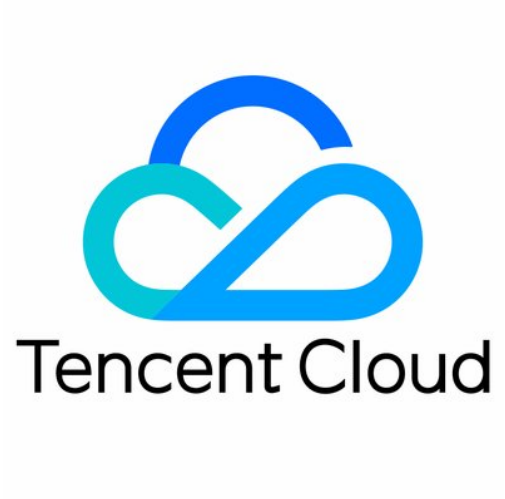
Serverless Application Center

Serverless Framework Component

Product Documentation



Contents

Serverless Framework Component Components Overview

Last updated：2024-12-02 10:48:10

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.

Serverless Components Advantages

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.

Serverless Framework Components Best Practices

@serverless/tencent-scf - SCF component
@serverless/tencent-express - Component used to quickly deploy Express.js-based backend services in SCF
@serverless/tencent-website - Component used to quickly deploy static websites in SCF

Supported Serverless Components

Currently, Serverless Components supports a rich set of development frameworks and applications in various programming languages as detailed below:
Basic components:
@serverless/tencent-postgresql - TencentDB for PostgreSQL serverless component
@serverless/tencent-apigateway - Tencent Cloud API Gateway component
@serverless/tencent-cos - Tencent Cloud COS component
@serverless/tencent-scf - Tencent Cloud SCF component

@serverless/tencent-cdn - Tencent Cloud CDN component

@serverless/tencent-vpc - Tencent Cloud VPC component

Advanced components:

@serverless/tencent-nextjs - Component used to quickly deploy Next.js-based applications in SCF

@serverless/tencent-nuxtjs - Component used to quickly deploy Nuxt.js-based applications in SCF

@serverless/tencent-express - Component used to quickly deploy Express.js-based backend services in SCF

@serverless/tencent-egg - Component used to quickly deploy Egg.js-based backend services in SCF

@serverless/tencent-koa - Component used to quickly deploy Koa.js-based backend services in SCF

@serverless/tencent-flask - Tencent Cloud Python Flask RESTful API component

@serverless/tencent-django - Tencent Cloud Python Django RESTful API component

@serverless/tencent-laravel - Tencent Cloud PHP Laravel RESTful API component

@serverless/tencent-thinkphp - Tencent Cloud ThinkPHP RESTful API component

@serverless/tencent-website - Component used to quickly deploy static websites in SCF

