Serverless 应用中心

实践教程

腾讯云

文档目录

实践教程
开发上线 Serverless 应用
开发项目

最近更新时间：2024-08-23 17:22:11

操作场景

本文以 tencent-express 组件部署一个 Express 网站为例， 模拟 Serverless Cloud Framework 开发项目、管理项目和部署发布上线全流程。示例链接 >>
开发项目过程可能会涉及以下分支：

分支类型	说明
master	用于生产环境部署。
testing	用于测试环境测试。
dev	用于日常开发。
feature-xxx	用于增加一个新功能，例如不同开发者会从 dev 拉取不同的特性分支进行开发。
hotfix-xxx	用于修复一个紧急 bug。

操作步骤

初始化项目

1. 参考 部署 Express.js 应用 文档，创建一个 express 项目，修改 yml 文件为以下内容：

```
#serverless.yml

app: expressDemoApp #  应用名称，默认为与组件实例名称
stage: ${env:STAGE} #  用于开发环境的隔离，默认为dev


component: http # (必填) 引用 component 的名称，当前用到的是 express-
tencent 组件
name: expressDemo # (必填) 组件创建的实例名称

inputs:
```

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

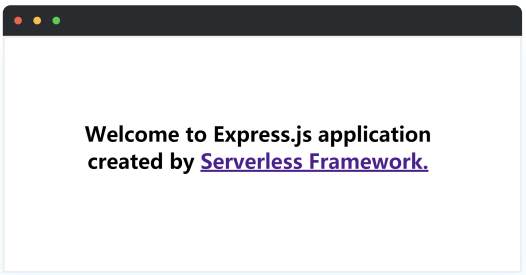
如您使用的是函数 URL，请将 `apigatewayConf` 修改为：

```
  faas:
    events:
      - http:
          parameters:
            netConfig:
              enableIntranet: false
              enableExtranet: true
            qualifier: $DEFAULT
            authType: NONE
```

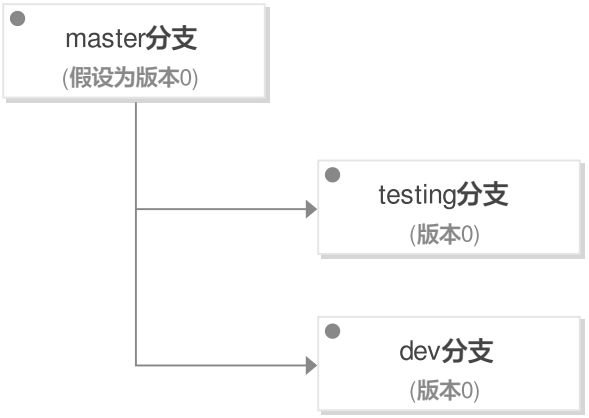
2. 在项目根目录下的 .env 文件中配置：

```
TENCENT_SECRET_ID=xxxxxxxxxx #您账号的 SecretId
TENCENT_SECRET_KEY=xxxxxxxx #您账号的 SecretKey
STAGE=prod #STAGE为prod环境，也可以scf deploy --stage prod 参数传递的方式设置
```

