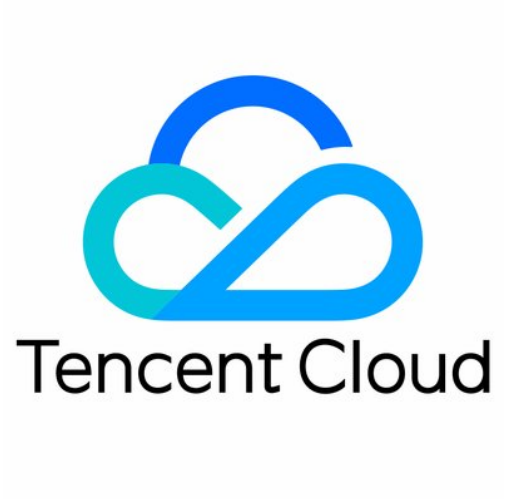
Serverless Cloud Function

Developer Tools

Product Documentation



Contents

Developer Tools
Serverless Framework CLI
Overview

Last updated : 2020-06-24 16:54:46

Overview

Well-received in the industry, Serverless Framework allows you to deploy a complete, available serverless application framework without the need to care about underlying resources. It features resource orchestrating, auto scaling, and event driving and covers the full development lifecycle from coding and debugging to testing and deploying, helping you quickly build serverless applications with the aid of Tencent Cloud resources.

Serverless Framework Components

Serverless Components is a scenario-based solution that supports orchestration and organization of multiple cloud resources for different use cases such as Express framework integration and website deployment. It can greatly simplify the configuration and management of cloud resources while interconnecting Tencent Cloud products such as gateways, COS, and CAM, so that you can focus more on your business development. For more information, please see Serverless Components on GitHub.

Benefits and features

- **Ease of use**
  Serverless Components is built around your scenarios (e.g., websites, blogs, payment systems, and image services). It abstracts underlying infrastructure configuration and enables you to implement your business scenarios with simple configurations.
- **Reusability**
  A serverless component can be easily created and deployed through a very simple `serverless.yml` file. Plus, its JavaScript library `serverless.js` supports extension and reuse with simple syntax.
- **Fast deployment**
  The deployment of most serverless components is about 20 times faster than traditional

configuration tools. They allow rapid deployment and remote verification which help effectively reduce the workload of local emulation and debugging.

SCF component

The SCF component of Tencent Serverless Framework enables you to quickly package and deploy your function projects. Based on serverless services (functions, triggers, etc.) in the cloud, it can implement "zero" configuration, convenient development, and rapid deployment of your first function. It supports a rich set of configuration extensions and provides easy-to-use, low-cost, and elastically scalable cloud-based function development, configuration, and deployment capabilities.

Getting Started

You can quickly get started with Serverless Framework CLI as instructed in Creating Function on CLI.

Permission Management

Last updated : 2020-11-26 18:24:32

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/${enter the account's uin}:roleName/SLS_QcsRole"
],
"effect": "allow"
},
{
"resource": [
"*"
],
"action": [
"name/sts:AssumeRole"
],
"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
QcloudCodingFullAccess	Full access to CODING DevOps
QcloudPostgreSQLFullAccess	Full access to TencentDB for PostgreSQL
QcloudAccessForSLSRole	This policy can be associated with the Serverless Framework (SLS) service role (SLS_QCSRole) for SLS' quick experience feature to access other Tencent Cloud service resources. It contains permissions of CAM-related operations.

⚠ 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.

Function Operations

Last updated : 2020-11-26 18:23:22

Overview

This document describes how to quickly create and deploy an SCF project.

Prerequisites

- Serverless Framework has been installed. For more information, please see Installing Serverless Framework.
- Your account has the Serverless Framework permissions. For more information, please see Account and Permission Configuration.

Directions

Creating function

Run the following command to quickly create a function in the Node.js language:

```
sls init scf-demo
```

Note :

`scf-demo` in the command can be replaced with a template for another programming language. Currently, the SCF component supports the following components: `go1-helloworld`, `nodejs1015-helloworld`, `php72-helloworld`, and `python36-helloworld`.

After the function is successfully created, there will be a `serverless.yml` configuration file in the `scf-demo` project directory, which defines the SCF resource information for each deployment. For more information on the configuration file, please see Configuration file.