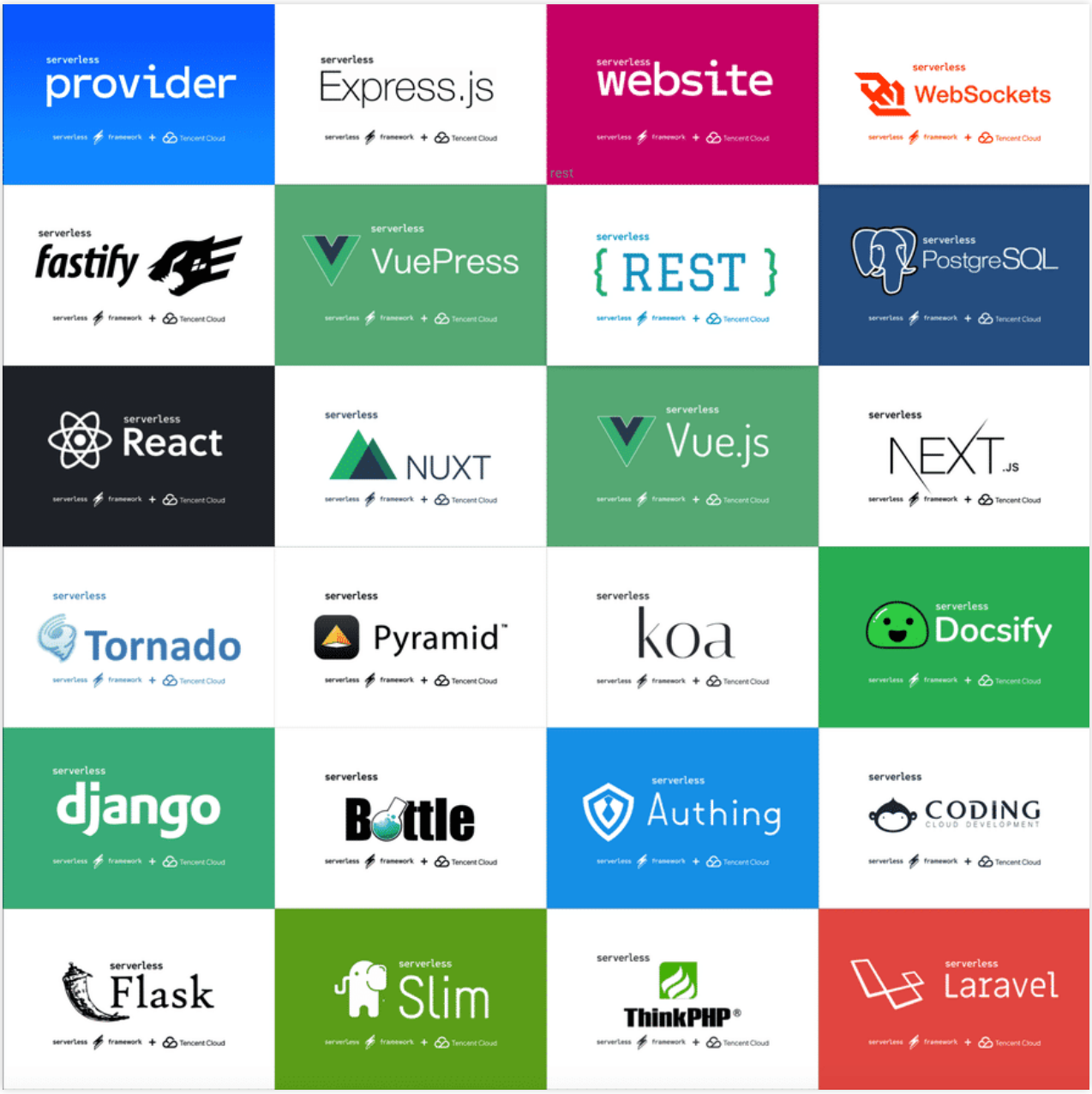
Third-party components:

@authing/serverless-oidc - Component used to quickly deploy Authing-based authentication

@twn39/tencent-fastify - Component used to quickly deploy Fastify.js-based backend services in SCF

@twn39/tencent-php-slim - Component used to quickly deploy backend services based on Slim PHP microframework in SCF

In addition, you can view all Serverless Components in the GitHub repository. Be sure to switch to the **v2** version when viewing the components.



SCF Component

Last updated：2024-12-02 10:48:10

Component Overview

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.

Getting Started

Prerequisites

You have installed Serverless Framework as instructed in Installing Serverless Framework.
Your account has the Serverless Framework permissions as detailed in Account and Permission Configuration.

Directions

Creation

Method 1. Select an SCF project template for creation as instructed in Serverless Framework.
Method 2. Directly run the `sls init` command for creation. You can quickly create an SCF function in Node.js as follows:

```
sls init scf-nodejs
```

Note:
The `helloworld` templates currently supported by the SCF component are `scf-golang`, `scf-php`, and `scf-python`. You only need to replace `scf-nodejs` in the command with the template name listed above to quickly initialize a template in the corresponding language.

Deployment

Run the following command, and a QR code will pop up. Directly scan the code to authorize for deployment:

```
sls deploy
```

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

Viewing

Run the following command to view the information of the deployed project:

```
sls info
```

Removal

Run the following command to remove the deployed project:

```
sls remove
```

Advanced Guide

serverless.yml

When `sls deploy` is executed, function resources will be created or updated according to the configuration in the `serverless.yml` file. The following is a simple `serverless.yml` file:

Note:

For more information on the configuration, please see the configuration documentation.

```
# SCF component configuration sample
# For all configuration items, please visit https://github.com/serverless-component

# Component information
component: scf # Name of the imported component, which is required. The `tencent-sc
name: scfdemo # Name of the created instance, which is required. Replace it with th

# 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
```

Information in the `serverless.yml` file:

Component information

Component	Required	Description
component	Yes	Component name. You can run `sls registry` 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. As it is also the resource ID, to ensure the uniqueness of the resource, we recommend you use the `${name}-${stage}-${app}` variable as the name.
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.

