optional
stage: dev # Information for identifying environment, which is optional. The default value is `dev`

globalOptions:
configRole: test-role1 # Name of specified role, which is optional

inputs:
name: scfFunctionName
src: ./src
runtime: Nodejs10.15 # Runtime environment of function. Valid values: Python2.7, Python3.6, Nodejs6.10, Nodejs8.9, Nodejs10.15, PHP5, PHP7, Go1, Java8.
region: ap-guangzhou
handler: index.main_handler
events:
- apigw:
name: serverless_api
parameters:
protocols:
- http
- https
serviceName:
description: The service of Serverless Framework
environment: release
endpoints:
- path: /index
method: GET
```

⚠ Note :

- If no role is bounded, the sub-account will use `SLS_QcsRole` for SLS deployment by default, and the `configRole` parameter does not need to be set in the configuration file.
- Once a role is bounded, please check the `configRole` name in the `yaml` file carefully. An error will be reported if the value is incorrect or empty. A sub-account can use only bounded roles but cannot use other roles.

Granting Permission to Sub-account

If you want to grant a permission to a sub-account, you need to provide the role name and the name of the policy to be associated together to the root account. Then, the root account can grant the permission in CAM Console > **Role**.

Account and Permission Configuration

Last updated : 2020-10-15 15:02:22

Authorization Method

When deploying a project in Serverless Framework, you need to grant Serverless Framework permissions to manipulate your Tencent Cloud service resources through a role **by scanning the code** or **with a key**.

Authorizing by scanning code

When you run the `sls` command, the system will query whether there is key information in the environment variables. If no key has been configured, a QR code will pop up for you to scan for authorization.

- After you scan the code with the root account, you can deploy the project through quick authorization. For more information on the permissions obtained during quick authorization, please see SLS_QcsRole Role Permission List.
- If you want to deploy by scanning the code with a sub-account, you need to configure policy authorization. For more information on the configuration, please see Configuring sub-account permission.

After you authorize by scanning the code, temporary key information will be generated (which will expire in 15 minutes) and written into the .env file in the current directory.

```
TENCENT_APP_ID=xxxxxx # `AppId` of authorizing account
TENCENT_SECRET_ID=xxxxxx # `SecretId` of authorizing account
TENCENT_SECRET_KEY=xxxxxx # `SecretKey` of authorizing account
TENCENT_TOKEN=xxxxx # Temporary token
```

Authorizing with key

To eliminate the need for repeated authorization due to information expiration in case of authorization by scanning the code, you can authorize with a key. Create an `.env` file in the root directory of the project to be deployed and configure the Tencent Cloud `SecretId` and `SecretKey` information:

```
# .env
TENCENT_SECRET_ID=xxxxxxxxxx # `SecretId` of your account
TENCENT_SECRET_KEY=xxxxxxxx # `SecretKey` of your account
```

You can get `SecretId` and `SecretKey` in API Key Management.

To ensure the account security, we recommend you use a sub-account key for authorization. The sub-account can deploy the project only after being granted the relevant permission. For more information on the configuration, please see Configuring sub-account permission.

Permission Configuration

When deploying a project in Serverless Framework, you need to use a role for authorization as follows:

- The authorizing account should have the permission to manipulate the Serverless Framework service.
- The authorizing account should have the permission to call roles.
- The role to be called should have the policies that can manipulate the corresponding resources.

Configuring root account permission

The root account has the permissions to manipulate the Serverless Framework service and call roles by default. The `SLS_QcsRole` role will be created by default when you activate Serverless Framework, which will have the corresponding policies of the associated services required by Serverless Framework during deployment. For the permissions of `SLS_QcsRole`, please see SLS_QcsRole role permission list.

Configuring sub-account permission

A sub-account does not have the operation permissions by default; therefore, you need to authorize it with the **root account (or a sub-account with the authorization permission)** in the following steps:

1. Grant the permission to manipulate the Serverless Framework service.
2. Grant the permission to call the `SLS_QcsRole` role.

> **Note :**
>
> The `SLS_QcsRole` role has the corresponding policies of the associated services required by Serverless Framework during deployment. You can control the policies as instructed in [Configuring permission to manipulate specified role](#Configuring permission to manipulate specified role).

Granting permission to manipulate Serverless Framework service

When granting the sub-account permission to manipulate the Serverless Framework service, you can select the permission to manipulate all resources or specific resources.

Granting sub-account permission to manipulate all Serverless Framework resources

You can allow a sub-account to manipulate all Serverless Framework resources in the following steps:

1. On the CAM User List page, select the target sub-account and click the username to enter the user details page.
2. Click **Associate Policy**. On the policy adding page, click **Select policies from the policy list**.
3. Search for and associate with `QcloudSLSFullAccess` and click **Next**.
4. Click **OK** to grant the sub-account the permission to manipulate all Serverless Framework resources.
   The policy syntax is as follows:

```
{
"version": "2.0",
```

```
"statement": [
{
"action": [
"sls:*"
],
"resource": "*",
"effect": "allow"
}
]
}
```

Granting sub-account permission to manipulate specific Serverless Framework resources

You can allow a sub-account to manipulate only specific Serverless Framework resources in the following steps:

1. On the CAM User List page, select the target sub-account and click the username to enter the user details page.
2. Click **Associate Policy**. On the policy adding page, click **Select policies from the policy list**.
3. Click **Create Custom Policy**, create a custom policy based on the policy syntax, and associate it with the user. The sample policy syntax is as shown below:

```
{
"version": "2.0",
"statement": [
{
"action": [
"sls:*"
],
"resource": "qcs::sls:ap-guangzhou::appname/${appname}/stagename/${stagename}",
"effect": "allow"
}
]
}
```

After the configuration is completed, the sub-account will have the permission to manipulate serverless applications only under `${appname}` and `${stagename}` .

Granting permission to call `SLS_QcsRole` role

A sub-account needs to be authorized by the root account to call the `SLS_QcsRole` role.

1. On the CAM User List page, select the target sub-account and click the username to enter the user details page.
2. Click **Associate Policy**. On the policy adding page, click **Select policies from the policy list**.
3. Click **Create Custom Policy** > **Create by Policy Syntax** > **Blank Template** and enter the following content. Be sure to replace the role parameter with your own `uin` (account ID):

```
{
"version": "2.0",
```

```
"statement": [
{
"action": [
"cam:PassRole"
],
"resource": [
"qcs::cam::uin/000000000000:roleName/SLS_QcsRole"
],
"effect": "allow"
}
]
}
```

4. Click **OK** to grant the sub-account the permission to manipulate `SLS_QcsRole` .

Configuring permission to manipulate specified role

In addition to the permission to call the `SLS_QcsRole` role, you can also grant the sub-account the permission to call a custom role and control the sub-account permissions with refined permission policies in the custom role. For more information, please see Configuring Role for Specified Operation.

SLS_QcsRole Role Permission List

Policy	Description
QcloudCOSFullAccess	Full access to COS
QcloudSCFFullAccess	Full access to SCF
QcloudSSLFullAccess	Full access to SSL Certificate Service
QcloudTCBFullAccess	Full access to TCB
QcloudAPIGWFullAccess	Full access to API Gateway
QcloudVPCFullAccess	Full access to VPC
QcloudMonitorFullAccess	Full access to Cloud Monitor
QcloudSLSFullAccess	Full access to SLS (Serverless Framework)
QcloudCDNFullAccess	Full access to CDN
QcloudCKafkaFullAccess	Full access to CKafka

⚠ Note :

The full access to SLS (Serverless Framework) is a new permission added to the new version of Serverless Framework. If you use the key on a legacy version for deployment and want to switch to the

new version, you need to delete the key and log in again.

Access Management

Last updated : 2020-07-31 14:21:56

CAM Overview

Cloud Access Management (CAM) is a web-based Tencent Cloud service that helps you securely manage and control access permissions, resources, and use permissions of your Tencent Cloud account. Using CAM, you can create, manage, and terminate users (groups), and control the Tencent Cloud resources that can be used by the specified user through identity and policy management.

Serverless Framework supports **resource-level authorization**. You can use policy syntax to grant sub-accounts permissions to manage individual resources. For more information, please see Authorization Scheme Examples.

Authorizable Resource Types

Serverless Framework supports resource-level authorization. You can grant a specified sub-account the API permission of a specified resource. APIs supporting resource-level authorization include:

API Name	Description	Six-Segment Example of Resource
SaveInstance	Saves the instance information of component	`qcs::sls:${Region}:uin/:appname/${AppName}/stagename/${StageName}`
GetInstance	Gets the instance information of component	`qcs::sls:${Region}:uin/:appname/${AppName}/stagename/${StageName}`
ListInstances	Gets the instance list information of component	`qcs::sls:${Region}:uin/:appname/${AppName}/stagename/${StageName}`
RunComponent	Runs component instance	`qcs::sls:${Region}:uin/:appname/${AppName}/stagename/${StageName}`

API Name	Description	Six-Segment Example of Resource
RunFinishComponent	Finishes running component instance	`qcs::sls:${Region}:uin/:appname/${AppName}/stagename/${StageName}`

Authorization Scheme Examples

Six-Segment resource description

Parameter	Required	Description
qcs	Yes	Tencent Cloud service abbreviation, which indicates a resource of Tencent Cloud.
project_id	Yes	Project information description, which is only used to enable compatibility with legacy CAM logic.
service_type	Yes	Product abbreviation, which is `sls` for Serverless Framework.
region	Yes	Region information, such as `bj` . For more information, please see Region List.
account	No	Root account of resource owner, such as `uin/164256472` . If it is empty, it indicates the root account of the CAM user who creates the policy.
resource	Yes	Detailed resource information of each product, which is `qcs::sls:${Region}:uin/:appname/${AppName}/stagename/${StageName}` for Serverless Framework

Sample

You can log in to the CAM Console as a root account to configure and manage the permissions of Serverless Framework. Currently, Serverless Framework provides two preset policies for **full access permission** and **read-only access permission**:

Full access permission

- Grant a sub-account full access to Serverless Framework (SLS).
- Policy name: QcloudSLSFullAccess

```
{
"version": "2.0",
"statement": [
{
"action": [
"sls:*"
```

```
],
"resource": "*",
"effect": "allow"
}
]
}
```

Read-only access permission

- Grant a sub-account read-only access to Serverless Framework (SLS).
- Policy name: QcloudSLSReadOnlyAccess

```
{
"version": "2.0",
"statement": [
{
"action": [
"sls:Get*",
"sls:List*"
],
"resource": "*",
"effect": "allow"
}
]
}
```

Sub-account Resource Management

- A sub-account can access and manage resources authorized by the root account.
- A sub-account that has the permission of resource creation and finance can purchase resources normally, and the costs will be paid by the root account.

Framework CLI Installation

Last updated : 2019-12-04 17:16:11

Read this on the main serverless docs site

Tencent-SCF - Installation

Installing Node.js

Serverless is a Node.js CLI tool so the first thing you need to do is to install Node.js on your machine.

Go to the official Node.js website, download and follow the installation instructions to install Node.js on your local machine.

Note: Serverless runs on Node v6 or higher.

You can verify that Node.js is installed successfully by running `node --version` in your terminal. You should see the corresponding Node version number printed out.

Installing the Serverless Framework

Next, install the Serverless Framework via npm which was already installed when you installed Node.js.

Open up a terminal and type `npm install -g serverless` to install Serverless.