Deploying function

Run the following command in the `scf-demo` directory to deploy the function:

```
sls deploy
```

A QR code will pop up. Please scan it to authorize and start deployment. After successful deployment, SCF resources will be automatically created.

Note :

If authentication fails, please authorize as instructed in Account and Permission Configuration.

View function information

Run the following command to view the information of the deployed SCF resources:

```
sls info
```

Removing function

Run the following command to remove the deployed SCF resources:

```
sls remove
```

Configuration file

Serverless Framework CLI relies on the configuration in the `serverless.yml` file when creating or updating a function. When the `sls deploy` command is executed to deploy a function, SCF resources will be created or updated according to the configuration in the `serverless.yml` file. Below is a simple example of this file. For more information on the configuration, please see Full Configuration.

```
# SCF component configuration sample
# For all configuration items, please visit https://github.com/serverless-components/tencent-scf/blob/master/docs/configure.md.

# Component information
component: scf # Name of the imported component, which is required. The `tencent-scf` component is used in this example
name: scfdemo # Name of the created instance, which is required. Replace it with the name of your instance

# Component parameters
inputs:
name: ${name}-${stage}-${app} # Function name
src: ./ # Code path
```

```
handler: index.main_handler # Entry
runtime: Nodejs10.15 # Function runtime environment
region: ap-guangzhou # Function region
events: # Trigger
- apigw: # Gateway trigger
parameters:
endpoints:
- path: /
method: GET
```

The `serverless.yml` file contains the following information:

Component information

Component	Required	Description
component	Yes	Component name. You can run the `sls registry` command to query components available for import.
name	Yes	Name of the created instance. An instance will be created when each component is deployed.

Parameter information

Parameters in `inputs` are component configuration parameters. The parameters of a simplest SCF component are as detailed below:

Parameter	Description
name	Function name, which also serves as a resource ID.
src	Code path.
handler	Function handler name.
runtime	Function runtime environment. Valid values: Python2.7, Python3.6, Nodejs6.10, Nodejs8.9, Nodejs10.15, Nodejs12.16, PHP5, PHP7, Go1, Java8, CustomRuntime.
region	Function region.
events	Trigger. Valid values: timer, apigw, cos, cmq, ckafka.

Common Operation Commands

Serverless Framework provides a set of operation commands for deployment orchestration, which can be used to quickly deploy function resources. For more information, please see List of Supported Commands.

For a function successfully deployed by running the `sls deploy` command, the following common operation commands provided by the SCF component can also be used.

> ⚠ **Note :**
>
> Such commands must be executed in the same directory as `serverless.yml` .

- **Publish function version**
  Run the following command to publish the `$LATEST` version of the `my-function` function as a fixed version:

```
sls publish-ver --inputs function=my-function
```

- **Create alias**
  Run the following command to create the `routing-alias` alias for the `my-function` function, with the routing rule of 50% traffic for version 1 and 50% traffic for version 2:

```
sls create-alias --inputs name=routing-alias function=my-function version=1
config='{"weights":{"2":0.5}}'
```

- **Update alias**
  Run the following command to update the flow rule of the `routing-alias` alias of the `my-function` function to 10% for version 1 and 90% for version 2:

```
sls update-alias --inputs name=routing-alias function=my-function version=1 config='{"weight
s":{"2":0.9}}'
```

- **List alias**
  Run the following command to list the `routing-alias` alias of the `my-function` function:

```
sls list-alias --inputs function=my-function
```

- **Delete alias**
  Run the following command to delete the `routing-alias` alias of the `my-function` function:

```
sls delete-alias --inputs name=routing-alias function=my-function
```

- **Trigger function**
  Run the following command to invoke the `functionName` function and pass the JSON parameter {"weights":{"2":0.1}}:

```
sls invoke --inputs function=functionName clientContext='{"weights":{"2":0.1}}'
```

Installing Serverless Framework

Last updated : 2020-11-26 15:25:42

This document describes how to quickly install Serverless Framework through binary installation or npm.