Account permission

When deploying an instance, you need to grant the corresponding account permissions to manipulate specific Tencent Cloud resources. Currently, you can authorize **by scanning the code** or **with a key**.
Authorization by scanning code: this method allows you to quickly authorize for deployment, but the generated credential is only temporary, and you need to scan the code again after the credential expires.
Authorization with key: this method grants persistent permissions, but you need to configure the `SecretId` and `SecretKey` of the account in advance.
For more information on the configuration, please see Account and Permission Configuration.

Development and debugging

You can run `sls dev` in the directory of the `serverless.yml` file to output cloud logs in real time. 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. Node.js allows you to enable the development

debugging feature, which can detect and automatically upload changes in the local code. For more information, please see Development Mode and In-cloud Debugging.

Application management

Deployment of a component instance in Serverless Framework is actually deployment of a single-component instance application.
During the development of an application project, there may be multiple component instances under the same application. For detailed directions on how to manage component instances for application project development, please see Application Management.

Component commands

The SCF component provides component-level commands. A function successfully deployed by running the `sls deploy` command can be manipulated with the following commands.

Note:

The command must be executed in the same directory as `serverless.yml` .

Publishing function version

Publish the `$LATEST` version of the `my-function` function as a fixed version:

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

Creating alias

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}}'
```

Updating alias

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 confi
```

Listing alias

List the `routing-alias` alias of the `my-function` function:

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

Deleting alias

Delete the `routing-alias` alias of the `my-function` function:

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

Triggering function

Invoke the `functionName` function and pass the JSON parameter `{"weights":{"2":0.1}}` :

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

API Gateway Component

Last updated：2024-12-02 10:48:10

Operation Scenarios

The API Gateway component is one of the basic components in the `serverless-tencent` component library. Through this component, you can create, configure, and manage API gateways with speed and ease.

Directions

Through the API Gateway component, you can perform a complete set of operations on an API service/API, such as creation, configuration, deployment, and deletion. The supported commands are as follows:

Installation

Install Serverless through npm:

```
npm install -g serverless
```

Configuration

Create the `serverless.yml` file locally:

```
touch serverless.yml
```

Configure `serverless.yml` as follows:

```
# serverless.yml

component: apigateway # Component name, which is required. `apigateway` is used in
name: apigwDemo # Instance name, which is required
app: appDemo # Next.js application name, which is optional
stage: dev # Information for identifying environment, which is optional. The defaul

inputs:
  region: ap-guangzhou
  protocols:
    - http
    - https
  serviceName: serverless
  environment: release
```

```
  endpoints:
    - path: /
      protocol: HTTP
      method: GET
      apiName: index
      function:
        functionName: myFunction
```

Detailed Configuration >>

Deployment

Run the following command to deploy by scanning code:

```
sls deploy
```

Note:

To grant persistent permission, please see Account Configuration.

Removal

You can run the following command to remove the deployed service:

```
sls remove
```

Account configuration (optional)

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

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

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

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

Note:

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

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

COS Component

Last updated：2024-12-02 10:48:10

Operation Scenarios

The COS component is one of the basic components in the `serverless-tencent` component library. Through this component, you can create, configure, and manage COS buckets with speed and ease.