```
npm install -g serverless
```

Once the installation process is done you can verify that Serverless is installed successfully by running the following command in your terminal:

```
serverless
```

To see which version of serverless you have installed run:

```
serverless --version
```

Setting up Tencent Cloud

To run serverless commands that interface with your Tencent Cloud account, you will need to setup your Tencent Cloud account credentials on your machine.

Follow these instructions on setting up Tencent Cloud credentials

Credentials

Last updated : 2019-12-04 17:16:39

Read this on the main serverless docs site

Tencent Cloud - Credentials

The Serverless Framework needs access to account credentials for your Tencent Cloud account so that it can create and manage resources on your behalf.

Create a Tencent Cloud Account

If you already have a Tencent Cloud account, skip to the next step to create an CAM User and Access Key

To create a Tencent Cloud account:

1. Open https://intl.cloud.tencent.com/, and then choose Sign up.
2. Follow the online instructions.

- Part of the sign-up procedure involves entering a PIN using the phone keypad.

Create an IAM User and Access Key

Services in Tencent Cloud, such as SCF, require that you provide credentials when you access them to ensure that you have permission to access the resources owned by that service. To accomplish this Tencent Cloud recommends that you use Cloud Access Management (CAM).

You need to create credentials Serverless can use to create resources in your Project.

1. Login to your account and go to the Cloud Access Management (CAM) page.
2. Click on **Access Key** and then **Create Key**.
3. View and copy the **APPID**, **SecretId** & **SecretKey** to a temporary place. (To get the SecretKey, you may need to enter a PIN using the phone keypad.)
4. Create a file named `credentials` containing the credentials that you have collected.

```
[default]
tencent_appid = 1251000000
tencent_secret_id = AKIDteBxxxxxxxxxxnZfk
tencent_secret_key = AKID2qsxxxxxxxxxxxxxtTo
```

Save the credentials file to a folder like `~/.tencent/credentials`. Copy the path of the file you saved.

Update `serverless.yml` (optional)

Open up your `serverless.yml` file and update the `provider` section with the path to your credentials file (this path needs to be absolute). It should look something like this:

```
provider:
name: tencent
credentials: ~/.tencent/credentials
```

Create

Last updated : 2019-12-04 17:17:03

Read this on the main serverless docs site

Tencent-SCF - Create

Creates a new service in the current working directory based on the provided template.

Create service in current working directory:

```
serverless create --template tencent-nodejs
```

Create service in new folder:

```
serverless create --template tencent-nodejs --path myService
```

Create service in new folder using a custom template:

```
serverless create --template-url https://github.com/serverless/serverless/tree/master/lib/plugins/create/templates/tencent-nodejs --path myService
```

Options

- `--template` or `-t` The name of one of the available templates. **Required if --template-url and --template-path are not present**.
- `--template-url` or `-u` The name of one of the available templates. **Required if --template and --template-path are not present**.
- `--template-path` The local path of your template. **Required if --template and --template-url are not present**.
- `--path` or `-p` The path where the service should be created.
- `--name` or `-n` the name of the service in `serverless.yml` .

Available Templates

To see a list of available templates run `serverless create --help`

Most commonly used templates:

- tencent-nodejs

- tencent-python
- tencent-php
- tencent-go

Note: The templates will deploy the latest version of runtime by default. When you want to configure specific version of runtime, like `Node.js6`, `Python2.7` or `PHP5`, you have to configure the `runtime` property in `serverless.yml`.

Examples

Creating a new service

```
serverless create --template tencent-nodejs --name my-project
```

This example will generate scaffolding for a service with `Tencent` as a provider and `nodejs8` as runtime. The scaffolding
will be generated in the current working directory.

Your new service will have a default stage called `dev` and a default region inside that stage called `ap-guangzhou`.
The provider which is used for deployment later on is Tencent Cloud.

Creating a named service in a (new) directory

```
serverless create --template tencent-nodejs --path tencent-project
```

This example will generate scaffolding for a service with `Tencent` as a provider and `nodejs8` as runtime. The scaffolding
will be generated in the `tencent-project` directory. This directory will be created if not present. Otherwise Serverless
will use the already present directory.

Additionally Serverless will rename the service according to the path you provide. In this example the service will be
renamed to `tencent-project`.

Packaging

Last updated : 2019-12-04 17:17:23

Read this on the main serverless docs site

Tencent-SCF - Packaging

Package CLI Command

Using the Serverless CLI tool, you can package your project without deploying it to Tencent Cloud. This is best used with CI / CD workflows to ensure consistent deployable artifacts.

Running the following command will build and save all of the deployment artifacts in the service's .serverless directory:

```
serverless package
```

However, you can also use the --package option to add a destination path and Serverless will store your deployment artifacts there (./my-artifacts in the following case):

