Serverless 应用中心

进阶指南

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

进阶指南
开发项目

最近更新时间：2024-07-18 14:09:11

前提条件

在开始项目开发之前，请确保您已经了解以下内容：

- 快速部署
- Serverless 应用
- 账号和权限配置

开发流程

一个项目的开发上线流程大致如下：

- **初始化项目**：将项目进行初始化。例如选择一些开发框架和模板完成基本的搭建工作。
- **开发阶段**：对产品功能进行研发。可能涉及到多个开发者协作，开发者拉取不同的 feature 分支，开发并测试自己负责的功能模块；最后合并到 dev 分支，联调各个功能模块。
- **测试阶段**：测试人员对产品功能进行测试。
- **发布上线**：对于已完成测试的产品功能发布上线。由于新上线的版本可能有不稳定的风险，所以一般会进行灰度发布，通过配置一些规则监控新版本的稳定性，等到版本稳定后，流量全部切换到新版本。

环境隔离

在开发项目的每个阶段，我们都需要一个独立运行的环境来对开发的操作进行隔离。
在 serverless.yml 文件中定义 stage，并把 stage 作为参数写入到组件的资源名称中，部署时以 `实例名 -{stage}- 应用名` 的方式生成资源。这样我们在不同阶段只要定义不同的 stage 就可以生成不同的资源，达到环境隔离的目的。
以 SCF 组件的 serverless.yml 为例：

```
# 应用信息
app: myApp
stage: dev # app环境名称，默认为dev
```

```
# 组件信息
component: scf
name: scfdemo

# 组件参数
inputs:
  name: ${name}-${stage}-${app} #函数名称，以变量 ${stage} 作为资源名称的一部分
  src: ./
  handler: index.main_handler
  runtime: Nodejs10.15
  region: ap-guangzhou
  events:
    - http:
        parameters:
          netConfig:
            enableIntranet: true
            enableExtranet: true
          qualifier: $DEFAULT
          authType: NONE
```

- 云函数 name 定义为 `${name}-${stage}-${app}` 。
- 开发测试阶段定义 stage 为 `dev` ，部署后云函数为 scfdemo-dev-myApp。
- 上线发布阶段定义 stage 为 `pro` ，部署后云函数为 scfdemo-pro-myApp。
- 不同阶段操作不同的云函数资源，从而达到开发与发布隔离的目的。

> **说明：**
> stage 可以直接在 serverless.yml 文件中定义，也可以通过 `scf deploy --stage dev` 直接传参。

如您仍在使用 API 网关，请将 `events` 修改为：

```
events:
    - apigw:
        parameters:
          endpoints:
            - path: /
              method: GET
```

权限管理

在开发项目中，需要对不同的人员进行权限分配和管理。例如对于开发人员，只允许其访问某个项目某个环境下的权限，可以参考 账号和权限配置，授予子账号 Serverless Cloud Framework 特定资源的操作权限。
以 myApp 项目 dev 环境为例，配置如下：

```
{
    "version": "2.0",
    "statement": [
        {
            "action": [
                "scf:*"
            ],
            "resource": "qcs::scf:ap-
guangzhou::appname/myApp/stagename/dev", #app 为 myApp,stage 为 dev
            "effect": "allow"
        }
    ]
}
```

灰度发布

灰度发布（又名金丝雀发布）是指在黑与白之间，能够平滑过渡的一种发布方式 。为保证线上业务的稳定性，开发上线项目推荐使用灰度发布。
Serverless 应用的灰度发布支持两种方式：**默认别名与自定义别名**。更多详情参考 Serverless 灰度发布 。

对比项	配置	流量规则设置	适用组件
默认别名	配置简单	只能在最后一次发布的函数版本和 $latest 版本间进行。	• 云函数组件 • 涉及云函数的相关组件
自定义别名	配置灵活	可以在两个任意函数版本间进行。	云函数组件

Serverless Cloud Framework 命令

开发项目到上线过程中，需要用到一些 serverless-cloud-framework 的相关命令。更多命令请查看 Serverless Cloud Framework 支持命令列表 。