3. 执行 scf deploy 部署成功后，访问生成的 url 链接，效果如下：



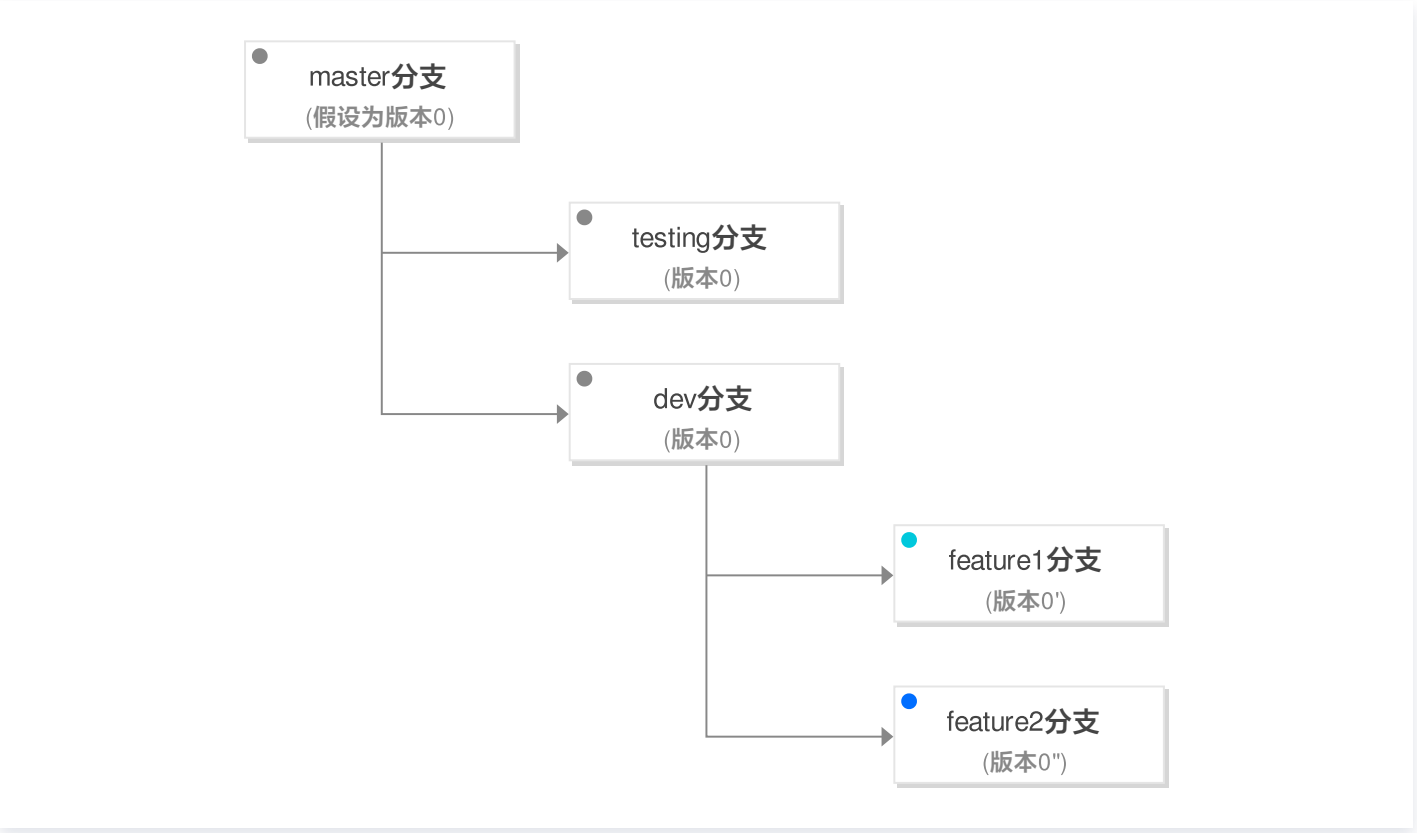
4. 创建远程仓库（示例链接），将项目代码提交到远程 master 分支。同时创建 testing、dev。此时三个分支的代码在同一个版本上（假设为版本0）。



开发与测试

背景

现在需要开发某个功能模块。假设需要有两位开发者：Tom、Jorge。两位开发者分别从 dev（版本0）上创建特性分支为 feature1、feature2 进行研发。



Tom 开始开发 feature1。在本示例中，为新增一个 feature.html，里面添加文案 "This is a new feature 1."。

开发

1. 在 scf.js 文件中新增路由器配置：

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

2. 新增 feature.html：

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

```
 <body>
  <h1>
   This is a new feature 1.
  </h1>
 </body>
</html>
```

3. 在 .env 文件中设置自己的 stage，以便在开发过程中得到独立的运行和调试环境。
   例如 Tom 在 serverless.yml 的项目目录下配置 .env 如下：