```
serverless package --package my-artifacts
```

Package Configuration

Sometimes you might like to have more control over your function artifacts and how they are packaged.

You can use the `package` and `exclude` configuration for more control over the packaging process.

Exclude / include

Exclude and include allows you to define globs that will be excluded / included from the resulting artifact. If you wish to
include files you can use a glob pattern prefixed with `!` such as `!re-include-me/**` in `exclude` or the dedicated `include` config.
Serverless will run the glob patterns in order.

At first it will apply the globs defined in `exclude`. After that it'll add all the globs from `include`. This way you can always re-include
previously excluded files and directories.

By default, serverless will exclude the following patterns:

- .git/**
- .gitignore
- .DS_Store
- npm-debug.log
- .serverless/**
- .serverless_plugins/**

and the serverless configuration file being used (i.e. `serverless.yml`)

Examples

Exclude all node_modules but then re-include a specific modules (in this case node-fetch) using `exclude` exclusively

```
package:
exclude:
- node_modules/**
- '!node_modules/node-fetch/**'
```

Exclude all files but `handler.js` using `exclude` and `include`

```
package:
exclude:
- src/**
include:
- src/function/handler.js
```

Note: Don't forget to use the correct glob syntax if you want to exclude directories

```
exclude:
- tmp/**
- .git/**
```

Deploy

Last updated : 2019-12-04 17:17:39

Read this on the main serverless docs site

Tencent-SCF - deploy

The `sls deploy` command deploys your entire service. Run this command when you have made infrastructure changes (i.e., you edited `serverless.yml`). Use `serverless deploy function -f myFunction` when you have made code changes and you want to quickly upload your updated code to Tencent SCF (Serverless Cloud Functions) or just change function configuration.

```
serverless deploy
```

Options

- `--config` or `-c` Name of your configuration file, if other than `serverless.yml|.yaml|.js|.json` .
- `--stage` or `-s` The stage in your service that you want to deploy to.
- `--region` or `-r` The region in that stage that you want to deploy to.
- `--package` or `-p` path to a pre-packaged directory and skip packaging step.
- `--force` Forces a deployment to take place, the triggers will be forced updated too.
- `--function` or `-f` Invoke `deploy function` (see above). Convenience shortcut - cannot be used with `--package` .

Artifacts

After the `serverless deploy` command runs, the framework runs `serverless package` in the background first then deploys the generated package. During the deployment, the framework creates a COS(Cloud Object Storage) bucket to storage the package by default.

Examples

Deployment without stage and region options

```
serverless deploy
```

This is the simplest deployment usage possible. With this command Serverless will deploy your service to the defined
provider in the default stage (`dev`) to the default region (`ap-guangzhou`).

Deployment with stage and region options

```
serverless deploy --stage pro --region ap-guangzhou
```

With this example we've defined that we want our service to be deployed to the `pro` stage in the region `ap-guangzhou` .

Deployment from a pre-packaged directory

```
serverless deploy --package /path/package/directory
```

With this example, the packaging step will be skipped and the framework will start deploying the package from the `/path/package/directory` directory.

Deploy-Function

Last updated : 2019-12-04 17:17:52

Read this on the main serverless docs site

Tencent-SCF - Deploy Function

The `sls deploy function` command deploys an individual SCF function when there is any change in the function. This is a much faster way of deploying changes in code.

```
serverless deploy function -f functionName
```

Options

- `--function` or `-f` The name of the function which should be deployed
- `--stage` or `-s` The stage in your service that you want to deploy to.
- `--region` or `-r` The region in that stage that you want to deploy to.

Examples

Deployment without stage and region options

```
serverless deploy function --function helloWorld
```

Deployment with stage and region options

```
serverless deploy function --function helloWorld --stage dev --region ap-guangzhou
```

Invoke

Last updated : 2019-12-04 17:18:03

Read this on the main serverless docs site

Tencent-SCF - Invoke

Invokes deployed function. It allows to send event data to the function, read logs and display other important information of the function invocation.

```
serverless invoke --function functionName
```

Options

- `--function` or `-f` The name of the function in your service that you want to invoke. **Required**.
- `--stage` or `-s` The stage in your service you want to invoke your function in.
- `--region` or `-r` The region in your stage that you want to invoke your function in.
- `--data` or `-d` String data to be passed as an event to your function.
- `--path` or `-p` The path to a json file with input data to be passed to the invoked function. This path is relative to the root directory of the service.

Examples

```
serverless invoke --function functionName --stage dev --region ap-guangzhou
```

This example will invoke your deployed function named `functionName` in region `ap-guangzhou` in stage `dev`. This will
output the result of the invocation in your terminal.

Function invocation with data

```
serverless invoke --function functionName --stage dev --region ap-guangzhou --data "hello world"
```

Function invocation with data passing

```
serverless invoke --function functionName --stage dev --region ap-guangzhou --path lib/event.json
```

This example will pass the json data in the `lib/event.json` file (relative to the root of the service) while invoking
the specified/deployed function.

Example of `event.json`

```
{
"resource": "/",
"path": "/",
"httpMethod": "GET"
}
```

Logs

Last updated : 2019-12-04 17:18:14

Read this on the main serverless docs site

Tencent-SCF - Logs

Lets you watch the logs of a specific function deployed in Tencent Cloud.