Installation Methods

Method 1. Binary installation

If Node.js is not installed in your local environment, you can install it directly in binary mode:

> ⚠ **Note :**
>
> Starting from September 1, 2020, Serverless components are no longer supported for Node.js 10.0 or above, please upgrade as needed.

macOS/Linux

Open the command line and enter the following command:

```
$ curl -o- -L https://slss.io/install | bash
```

If you have already installed a binary version, you can run the following command to upgrade it:

```
$ serverless upgrade
```

Windows

Windows supports installation through Chocolatey. Open the command line and enter the following command:

```
$ choco install serverless
```

If you have already installed a binary version, you can run the following command to upgrade it:

```
$ choco upgrade serverless
```

Method 2. Installation through npm

> **Note :**
>
> It is recommended that you use Node.js 10.0 or above; otherwise, errors may occur during Component V2 deployment.

Run the following command on the command line:

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

> **Note :**
>
> If macOS prompts that you have no permission, you need to run `sudo npm install -g serverless` for installation.

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

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

Please make sure Node.js is installed in your system environment as instructed in Node.js Installation Guide.

View Version Information

After the installation is completed, run the `serverless -v` command to view the version information of Serverless Framework:

```
$ serverless -v
```

Development Mode and In-cloud Debugging

Last updated : 2020-07-10 10:45:19

Development Mode

Serverless Framework CLI supports the development mode (`dev` mode). For projects in development mode, you can write their code and develop and debug them more easily, as 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 as shown below:

> Currently, `serverless dev` is supported by only the Node.js 10 & 12.16 runtime environment.

```
$ serverless 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 code.
-------------------- The realtime log --------------------
17:13:38 - express-api-demo - deployment
region: ap-guangzhou
apigw:
serviceId: service-b77xtixx
subDomain: service-b77xtixx-12539702xx.gz.apigw.tencentcs.com
environment: release
url: http://service-b77xtixx-12539702xx.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 updated, it will be automatically deployed again to sync the local file to the cloud.

Exiting development mode

You can press `Ctrl+C` to exit the development mode, and the following result will be returned:

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

In-cloud Debugging

You can enable in-cloud debugging for projects whose runtime environment is Node.js 10. You can use a debugging tool such as Chrome DevTools or VS Code Debugger to connect to the remote environment for debugging.

Notes

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:

- In-cloud debugging uses an actually running SCF instance for debugging.
- 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.
- When debugging is paused at a breakpoint:
  - If it stops running for a long period of time and there is no return, the trigger such as API gateway may prompt timeout.
  - 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.
- 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 900 seconds will be used as the function execution duration for statistics and measurement.
- 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 forcibly end debugging or exit debugging exceptionally, 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 adjust the timeout configuration.

Enabling in-cloud debugging

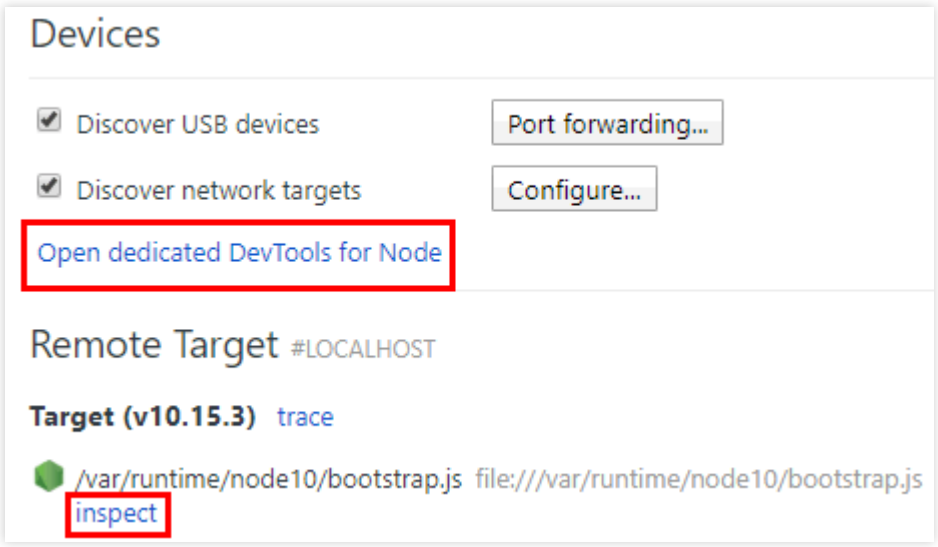
When you enter the development mode, if the project is a function whose runtime environment is Node.js 10 or above, in-cloud debugging will be automatically enabled and debugging information will be output.
For example, when you enable the development mode, if the output result contains information similar to the following content, in-cloud debugging has been enabled for this project.