> **说明：**
> `serverless-cloud-framework` 命令的简写为 `scf` 。

初始化项目：

```
scf
```

下载模板项目 scf-starter，模板支持可通过 scf registry 查询：

```
scf init scf-starter
```

下载模板项目 scf-starter，并初始化该项目为 myapp：

```
scf init scf-starter --name my-app
```

部署应用：

```
scf deploy
```

部署应用，指定 stage 为 dev：

```
scf deploy --stage  dev
```

部署应用，并打印部署信息：

```
scf deploy --debug
```

部署并发布函数版本：

```
scf deploy --inputs publish=trues
```

部署并切换20%流量到 $latest 版本：

```
scf deploy --inputs traffic=0.2
```

项目实践

参考 开发上线 Serverless 应用 。

灰度发布

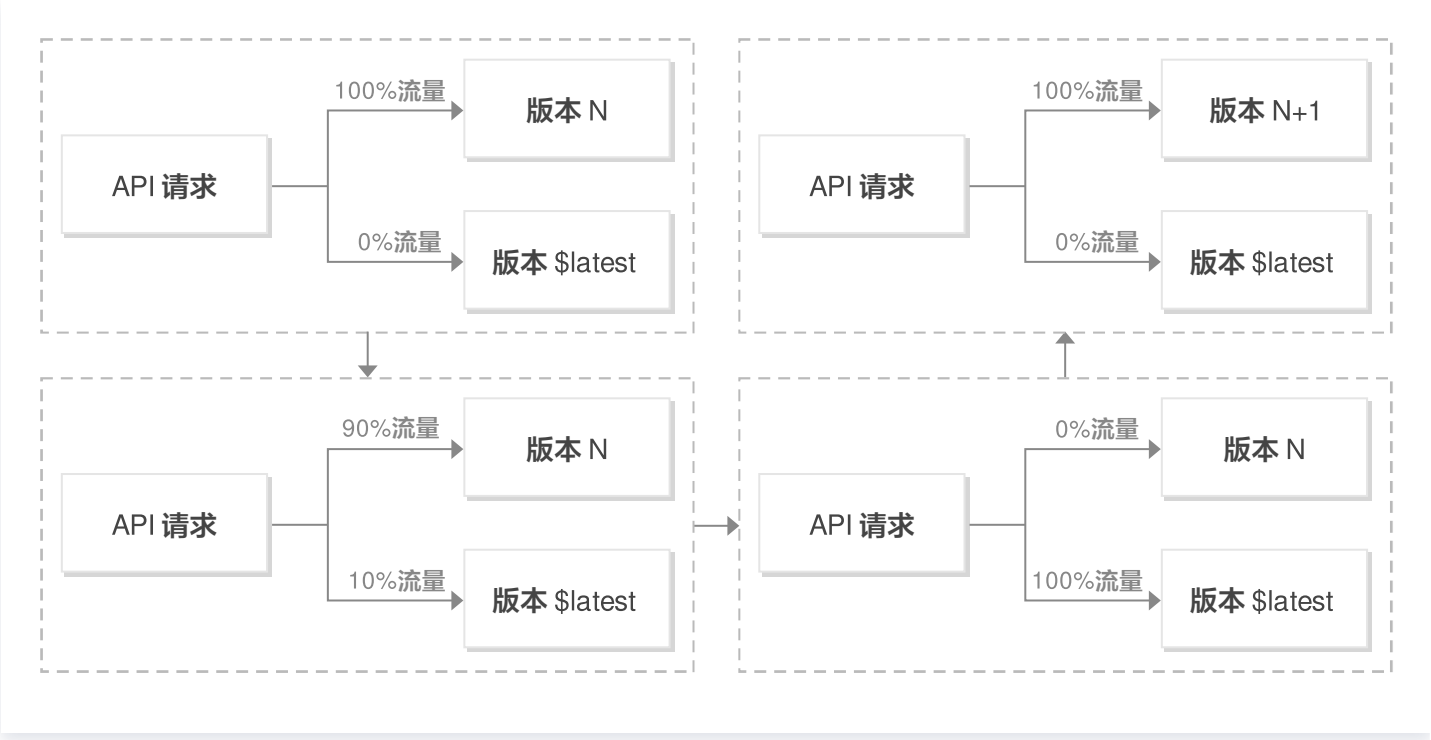
最近更新时间：2024-03-28 17:36:41

概述

灰度发布（又名金丝雀发布）是指在黑与白之间，能够平滑过渡的一种发布方式。Serverless 应用的灰度发布是配置云函数别名的流量规则，针对别名中两个不同版本的云函数进行流量规则配置。Serverless Cloud Framework 中可通过**默认别名**进行流量规则配置。

默认别名

默认别名是配置云函数的 $default（默认流量）别名。该别名中固定有两个云函数版本，一个为 $latest 版本，一个为最后一次函数发布的版本。部署时配置的 traffic 参数为 $latest 版本流量占比，默认另一部分流量切到当前云函数最后一次发布的版本。
每次上线一个新功能，执行 `scf deploy` 会部署到 $latest 版本上。我们将切部分流量在 $latest 版本上进行观察，然后逐步将流量切到 $latest 版本。当流量切到 100% 时，我们会固化这个版本，并将流量全部切到固化后的版本。



命令说明

说明：

> 旧版本命令为 `scf deploy --inputs.key=value` 。Serverless CLI V3.2.3 后命令统一格式为 `scf deploy --inputs key=value` ，旧版本命令在新版本 Serverless CLI 中不可用，升级 Serverless CLI 的用户请使用新版本命令。

函数发布版本