```
serverless logs -f hello

# Optionally tail the logs with -t
serverless logs -f hello -t
```

Options

- `--function` or `-f` The function you want to fetch the logs for. **Required**
- `--stage` or `-s` The stage you want to view the function logs for. If not provided, the plugin will use the default stage listed in `serverless.yml` . If that doesn't exist either it'll just fetch the logs from the `dev` stage.
- `--region` or `-r` The region you want to view the function logs for. If not provided, the plugin will use the default region listed in `serverless.yml` . If that doesn't exist either it'll just fetch the logs from the `ap-guangzhou` region.
- `--startTime` A specific unit in time to start fetching logs from (ie: `2019-7-12 00:00:00`).
- `--tail` or `-t` You can optionally tail the logs and keep listening for new logs in your terminal session by passing this option.
- `--interval` or `-i` If you choose to tail the output, you can control the interval at which the framework polls the logs with this option. The default is `1000` ms.

Examples

Note: There's a small lag between invoking the function and actually having the log event registered in Tencent Cloud Log Service. So it takes a few seconds for the logs to show up right after invoking the function.

```
serverless logs -f hello
```

This will fetch the logs from last 10 minutes as startTime was not given.

```
serverless logs -f hello -t
```

Serverless will tail the function log output and print new log messages coming in starting from 10 seconds ago.

Rollback

Last updated : 2019-12-04 17:18:27

Read this on the main serverless docs site

Tencent-SCF - Rollback

Rollback a service to a specific deployment.

```
serverless rollback --timestamp timestamp
```

If `timestamp` is not specified, Framework will show your existing deployments.

Options

- `--timestamp` or `-t` The deployment you want to rollback to.
- `--verbose` or `-v` Shows any history deployment.

Examples

At first you want to run `serverless deploy list` to show your existing deployments. This will provide you with a list of the deployments stored in your COS bucket. You can then use the timestamp of one of these deployments to rollback to this specific deployment.

Example:

```
$ sls deploy list
Serverless: Listing deployments:
Serverless: -------------
Serverless: Timestamp: 1572625699
Serverless: Datetime: 2019-11-01T16:28:19.896Z
Serverless: Files:
Serverless: - my-service-dev-1572625699.json
Serverless: - my-service-dev-SNixdp-2019-11-01-16-28-19.zip
Serverless: -------------
Serverless: Timestamp: 1572350506
Serverless: Datetime: 2019-10-29T12:01:46.816Z
Serverless: Files:
Serverless: - my-service-dev-1572350506.json
Serverless: - my-service-dev-2SDp7w-2019-10-29-12-01-46.zip

$ sls rollback -t 1572625699
```

```
Serverless: Rollback function my-service-dev-function_one
Serverless: Rollback function my-service-dev-function_one
Serverless: Rollback configure for function my-service-dev-function_one
Serverless: Setting tags for function my-service-dev-function_one
Serverless: Rollback trigger for function my-service-dev-function_one
Serverless: Deployed function my-service-dev-function_one successful
```

Remove

Last updated : 2019-12-04 17:18:38

Read this on the main serverless docs site

Tencent-SCF - Remove

The `sls remove` command will remove the deployed service, defined in your current working directory, from the provider.

```
serverless remove
```

Options

- `--stage` or `-s` The name of the stage in service, `dev` by default.
- `--region` or `-r` The name of the region in stage, `ap-guangzhou` by default.

Examples

Removal of service in specific stage and region

```
serverless remove --stage dev --region ap-guangzhou
```

This example will remove the deployed service of your current working directory with the stage `dev` and the region `ap-guangzhou` .

Info

Last updated : 2019-12-04 17:18:49

Read this on the main serverless docs site

Tencent-SCF - Info

Displays information about the deployed service, such as runtime, region, stage and function list.

```
serverless info
```

Options

- `--stage` or `-s` The stage in your service you want to display information about.
- `--region` or `-r` The region in your stage that you want to display information about.

Provided lifecycle events

- `info:info`

Examples

```
$ sls info
Serverless:

Service Information
service: my-service
stage: dev
region: ap-guangzhou

Deployed functions:
my-service-dev-function_one

Undeployed function:
my-service-dev-function_two
```

Deploy-List

Last updated : 2019-12-04 17:19:01

Read this on the main serverless docs site

Tencent-SCF - Deploy List

The `sls deploy list [functions]` command will list information about your deployments.

You can either see all available deployments in your COS deployment bucket by running `serverless deploy list` or you can see the deployed functions by running `serverless deploy list functions`.

The displayed information is useful when rolling back a deployment or function via `serverless rollback`.

Options

- `--stage` or `-s` The stage in your service that you want to deploy to.
- `--region` or `-r` The region in that stage that you want to deploy to.

Examples

List existing deploys

```
serverless deploy list
```

List deployed functions and their versions

```
serverless deploy list functions
```

Metrics

Last updated : 2019-12-04 17:19:16

Read this on the main serverless docs site

Tencent-SCF - Metrics

Lets you watch the metrics of a specific function.

```
serverless metrics
```

Options

- `--function` or `-f` The function you want to fetch the metrics for.
- `--stage` or `-s` The stage you want to view the function metrics for. If not provided, the plugin will use the default stage listed in `serverless.yml`. If that doesn't exist either it'll just fetch the metrics from the `dev` stage.
- `--region` or `-r` The region you want to view the function metrics for. If not provided, the plugin will use the default region listed in `serverless.yml`. If that doesn't exist either it'll just fetch the metrics from the `ap-guangzhou` region.
- `--startTime` A specific unit in time to start fetching metrics from (ie: `"2019-7-12 00:10:00"`).
- `--endTime` A specific unit in time to end fetching metrics from (ie: `"2019-7-12 00:10:00"`).

Examples

Note: There's a small lag between invoking the function and actually having access to the metrics. It takes a few seconds for the metrics to show up right after invoking the function.

See service wide metrics for the last 24h