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

4. 执行 scf deploy 部署成功后，返回显示如下：

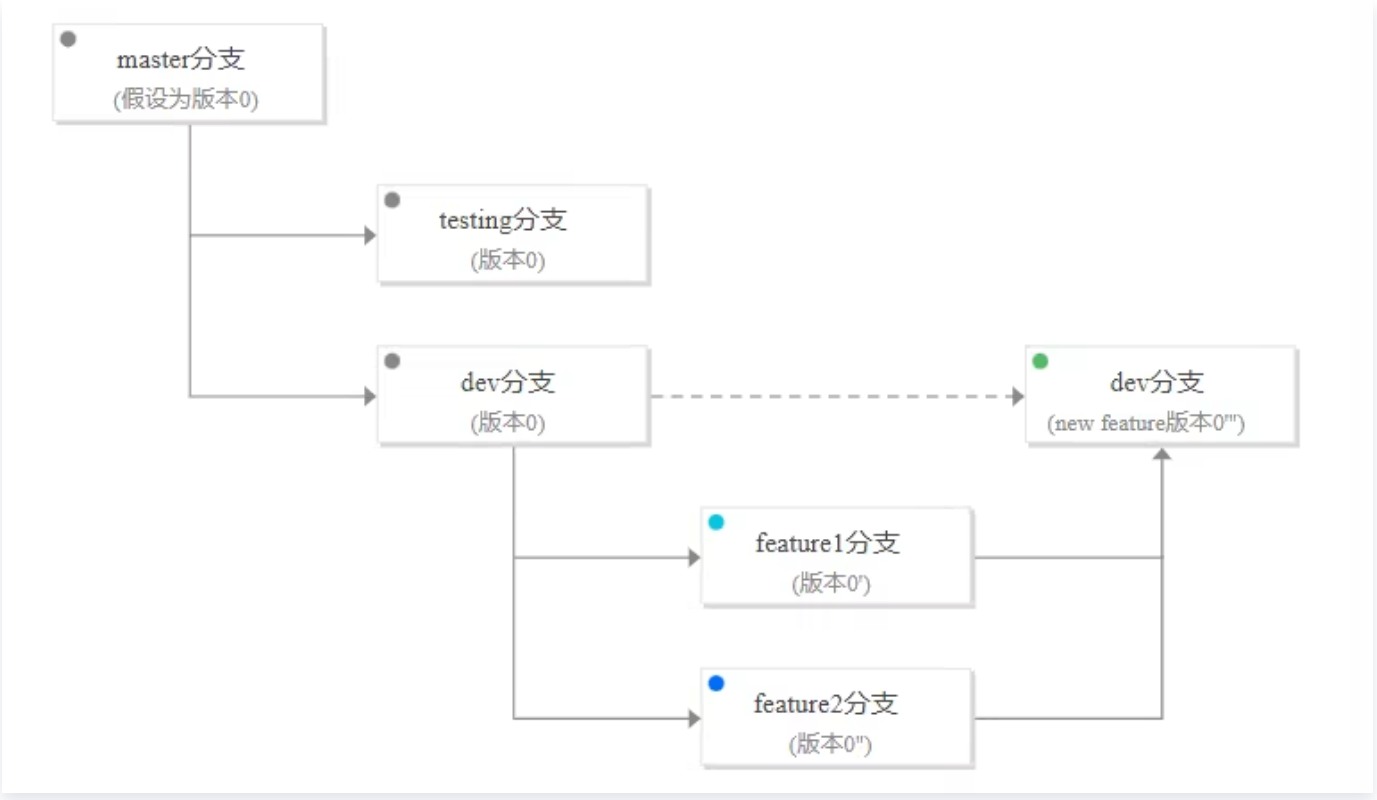
```
region: ap-guangzhou
apigw:
  serviceId:   service-xxxxxx
  subDomain:   service-xxxxxx-123456789.gz.apigw.tencentcs.com
  environment: release
  url:         https://service-xxxxxx-
123456789.gz.apigw.tencentcs.com/release/
scf:
  functionName: express-demo-feature1
  runtime:      Nodejs10.15
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

Full details:
https://serverless.cloud.tencent.com/instances/expressDemoApp%3Afeatur
e1%3AexpressDemo

10s » expressDemo » Success
```

5. 访问生成的 url（https://service-xxxxxx-123456789.gz.apigw.tencentcs.com/release/feature），效果如下：

至此，Tom 开发功能完成并自测通过。假设同时，Jorge 同时也完成自己的特性开发，并自测通过。在本示例中，为新增一个 feature.html，里面添加文案 "This is a new feature 2."。