部署时发布项目下所有函数版本：

```
scf deploy --inputs publish=true
```

函数流量设置

部署后切换 20% 流量到 $latest 版本：

```
scf deploy --inputs traffic=0.2
```

- Serverless Cloud Framework 流量切换修改的是云函数别名为 $default 的流量规则。
- 每次配置针对的是 $latest，最后一次云函数发布的版本的配置。
- traffic 配置的值为 $latest 版本对应的流量占比，最后一次云函数发布的版本的流量占比为 1-$latest 流量占比。例如 traffic=0.2，实则配置 $default 的流量规则为 {$latest:0.2, 最后一次云函数发布的版本:0.8}。
- 如果函数还未发任何固定版本，只存在 $latest 版本的函数情况下，traffic 无论如何设置，都会是 $latest:1.0。

操作步骤

当一个功能测试完毕，需要进行灰度发布，操作如下：

1. 配置生产环境信息到 .env 文件（ STAGE=prod 为生产环境 ）：

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

2. 部署到线上环境 $latest，并切换 10% 的流量在 $latest 版本（ 90% 的流量在最后一次发布的云函数版本 N 上）：

```
scf deploy --inputs traffic=0.1
```

3. 对 $latest 版本进行监控与观察，等版本稳定之后把流量 100% 切到该版本上：

```
scf deploy --inputs traffic=1.0
```

4. 流量全部切换成功后，对于一个稳定版本，我们需要对它进行标记，以免后续发布新功能时，如果遇到线上问题，方便快速回退版本。部署并发布函数版本 N+1，切换 100% 流量到版本 N+1：

```
scf deploy --inputs publish=true  traffic=0
```

层部署使用指引

最近更新时间：2024-12-17 16:00:33

由于云函数限制，目前只支持上传小于50MB的代码包，当您的项目过大时，您可以将依赖放在层中而不是部署包中，可确保部署包保持较小的体积。层的具体使用请参考 层管理相关操作 。

创建层

新建层并上传依赖，您可以通过以下两种方式操作：

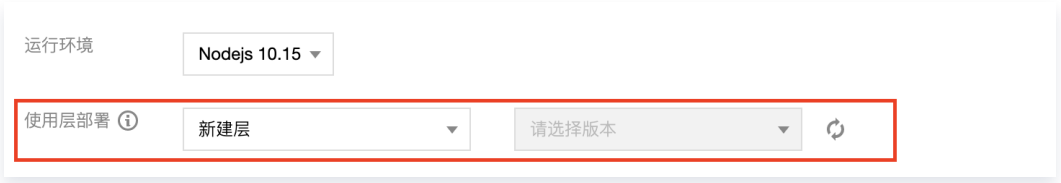
- 通过 Serverless 应用控制台 创建
- 使用 Serverless Cloud Framework 的 Layer 组件（参考 Layer 组件 ）