```
serverless metrics
```

Displays service wide metrics for the last 24h.

See service wide metrics for a specific timespan

```
serverless metrics --startTime "2019-11-01 00:00:00" --endTime "2019-11-02 00:00:00"
```

Displays service wide metrics for the time between November 1, 2019 and November 2, 2019.

See all metrics for the function `hello` of the last 24h

```
serverless metrics --function hello
```

Displays all `hello` function metrics for the last 24h.

See metrics for the function `hello` of a specific timespan

```
serverless metrics --function hello --startTime "2019-11-01 00:00:00" --endTime "2019-11-02 00:00:00"
```

Displays all `hello` function metrics for the time between November 1, 2019 and November 2, 2019.

Configuration Parameter

Last updated : 2020-11-09 17:54:39

Currently, full configurations of Serverless Framework Component v2 applications are as follows:

- tencent-scf Full Configuration
- tencent-express Full Configuration
- tencent-website Full Configuration

Developing and Debugging

Last updated : 2020-11-09 17:53:16

Development Mode

The development mode enables you to write code for and develop and debug projects in development status more easily, so that you can continuously focus on the process from development to debugging while minimizing the interruptions caused by other tasks such as packaging and update.

Entering development mode

Under a project, you can run `serverless dev` to enter the development mode:

Below is an example:

```
$ sls dev
serverless ⚡ framework
Dev Mode - Watching your Component for changes and enabling streaming logs, if supported...

Debugging listening on ws://127.0.0.1:9222.
For help see https://nodejs.org/en/docs/inspector.
Please open chorme, and visit chrome://inspect, click [Open dedicated DevTools for Node] to debug your cod
e.
---------------------- The realtime log --------------------
17:13:38 - express-api-demo - deployment
region: ap-guangzhou
apigw:
serviceId: service-b77xtibo
subDomain: service-b77xtibo-1253970226.gz.apigw.tencentcs.com
environment: release
url: http://service-b77xtibo-1253970226.gz.apigw.tencentcs.com/release/
scf:
functionName: express_component_6r6xkh60k
runtime: Nodejs10.15
namespace: default

express-api-demo › Watching
```

After you enter the development mode, the Serverless tool will output the deployed content and start continuous file monitoring. When a code file is modified, it will be automatically deployed again to sync the local file to the cloud.

Deploy again and output the deployment information:

```
express-api-demo › Deploying ...
Debugging listening on ws://127.0.0.1:9222.
For help see https://nodejs.org/en/docs/inspector.
Please open chorme, and visit chrome://inspect, click [Open dedicated DevTools for Node] to debug your cod
```

```
e.
---------------------- The realtime log ----------------------
21:11:31 - express-api-demo - deployment
region: ap-guangzhou
apigw:
serviceId: service-b7dlqkyy
subDomain: service-b7dlqkyy-1253970226.gz.apigw.tencentcs.com
environment: release
url: http://service-b7dlqkyy-1253970226.gz.apigw.tencentcs.com/release/
scf:
functionName: express_component_uo5v2vp
runtime: Nodejs10.15
namespace: default
```

⚠ Note :

Currently, `serverless dev` supports only by the Node.js 10 runtime environment. It will support real-time logging in more environments such as Python and PHP.

Exiting development mode

You can press Ctrl+C to exit the development mode (`dev` mode).

```
express-api-demo › Disabling Dev Mode & Closing ...

express-api-demo › Dev Mode Closed
```

In-cloud Debugging: Node.js 10+

For projects whose runtime environment is Node.js 10+, you can connect them to in-cloud debugging by enabling in-cloud debugging and using a debugging tool such as Chrome DevTools or VS Code Debugger.

Enabling in-cloud debugging

When you enter the development mode as instructed above, if the project is a function whose runtime environment is Node.js 10 or above, in-cloud debugging will be automatically enabled and debugging information will be output.

For example, when you enable the development mode, if the following information is output, in-cloud debugging has been enabled for this function.