```
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 code.
```

Using Chrome DevTools

The following steps are used as an example to describe 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 to access it.
3. You can open DevTools in two ways as shown below:



i. (Recommended) Click **Open dedicated DevTools for Node** under "Devices".
ii. Select **inspect** under a 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. In DevTools opened after you click **Open dedicated DevTools for Node**, you can click the **Sources** tab to view the remote code. The actual code of the function is in the `/var/user/` directory.
   On the **Sources** tab, the code that you want to view may be loaded. More remote files will be displayed as the debugging proceeds.
5. Open a file as needed and set a breakpoint at the specified position in it.
6. 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.
7. 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.

Exiting in-cloud debugging

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

List of Supported Commands

Last updated : 2020-06-24 16:54:47

Serverless Framework supports the following CLI commands:

- `serverless registry` : views the list of available components.
- `serverless registry publish` : publishes component to Serverless registry.
  `--dev` : publishes component on `@dev` version for development or test.
- `serverless deploy` : deploys component instance in cloud.
  `--debug` : lists log information such as deployment operations and status output by `console.log()` during component deployment.
  `--all` : traverses components in subdirectories under the root directory, generates dependencies, and deploys resources based on dependency order.
- `serverless remove` : removes component instance from cloud.
  `--debug` : lists log information such as removal operations and status output by `console.log()` during component removal.
  `--all` : traverses components in subdirectories under the root directory, generates dependencies, and removes resources based on dependency order.
- `serverless info` : gets and displays information related to component instance.
  `--debug` : lists more `state` values.
- `serverless dev` : starts development mode ("DEV Mode") and automatically deploys changed information when component status changes are detected. In development mode, information such as execution logs, invocation information, and errors can be output on the command line in real time. In addition, it supports in-cloud debugging for Node.js applications.

Account and Permission Configuration

Last updated : 2020-07-28 17:53:30

Currently, Serverless Framework can deploy a project properly only with the relevant role permissions in the `SLS_QcsRole` role under the account. This role contains the policies for products that are used in deployment with Serverless Framework. You can configure the permissions for a root account or sub-account.

Root Account Permission Configuration

Currently, you can grant permissions by configuring the account key. As the root account has the permissions to create roles and bind policies, you can associate it with `SLS_QcsRole` for Serverless Framework access in the following way:

Authorization through account key configuration

If you want to configure persistent environment variables/key information so that you do not need to deploy them by scanning the code every time, you can create a `.env` file under the project directory and save the `SecretId` and `SecretKey` information.

```
# .env
TENCENT_SECRET_ID=123 // Your `SecretId`
TENCENT_SECRET_KEY=123 // Your `SecretKey`
```

Serverless Framework will check whether the user is in Mainland China by default during deployment. If your development environment is outside Mainland China and you want to use Serverless Framework in the Mainland China edition, you can add the following configuration in the `.env` file to start the Mainland China edition by default, which provides an interactive quick deployment process (for more information, please see Getting Started).

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

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.

Sub-account Permission Configuration

If you want to grant a sub-account the permission to deploy by scanning code, you need to ensure that the sub-account has permissions to create roles and bind role policies. You can add the preset policy `QcloudCamRoleFullAccess` or `QcloudCamSubaccountsAuthorizeRoleFullAccess` to the sub-account.

You can also add `SLS_QcsRole` by using the root account in the CAM Console to grant access to Serverless Framework resources. The role entity is `sls.cloud.tencent.com`, which includes the following policy permissions:

- QcloudCDNFullAccess
- QcloudTCBFullAccess
- QcloudSLSFullAccess
- QcloudSSLFullAccess
- QcloudCKafkaFullAccess
- QcloudMonitorFullAccess
- QcloudVPCFullAccess
- QcloudCOSFullAccess
- QcloudAPIGWFullAccess
- QcloudSCFFullAccess

After the creation is successful, the root account needs to bind the following two policies to the sub-account:

1. Call permission policy of a specified role
2. API permission policy of Serverless Framework

Granting sub-account permission to call specified 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.

Granting sub-account permission to use APIs of Serverless Framework

Two authorization methods are provided below for your reference:

Method 1. Grant the sub-account permission to manipulate all Serverless Framework resources

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"
}
]
}
```

Method 2. Grant the 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 to 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}` .