联调

1. 两人把各自 feature 分支的代码合并到 dev 分支。（可能会存在冲突需要人为解决）



2. 在 dev 进行联调。联调环境中的 .env 配置如下：

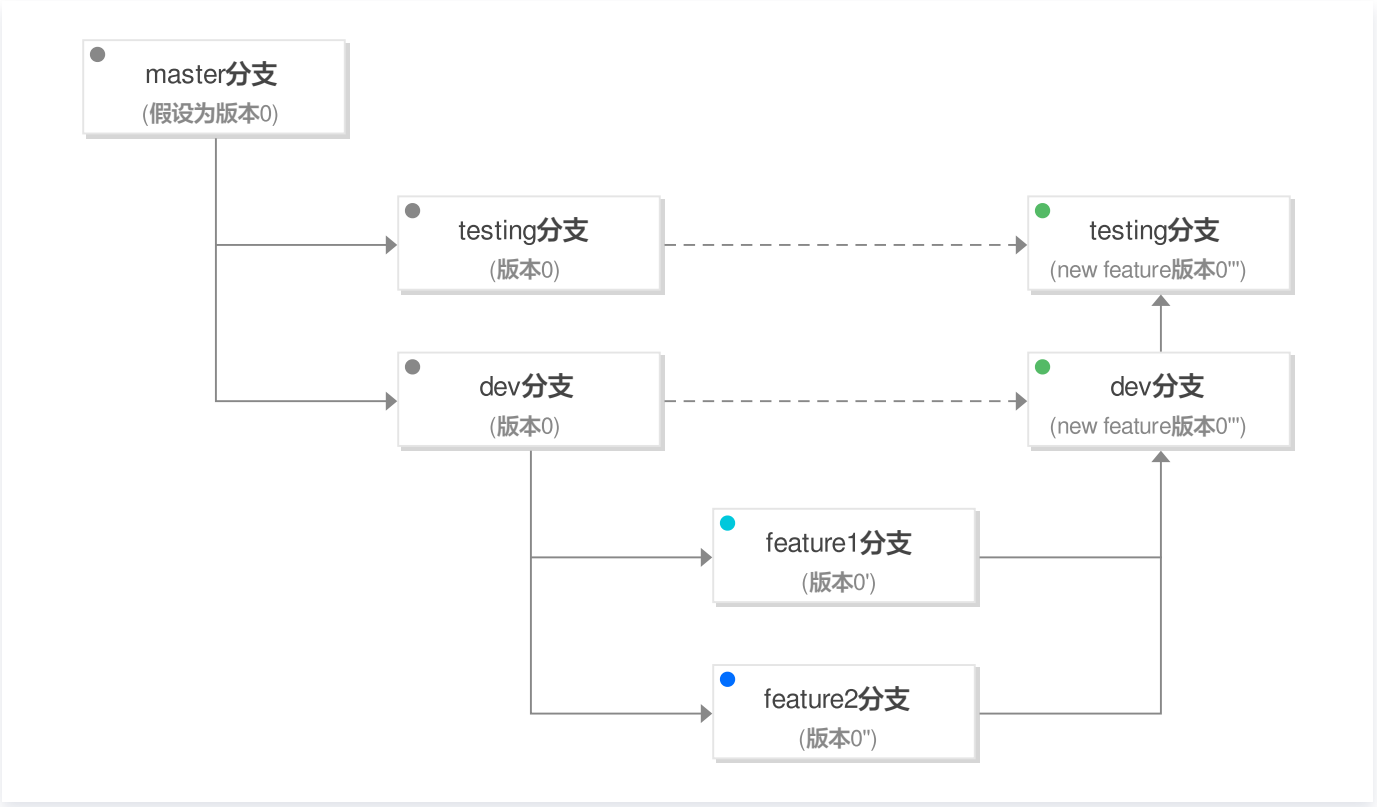
```
TENCENT_SECRET_ID=xxxxxxxxxx
TENCENT_SECRET_KEY=xxxxxxxx
STAGE=dev
```

3. 执行 `scf deploy` 联调部署后，访问 url（https://service-xxxxxx-123456789.gz.apigw.tencentcs.com/release/feature），效果如下：