```
Debugging listening on ws://127.0.0.1:9222.
For help see https://nodejs.org/en/docs/inspector.
Please open chorme, and visit chrome://inspect, click [Open dedicated DevTools for Node] to debug your cod
e.
```

Connecting debugger: Chrome DevTools

The following describes how to use DevTools in Chrome to connect to a remote environment for debugging:

1. Start the Chrome browser.
2. Enter `chrome://inspect/` in the address bar.
3. Open DevTools by clicking `Open dedicated DevTools for Node` in the `Devices` column or clicking `inspect` under the specific target in `Remote Target #LOCALHOST`. If you cannot open the target or there are no targets, please check whether configuration of `localhost:9229` or `localhost:9222` exists in `Configure` under `Devices`, which corresponds to the output after in-cloud debugging is enabled.
4. After opening DevTools by clicking `Open dedicated DevTools for Node`, you can switch to the "Sources" tab to view the remote code. The actual code of the function is in the `/var/user/` directory.
5. On the **Sources** tab, the displayed code may be loaded. More remote files will be displayed as the debugging proceeds.
6. Double-click a file to open it and set a breakpoint at the specified position in it.
7. If you trigger the function in any means such as URL access, page, command, or API, the remote environment will start running and be interrupted at the breakpoint to wait for further operations.
8. On the tool bar on the right of DevTools, you can continue the execution of an interrupted program or perform other operations such as step-over, step-into, and step-out on it. You can also directly view the current variables or set the variables that you want to track. For more information on how to use DevTools, please see the DevTools user guide.

Notes on in-cloud debugging

SCF in-cloud debugging is currently in beta test. You are recommended to try it out and share your questions and suggestions with us.
Before using SCF in-cloud debugging, you need to note the following:

1. In-cloud debugging uses an actually running SCF instance for debugging.
2. Because of the randomness of event triggering, if there are multiple instances, an event may be triggered on a random instance. Therefore, not all requests can hit the debugging instance and trigger debugging.
3. When the debugging breakpoint pauses, if it stops running for a long period of time and there is no return, the trigger such as API gateway may prompt timeout.
4. When the debugging breakpoint pauses, if the instance is still in countdown status and continues running until the execution is completed after debugging completion, the total consumed time will be recorded as the function execution duration.
5. The maximum duration of a single execution from triggering of instance execution to debugging completion is 900 seconds. If the debugging is interrupted for over 900 seconds, the execution will be forcibly ended, and a timeout after 900 seconds will be used as the function execution duration for statistics and measurement.
6. The debugging capability on the current version will set the function timeout period to 900 seconds. If you exit debugging properly, the timeout period will be reset to a normal value. If you exceptionally exit or forcibly end debugging, the function timeout period will fail to be set to a normal value. In this case, you can

deploy the function again (on the CLI) or manually edit it (in the console) to modify the timeout configuration.

Exiting in-cloud debugging

When you exit the development mode, in-cloud debugging will be disabled automatically.

List of Supported Commands

Last updated : 2020-09-21 14:53:45

Serverless Framework supports the following CLI commands:

- `serverless registry` : views the list of available components.
- `serverless registry publish` : publishes the component to the Serverless registry.

  `--dev` : publishes the component on the `@dev` version for development or testing.
- `serverless init xxx` : downloads a specified template from the Registry by entering the name of the desired template after `init` , such as "$ serverless init fullstack".

  `sls init xxx --name my-app` : customizes the project directory name.

  `--debug` : lists log information during the template download process.
- `serverless deploy` : deploys the component instance in the cloud.

  `--debug` : lists log information such as the deployment operations and the status output by `console.log()` during component deployment.

  `---inputs.publish` : during deployment, publishes all function versions under the project.

  `---inputs.traffic=0.1` : switches 10% of the traffic to the `$latest` function version during deployment and the rest of the traffic to the last published function version.
- `serverless remove` : removes the component instance from the cloud.

  `--debug` : lists log information such as the removal operations and the status output by `console.log()` during component removal.
- `serverless info` : gets and displays information related to the component instance.

  `--debug` : lists more `state` values.
- `serverless dev` : starts the development mode ("DEV Mode") and automatically deploys changed information when component status changes are detected. In the development mode, information such as execution logs, invocation information, and errors can be outputted on the command line in real time. In addition, it supports in-cloud debugging for Node.js applications.

Serverless Template

Last updated : 2020-11-09 17:41:07

Operation Scenarios

Serverless Framework enables you to build your own Serverless components or project templates and publish them in the Serverless Registry for use by your team and others. This document describes how to develop your own templates through Registry and reuse them.

Prerequisites

You have installed Serverless Framework on at least the following versions:

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

Directions

Initializing template project with Registry

1. View available components or templates

You can view available components or templates in the following two ways:

1. Access the Serverless Registry page.
2. Run the `sls registry` command to list all recommended components or project templates:

```
$ sls registry

serverless ⚡ registry

Featured Components:

apigateway - https://github.com/serverless-components/tencent-apigateway
cdn - https://github.com/serverless-components/tencent-cdn
cos - https://github.com/serverless-components/tencent-cos
django - https://github.com/serverless-components/tencent-django/
...
```

```
Featured Templates:

fullstack - Deploy a full stack application.
fullstack-nosql - Deploy a nosql full stack application.
ocr-app - Deploy a serverless OCR application.

Serverless › Find more here: https://registry.serverless.com
```

2. Initialize a project from the template

Once you have determined the project template to be used, you can use the built-in `init` command to initialize your project. The `init` command will automatically download the template you select from the Registry and create a project folder for you in the following steps:

```
$ sls init fullstack # Create a project by using the `fullstack` project template
$ cd fullstack # Enter the project folder
```

3. Quickly deploy your project

After the initialization is completed, you can develop your project in the local project folder and quickly deploy it in the cloud by running the `sls` command:

```
$ sls deploy --all # Deploy the entire `fullstack` project
```

Developing your own template through Registry

It is very easy to publish the project you deploy based on Serverless Component as a template. You do not need to make any changes to the project; instead, you simply need to add a `serverless.yml` file (or modify the existing `serverless.yml` file) and add some template metadata that you want to tell the Registry as described below:

```
# serverless.yml
name: fullstack # Project template name
displayName: fullstack application # Name of the project template displayed in the console
author: Tencent Cloud, Inc. # Author name
org: Tencent Cloud, Inc. # Organization name, which is optional
type: template # Project type, which can be either `template` or `component`. It is `template` here
description: Deploy a full stack application. # Describe your project template
description-i18n:
zh-cn: Quickly deploy a full-stack application # Description
keywords: tencent, serverless, express, website, fullstack # Keywords
repo: https://github.com/serverless-components/tencent-fullstack # Source code
readme: https://github.com/serverless-components/tencent-fullstack/tree/master/README.md # Detailed descrip
tion file
license: MIT # Copyright notice
src: # Describe the files in the project to be published as a template
src: ./ # Specify a relative directory, the files under which will be published as a template
exclude: # Describe the files in the specified directory to be excluded
# The following files are typically excluded
```