Creating and Deploying Function

Last updated : 2020-07-10 10:53:01

Operation Scenarios

This document describes how to use the SCF component provided by Serverless Framework to quickly create and deploy an SCF project.

Prerequisites

You have installed Serverless Framework as instructed in Installing Serverless Framework.

Directions

Creating function directory

1. Run the following command on the command line to create a directory and enter it (this document uses `tencent-scf` as an example):

```
mkdir tencent-scf && cd tencent-scf
```

2. Run the following commands in sequence to quickly create an SCF application:

```
serverless create --template-url https://github.com/serverless-components/tencent-scf/tree/v2/example
```

```
cd example
```

After the application is created successfully, its directory structure is as follows

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

Deploying function

1. Enter the directory where `serverless.yml` is and run the following command to deploy the function:

```
serverless deploy
```

2. Log in to your Tencent Cloud account and grant applicable permissions. If you want to configure persistent environment variables or key information, please do so as instructed in Account Configuration.
   After the function is deployed successfully, you can view the URL provided by the gateway trigger of the corresponding function in the command line output and access the URL in a browser to view the function deployment result.

> 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).

Configuring deployment

The SCF component supports "zero" configuration deployment, that is, it can be deployed directly with the default values in the configuration file. Nonetheless, you can also modify more optional configuration items as needed to further customize the project to be deployed.
The following is the description of the SCF component configuration file `serverless.yml` . For more information, please see Full Configuration and Configuration Description.

```
# serverless.yml
component: scf # Name of the imported component, which is required. The `tencent-scf` component i
s 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 `de
v`
inputs:
name: scfFunctionName
src: ./src
runtime: Nodejs10.15 # Runtime environment of function. Valid values: Python2.7, Python3.6, Nodej
s6.10, Nodejs8.9, Nodejs10.15, Nodejs12.16, PHP5, PHP7, Golang1, 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
```

After updating the fields in the configuration file, run the `serverless deploy` or `serverless` command again to update the configuration to the cloud.

Subsequent Operations

After deploying the function, you can use the development and debugging capabilities provided by the component to re-develop the project into a production-ready application.

Serverless Framework CLI (Recommended)

Last updated : 2020-07-10 10:56:52

Operation Scenarios

Tencent Cloud SCF uses Tencent Serverless Framework. Based on serverless services (functions, triggers, etc.) in the cloud, it can implement "zero" configuration, convenient development, and rapid deployment of your first function. It supports a rich set of configuration extensions and provides the easiest-to-use, low-cost, and elastically scalable cloud-based function development, configuration, and deployment capabilities.

SCF component features:

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

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-scf && cd tencent-scf
```

Use the following command and template link to quickly create an SCF application:

```
$ serverless create --template-url https://github.com/serverless-components/tencent-scf/tree/v2/example
$ cd example
```

After download, the directory structure is as follows:

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

3. Deploy

Run `serverless deploy` in the directory under the `serverless.yml` file to deploy the function. After the deployment is completed, you can view the URL address provided by the corresponding gateway trigger of the function in the output on the command line. Then, you can click the address to view the deployment effect of the function.

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

Tencent Cloud SCF 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 is the complete configuration description for `serverless.yml` of the Tencent Cloud SCF component:

```
# serverless.yml

component: scf # (Required) Reference the name of the component, which is the `tencent-scf` component in this example
name: scfdemo # (Required) Name of the instance created by this component
org: test # (Optional) Used to record organization information. Default value: your Tencent Cloud account's `appid`
app: scfApp # (Optional) SCF application name
stage: dev # (Optional) Used to distinguish between environments. Default value: dev
```

```
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, Golang1, 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
```

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 SCF 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.

Currently, in addition to real-time log output, for Node.js applications, cloud debugging is also supported. After the `serverless dev` command is started, it will automatically listen on the remote port and set the function timeout period to 900 seconds temporarily. At this point, you can find the remote debugging path by accessing `chrome://inspect/#devices` and directly debug the code with

breakpoints. After the debugging mode ends, you need to deploy the function again to update the code and set the timeout period back to the original value.

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 SCF application. 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
```

- 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.