Prerequisites

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

Directions

Installation

Install Serverless through npm:

```
npm install -g serverless
```

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

```
npm update -g serverless
```

Configuration

Create the `serverless.yml` file locally and configure it as follows:

```
touch serverless.yml
```

```
# serverless.yml

org: orgDemo
app: appDemo
stage: dev
component: cos
name: cosDemo
```

```
inputs:
  bucket: my-bucket
  region: ap-guangzhou
```

Detailed Configuration >>

Deployment

Deploy by running the following command, and the information below will be returned:

```
[root@iZh8dhuyhmexn3Z demo]# sls deploy

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

region: ap-guangzhou
bucket: my-bucket-xxxxxxx
url:    http://my-bucket-xxxxxxx.cos.ap-guangzhou.myqcloud.com

Full details: https://serverless.cloud.tencent.com/instances/appDemo%3Adev%3AcosDem

3s › cosDemo › Success
```

Note:

To grant persistent permission, please see Account Configuration.

Removal

Run the `sls remove` command to remove the deployed bucket, and the following information will be returned:

```
[root@iZh8dhuyhmexn3Z demo]# sls remove

serverless ⚡ framework
Action: "remove" - Stage: "dev" - App: "appDemo" - Instance: "cosDemo"

3s › cosDemo › Success
```

Account configuration (optional)

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

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

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

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

Note:

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

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

CDN Component

Last updated：2024-12-02 10:48:10

Operation Scenarios

The CDN component is one of the basic components in the `serverless-tencent` component library. Through this component, you can create, configure, and manage CDN services with speed and ease.

Prerequisites

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

Directions

Installation

Install Serverless through npm:

```
npm install -g serverless
```

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

```
npm update -g serverless
```

Configuration

Create the `serverless.yml` file locally:

```
touch serverless.yml
```

Configure `serverless.yml` as follows:

```
# serverless.yml

component: cdn
name: cdnDemo
app: appDemo
stage: dev
```

```
inputs:
  area: overseas
  domain: mysite.com # Domain name
  origin:
    origins:
      - xxx.cos.ap-guangzhou.myqcloud.com  # Origin server, which can be a domain n
    originType: cos
    originPullProtocol: https
  serviceType: web
  forceRedirect:
    switch: on
    redirectType: https
    redirectStatusCode: 301
  https:
    switch: on
    http2: on
    certInfo:
      certId: 'abc'
      # certificate: 'xxx'
      # privateKey: 'xxx'
```

Detailed Configuration >>

Deployment

Run the following command to deploy by scanning code:

```
sls deploy
```

Note:

Make sure that you have activated CDN.

To grant persistent permission, please see Account Configuration.

Removal

Run the following command to remove the deployed CDN configuration:

```
sls remove
```

Account configuration (optional)

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

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

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

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

Note:

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

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

VPC Component

Last updated：2024-12-02 10:48:10

Operation Scenarios

Tencent Cloud VPC component supports configuring `serverless.yml` to quickly create VPCs and subnets with specified names and output `VPCID` and `SubnetID` in order to facilitate the configuration of network information required by other components.

Directions

Installation

Install the latest version of Serverless Framework through npm:

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

Configuration

Create a `vpcDemo` directory and create a `serverless.yml` file in it.

```
$ mkdir vpcDemo && cd vpcDemo
$ touch serverless.yml
```

Configure `serverless.yml` as follows:

```
# serverless.yml
org: orgDemo # Organization information, which is optional. The default value is th
app: appDemo # VPC application name, which is optional.
stage: dev # Information for identifying environment, which is optional. The defaul

component: vpc # Name of the imported component, which is required. The `tencent-vp
name: vpcDemo # Name of the instance created by this component, which is required.

inputs:
  region: ap-guangzhou
  zone: ap-guangzhou-2
  vpcName: serverless
  subnetName: serverless
```

Detailed Configuration >>

Deployment

Run `sls deploy` to deploy:

```
$ sls deploy
serverless ⚡ framework
Action: "deploy" - Stage: "dev" - App: "appDemo" - Instance: "vpcDemo"

region:     ap-guangzhou
zone:       ap-guangzhou-2
vpcId:      vpc-xxxxxxxx
vpcName:    serverless
subnetId:   subnet-xxxxxxxx
subnetName: serverless


3s › vpcDemo › Success
```

Note:

`sls` is short for the `serverless` command.