至此联调完成，整个功能已经开发完毕。

测试

1. 把联调通过的 dev 分支合并到 testing 代码，进入测试。



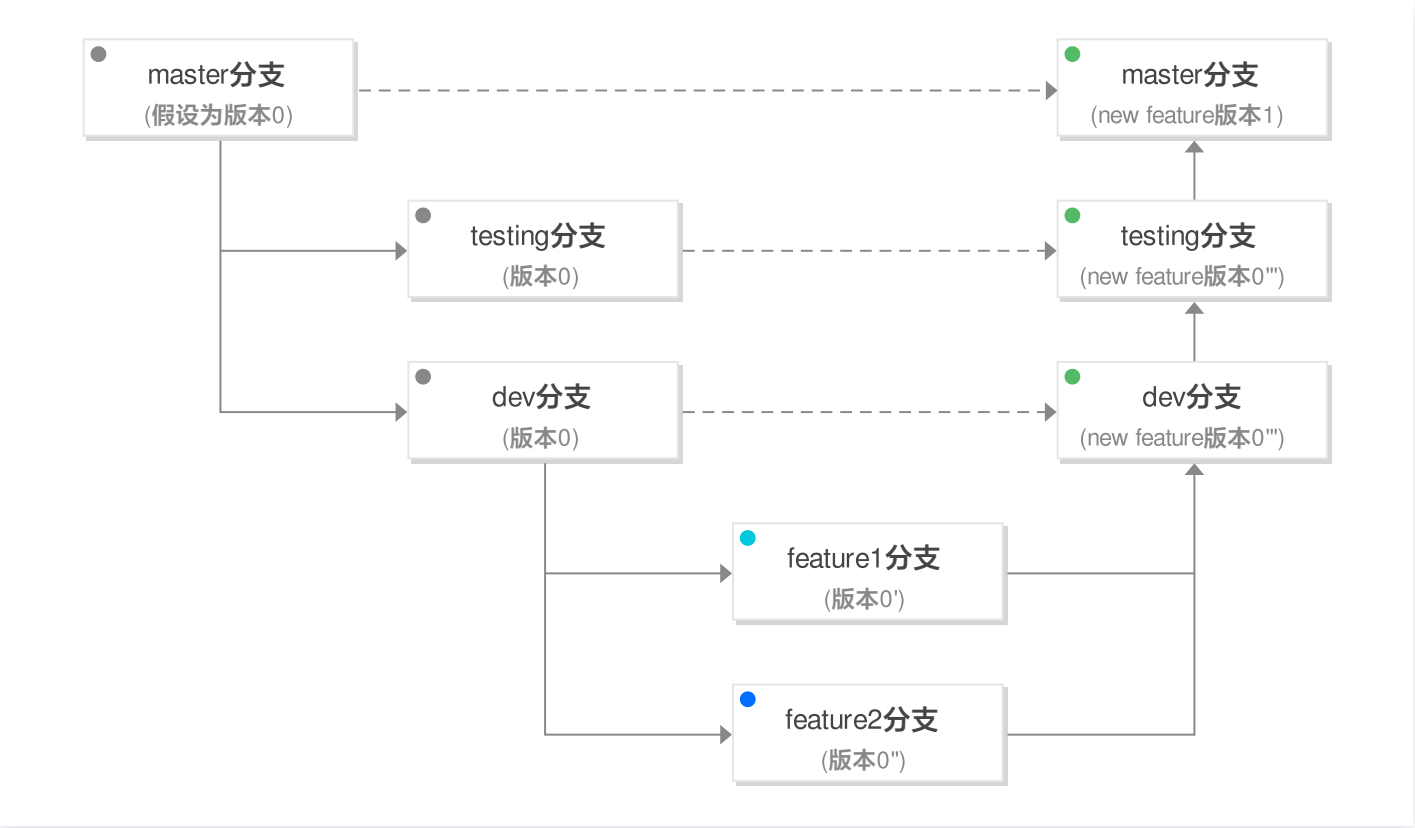
2. 测试环境中的 .env 配置如下：

```
TENCENT_SECRET_ID=xxxxxxxxxx
TENCENT_SECRET_KEY=xxxxxxxx
STAGE=testing
```