使用层

您可以通过控制台配置和本地配置两种方法，在项目配置中使用层部署，具体如下：

控制台配置

- 对于 Node.js 框架应用，Serverless Cloud Framework 会自动为您创建名为 `${appName}-layer` 的层，并自动帮您把应用的依赖项 node_modules 上传到该层中。
- 导入已有项目时，您也可以选择使用新建层或已有层完成部署，选择新建层时，Serverless Cloud Framework 会自动帮您把应用的依赖项 node_modules 上传到该层中。



> **说明：**
> 新建层操作仅支持 Node.js 框架，其它框架使用层时，请确保已经完成层的创建并已经把相关依赖项上传到层中。

通过 Layer 组件配置

1. 此处以 Next.js 组件为例，调整本地项目目录，新增 layer 文件夹，并创建 **serverless.yml** 文件，完成层的名称与版本配置，yml 模板如下：

```
app: appDemo
stage: dev

component: layer
name: layerDemo
```

```
inputs:
 name: test
 region: ap-guangzhou
 src: ../node_modules #需要上传的目标文件路径
 runtimes:
- Nodejs10.14
```

查看详细配置，请参考 layer 组件全量配置文档 。
更新后的项目目录结构如下：

```
.
├── node_modules
├── src
|   ├── serverless.yml # 函数配置文件
|   └── index.js # 入口函数
├── layer
|   └── serverless.yml # layer 配置文件
└── .env # 环境变量文件
```

2. 打开项目配置文件，增加 layer 配置项，并引用 layer 组件的输出作为项目配置文件的输入，模板如下：

```
app: appDemo
stage: dev

component: nextjs
name: nextjsDemo

inputs:
  src:
 src: ./
 exclude:
   - .env

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

```
  - name: ${output:${stage}:${app}:layerDemo.name} #  layer 名称
  version: ${output:${stage}:${app}:layerDemo.version} #  版本
```

引用格式请参考 变量引用说明 。

3. 在项目根目录下，执行 `scf deploy` ，即可完成 Layer 的创建，并将 Layer 组件的输出作为 Next.js 组件的输入完成层的配置。

变量引用说明

`serverless.yml` 支持多种方式引用变量：

- 顶级参数引用

  在 `inputs` 字段里，支持直接引用顶级配置信息，引用语法如下： `${org}` 、 `${app}` 。

- 环境变量引用

  在 `serverless.yml` 中，可以直接通过 `${env}` 的方式，直接引用环境变量配置（包含 .env 文件中的环境变量配置，以及手动配置在环境中的变量参数）

  例如，通过 `${env:REGION}` ，引用环境变量 REGION。

- 引用其它组件输出结果

  如果希望在当前组件配置文件中引用其他组件实例的输出信息，可以通过如下语法进行配置：

  `${output:[app]:[stage]:[instance name].[output]}`

示例 yml：

```
org: xxx
app: demo
component: scf
name: rest-api
stage: dev

inputs:
  name: ${org}-${stage}-${app}-${name} # 命名最终为 "acme-prod-ecommerce-
rest-api"
  region: ${env:REGION} # 环境变量中指定的 REGION= 信息
  vpcName: ${output:prod:my-app:vpc.name} # 获取其他组件中的输出信息
  vpcName: ${output:${stage}:${app}:vpc.name} # 上述方式也可以组合使用
```

自定义域名及 HTTPS 访问配置

最近更新时间：2024-08-23 17:22:11

通过 Serverless Component 快速构建一个 Serverless Web 网站服务后，如果您希望配置自定义域名及支持 HTTPS 的访问，则可以按照本文提供的两种方案快速配置。

前提条件

- 已经部署了网站服务，获取了 COS/API 网关的网站托管地址。具体部署方法请参见 快速部署 Hexo 博客 。
- 已拥有自定义域名（例如 www.example.com），并确保输入的域名已 备案 。
- 如果需要 HTTPS 访问，可以申请证书并且 获得证书 ID （例如：certificateId：axE1bo3），个人站点可以直接申请 免费 SSL 证书申请流程 。

方案一：通过 CDN 加速配置支持自定义域名的 HTTPS 访问

配置前，需要确保账号实名并已经 开通 CDN 服务 。

增加配置

在 `serverless.yml` 中，增加 CDN 自定义域名配置：