Information viewing

Run `sls info` to view the information of successful deployment:

```
$ sls info

serverless ⚡ framework

Status:       active
Last Action:  deploy (5 minutes ago)
Deployments:  2

region:     ap-guangzhou
zone:       ap-guangzhou-2
vpcId:      vpc-xxxxxxx
vpcName:    serverless
subnetId:   subnet-xxxxxxx
subnetName: serverless

vpcDemo › Info successfully loaded
```

Removal

You can run the following commands to remove the deployed VPC:

```
$ sls remove
```

```
serverless ⚡ framework
Action: "remove" - Stage: "dev" - App: "appDemo" - Instance: "vpcDemo"

6s › vpcDemo › Success
```

Account configuration (optional)

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

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

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

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

Note:

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

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

Layer Component

Last updated：2024-12-02 10:48:09

Overview

The Layer component is one of the basic components in the `serverless-tencent` component library. Through this component, you can create, configure, and manage SCF layer resources with speed and ease.

Prerequisites

Node.js has been installed.

Note:

Starting from September 1, 2020, Serverless components no longer support Node.js versions below 10.0. Please upgrade if needed.

Directions

Installation

Install Serverless through npm:

```
npm install -g serverless
```

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

```
npm update -g serverless
```

Configuration

Create the `serverless.yml` file locally and configure it as follows:

```
touch serverless.yml
```

```
# serverless.yml

component: layer
name: layerDemo
```

```
app: appDemo
stage: dev

inputs:
  region: ap-guangzhou
  name: layerDemo
  src: ./layer-folder
  runtimes:
    - Nodejs10.15
```

Detailed Configuration >>

Deployment

Run the following command to deploy by scanning code:

```
sls deploy
```

Note:

The QR code has a validity period. To grant persistent permission, please see Account Configuration.

Removal

You can run the following command to remove the deployed service:

```
sls remove
```

Account configuration (optional)

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

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

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

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

Note:

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

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

PostgreSQL Component

Last updated：2024-12-02 10:48:10

Overview

PostgreSQL for Serverless (ServerlessDB) is a database product that allocates resources on demand based on PostgreSQL. Its database automatically allocates resources based on your actual number of requests. With PostgreSQL for Serverless, you can create a database instance for easy use without caring about the instance specifications. You only need to pay for the actual usage when the database is active.
Through the PostgreSQL for Serverless component, you can create, configure, and manage PostgreSQL instances with speed and ease.
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: the default configuration will be done by Serverless.
Fast deployment: you can create or update your database in just a few seconds.
Convenient collaboration: the database status information and deployment logs in the cloud make multi-person collaborative development easier.

Directions

Installation

Use npm to install Serverless CLI globally:

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

Account configuration

Create the `.env` file locally:

```
$ 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.

Configuration

Create a directory and enter it:

```
$ mkdir tencent-postgreSQL && cd tencent-postgreSQL
```

Create a `serverless.yml` file in a new directory:

```
$ touch serverless.yml
```

Configure `serverless.yml` as follows:

```
# serverless.yml
component: postgresql # Name of the imported component, which is required. The `pos
name: serverlessDB # Name of the instance created by this component, which is requi
org: test # Organization information, which is optional. The default value is the `
app: serverlessDB # SQL application name, which is optional
stage: dev # Information for identifying environment, which is optional. The defaul

inputs:
  region: ap-guangzhou  # Valid values: ap-guangzhou, ap-shanghai, ap-beijing
  zone: ap-guangzhou-2  # Valid values: ap-guangzhou-2, ap-shanghai-2, ap-beijing-3
  dBInstanceName: serverlessDB
  vpcConfig:
    vpcId: vpc-xxxxxxx
    subnetId: subnet-xxxxxx
  extranetAccess: false
```

The PostgreSQL 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.

Detailed Configuration >>

Note:

Currently, PostgreSQL for Serverless is available for creation and deployment only in **Beijing Zone 3**, **Guangzhou Zone 2**, and **Shanghai Zone 2**. Therefore, when entering the region and AZ information in the `yaml` file, please be sure to use the correct region and corresponding VPC and subnet information.

Deployment

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

Note:

`sls` is short for the `serverless` command.

```
$ sls deploy
```

Removal

You can run the following commands to remove the deployed database instance:

```
$ sls remove
```

Best Practice

After deploying the PostgreSQL Serverless database, you can refer to Deploying Full-Stack Website with Vue + Express + PostgreSQL to use this database instance.

More Components

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