3. 执行 `scf deploy` 部署成功后，测试人员开始进行相关测试，直至功能稳定通过。

发布上线

测试通过后，将测试代码合并到 master 分支，准备发布上线。



设置生产环境中的 .env 为：

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

执行部署命令：

```
scf deploy
```

至此，我们完成了一个 serverless-express 项目的开发和上线发布。

灰度发布

最近更新时间：2023-08-28 21:50:54

操作场景

在业务进行版本更新及切换时，为了保证线上业务稳定，建议采取灰度发布的方式。本文以已部署的 express 项目为例，为您介绍两种灰度发布的操作步骤。

前提条件

已完成 开发项目 。

操作步骤

1. 设置生产环境中的 .env：

```
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，切换所有流量到版本 N+1：

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

> **说明**
> 云函数组件支持了自定义别名的灰度发布，可以在任意两个函数版本间进行流量规则配置，详细说明请参考 Serverless 灰度发布 。

自动化部署

最近更新时间：2025-06-12 15:25:41

操作场景

在 Serverless 应用开发过程中，通常需要手动执行部署命令，将本地项目部署到云端。为提升开发效率，可以通过引入 CI（持续集成）能力，实现 Serverless 应用的自动化部署。本文向您介绍基于 GitHub 的自动化部署。

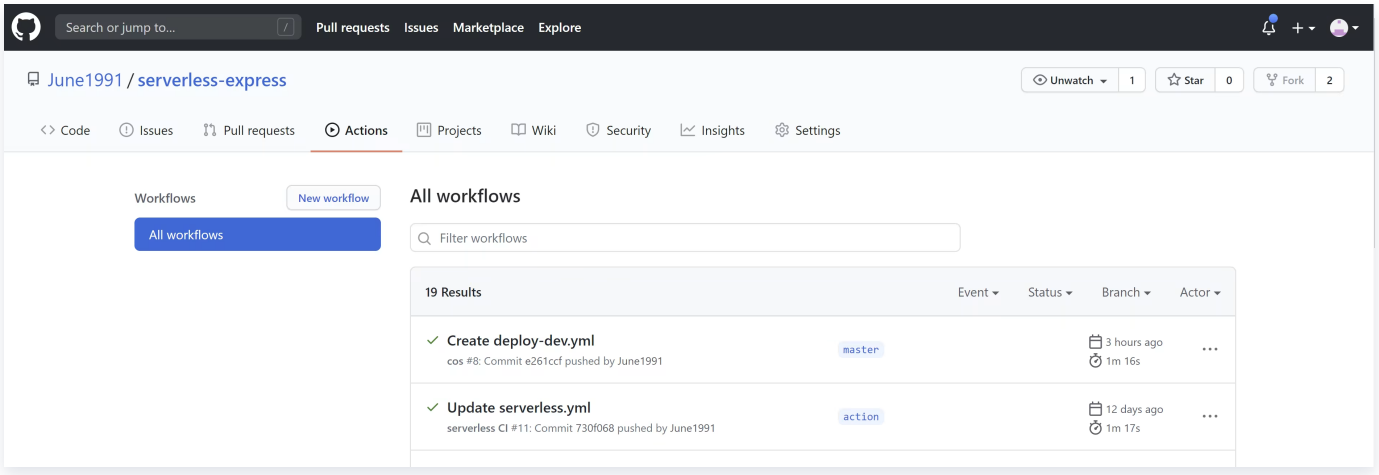
前提条件

- 已创建 Serverless 应用项目。参考 开发项目 创建您的 Serverless 项目并创建各个环境与分支。
- 已托管您的 Serverless 项目到 Github。

操作步骤

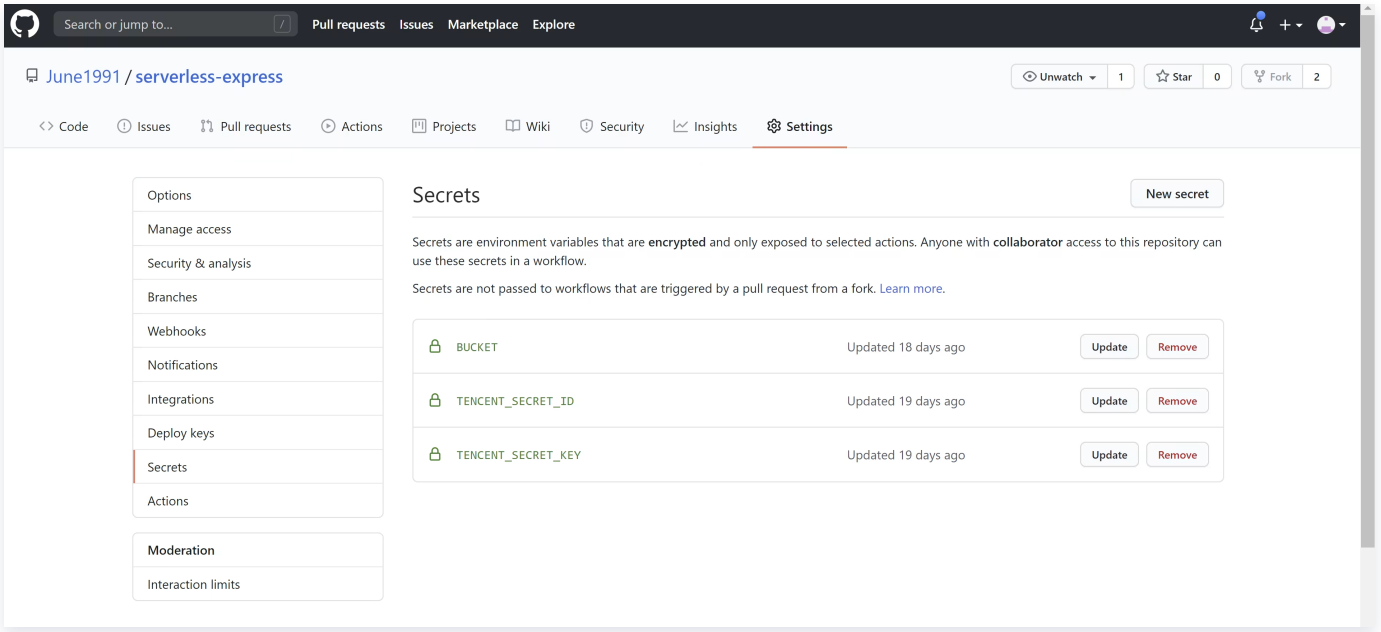
在开发测试阶段，为了方便开发、测试和调试，希望代码每次提交后进行自动化部署。操作如下：

1. 选取一个您需要执行自动化部署的分支（本示例选择 dev 分支）。
2. 在该分支下创建您的 action。



> **注意：**
> GitHub 规定如果事件发生在特定仓库分支上，则工作流程文件必须存在于该分支的仓库中 。

3. 配置腾讯云密钥。



4. 配置 action 部署步骤。