```
# 1. Files containing `secrets`
# 2. Files managed by `.git` git source code
# 3. Third-party dependencies such as `node_modules`
- .env
- '**/node_modules'
- '**/package-lock.json'
```

After the `serverless.yml` file is configured, you can use the `sls registry publish` command to publish the project to the Registry as a template, so that others can reuse it by entering its name.

Deploying Static Website

Last updated : 2020-09-04 15:56:33

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 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 `tencent-website` component is used in this example
name: websitedemo # Name of the instance created by this `website` component, which is required
org: test # Organization information, which is optional. The default value is the `appid` of your Tencent Cloud account
app: websiteApp # Website application name, which is optional
stage: dev # Information for identifying environment, which is optional. The default value is `dev`

inputs:
src:
root: ./
src: ./src
hook: npm run build
index: index.html
websitePath: ./
region: ap-guangzhou
bucketName: my-bucket
protocol: http
hosts:
- host: anycoder.cn
```

```
  https:
  certId: 123
```

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.

Grayscale Release

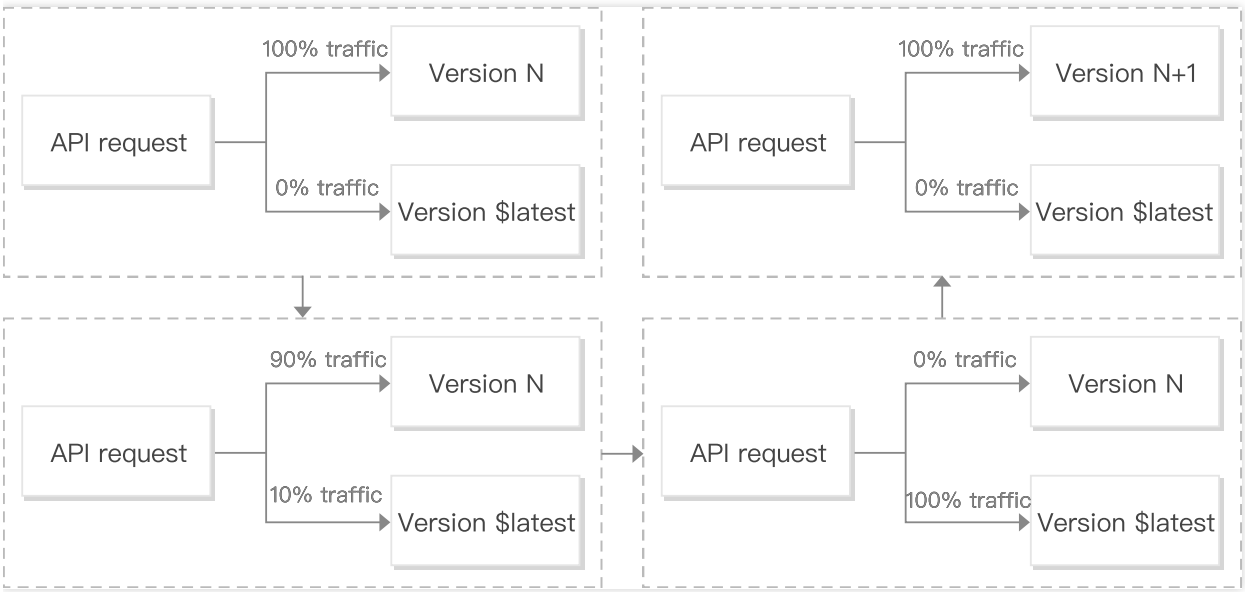
Last updated : 2020-10-16 15:44:27

Overview

Grayscale release (aka canary release) is a release method that can smoothly transition between black and white.

Grayscale release of a serverless application is to configure the traffic rule of the SCF function whose alias is `$default` (default traffic) to configure the traffic of two function versions, where one is the `$latest` version of the function, while the other is the last published version. During the deployment, the `traffic` parameter specifies the traffic percentage on the `$latest` version, while the rest traffic will be switched to the last published version of the current function by default.

Every time a feature is published, you can run `sls deploy` to deploy it onto the `$latest` version. You can switch some traffic to the `$latest` version to check the performance and gradually switch the rest traffic to it. When 100% traffic has been switched to it, it will be fixed, and all traffic will be switched to the fixed version.



Command Description

Function release version

Publish all function versions under the project during deployment:

```
sls deploy --inputs.publish
```

Publish the `fun01` and `fun02` function versions under the project during deployment:

```
sls deploy --inputs.publish="fun01,fun02"
```

Setting function traffic

After deployment, switch 20% traffic to the `$latest` version

```
sls deploy --inputs.traffic=0.2
```

- In Serverless Framework, the traffic rule of the SCF function whose alias is `$default` is modified to switch the traffic.
- The objects to be configured are always the `$latest` version and the last published function version.
- The value of `traffic` is configured as the traffic percentage of the `$latest` version. The traffic percentage of the last published SCF function version is 1 minus the traffic percentage of the `$latest` version (for example, if traffic=0.2, the traffic rule of `$default` will be `{$latest:0.2, last published function version:0.8}`).
- If no fixed versions are published for the function and only the `$latest` version exists, no matter how `traffic` is set, it will always be `$latest:1.0` .

Directions

You can publish a tested feature through grayscale release in the following steps:

1. Configure the production environment information into the .env file (`STAGE=prod` indicates 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 100% traffic to it:
   ```
   sls deploy --inputs.publish --inputs.traffic=0
   ```