```
# serverless.yml

component: website
name: myWebsite
app: websiteApp
stage: dev

inputs:
  src:
    src: ./public
    index: index.html
    error: index.html
  region: ap-guangzhou
  bucketName: my-hexo-bucket
  protocol: https
  # 新增的 CDN 自定义域名配置
  hosts:
    - host: www.example.com # 希望配置的自定义域名
      https:
        switch: on
        http2: off
```

```
        certInfo:
          certId: 'abc'
          # certificate: 'xxx'
          # privateKey: 'xxx'
```

查看完整配置项说明 >>

部署服务

再次通过 `scf deploy` 命令进行部署，并可以添加 `--debug` 参数查看部署过程中的信息。
如您的账号未 登录 或 注册 腾讯云，您可以直接通过微信扫描命令行中的二维码进行授权登录和注册。

> **说明：**
> `serverless-cloud-framework` 命令的简写为 `scf` 。

```
$ scf deploy

    myWebsite:
      url:  https://my-hexo-bucket-1250000000.cos-website.ap-
guangzhou.myqcloud.com
      env:
      host:
        - https://www.example.com (CNAME: www.example.com.cdn.dnsv1.com)
    17s › myWebsite › done
```

添加 CNAME

部署完成后，在命令行的输出中可以查看到一个以 `.cdn.dnsv1.com` 为后缀的 CNAME 域名。参考 CNAME 配置文档，在 DNS 服务商处设置好对应的 CNAME 并生效后，即可访问自定义 HTTPS 域名。

方案二：对 API 网关域名进行自定义域名配置

增加配置

在 `serverless.yml` 中，增加 API 网关自定义域名配置。本文以 egg.js 框架为例，配置如下：

```
# serverless.yml

component: apigateway # （必填）组件名称，此处为 apigateway
name: restApi # （必填）实例名称
app: appDemo # （可选）该应用名称
```

```
stage: dev # （可选）用于区分环境信息，默认值为 dev

inputs:
   region: ap-shanghai
   protocols:
     - http
     - https
   serviceName: serverless
   environment: release
   customDomains:
     - domain: www.example.com
       # 如要添加 https，需先行在腾讯云 - SSL 证书进行认证获取 certificateId
       certificateId: abcdefg
       protocols:
         - http
         - https
   endpoints:
     - path: /users
       method: POST
       function:
         functionName: myFunction # 网关所连接函数名
```

查看完整配置项说明 >>

部署服务

再次通过 `scf deploy` 命令进行部署，并可以添加 `--debug` 参数查看部署过程中的信息。
如您的账号未 登录 或 注册 腾讯云，您可以直接通过微信扫描命令行中的二维码进行授权登录和注册。

```
$ scf deploy
    restApi:
      protocols:
        - http
        - https
      subDomain:     service-lqhc88sr-1250000000.sh.apigw.tencentcs.com
      environment:   release
      region:        ap-shanghai
      serviceId:     service-lqhc88sr
      apis:
        -
          path:   /users
          method: POST
```

```
          apiId:  api-e902tx1q
      customDomains:
        - www.example.com (CNAME: service-lqhc88sr-
1250000000.sh.apigw.tencentcs.com)
    8s › restApi › done
```

添加 CNAME 记录

部署完成后，在命令行的输出中可以查看到一个以 `.apigw.tencentcs.com` 为后缀的 CNAME 域名。参考 添加 CNAME 记录，在 DNS 服务商处设置好对应的 CNAME 并生效后，即可访问自定义 HTTPS 域名。