```
# 当代码推送到 dev 分支时，执行当前工作流程
# 更多配置信息：https://docs.github.com/cn/actions/getting-started-with-github-actions
name: deploy serverless
on: #监听的事件和分支配置
  push:
    branches:
      - dev
jobs:
  test: #配置单元测试
    name: test
    runs-on: ubuntu-latest
    steps:
      - name: unit test
        run: ''
  deploy:
    name: deploy serverless
    runs-on: ubuntu-latest
    needs: [test]
    steps:
      - name: clone local repository
        uses: actions/checkout@v2
```

```
      - name: install serverless
        run: npm install -g serverless-cloud-framework
      - name: install dependency
        run: npm install
      - name: build
        run: npm build
      - name: deploy serverless
        run: scf deploy --debug
        env: # 环境变量
          STAGE: dev #您的部署环境
          SERVERLESS_PLATFORM_VENDOR: tencent #serverless 境外默认为
aws，配置为腾讯
          TENCENT_SECRET_ID: ${{ secrets.TENCENT_SECRET_ID }} #您的腾
讯云账号 secret ID
          TENCENT_SECRET_KEY: ${{ secrets.TENCENT_SECRET_KEY }} #您的
腾讯云账号 secret key
```

完成上述配置后，开发者每次提交代码到 dev 分支时，就会自动部署。

部署 Stable Diffusion AI 绘画应用（自定义模型版）

最近更新时间：2025-07-24 10:10:52

应用介绍

该应用提供 stable-diffusion-webui 项目的 Serverless 化部署能力。应用创建成功后，您可以使用 stable diffusion webui 的全部能力，例如文生图、图生图，以及 Lora、ControlNet 等高阶能力。此外，还提供了管理自定义模型、插件的能力。应用在实际调用 GPU 进行图片生成、模型加载、插件安装等任务时计费，编写提示词、网页浏览图库图片等前端页面上的操作不计费。

应用资源

Stable Diffusion AI 绘画应用部署后，将为您创建以下资源：

- **云函数**：获取外部链接数据，通过全景录制实时进行录制。
- **CFS 文件存储（可选）**：Stable Diffusion 的自定义模型、插件等文件将会存储到 CFS 中会产生一定计费，详情可参见 CFS 计费规则 。
- **CLS 日志（可选）**：处理过程的日志会存储在日志服务 CLS 中，可能会产生一定计费，详情可参见 CLS 计费规则 。
- **API 网关**：通过 API 网关触发器进行图片生成任务的请求触发。

前提条件

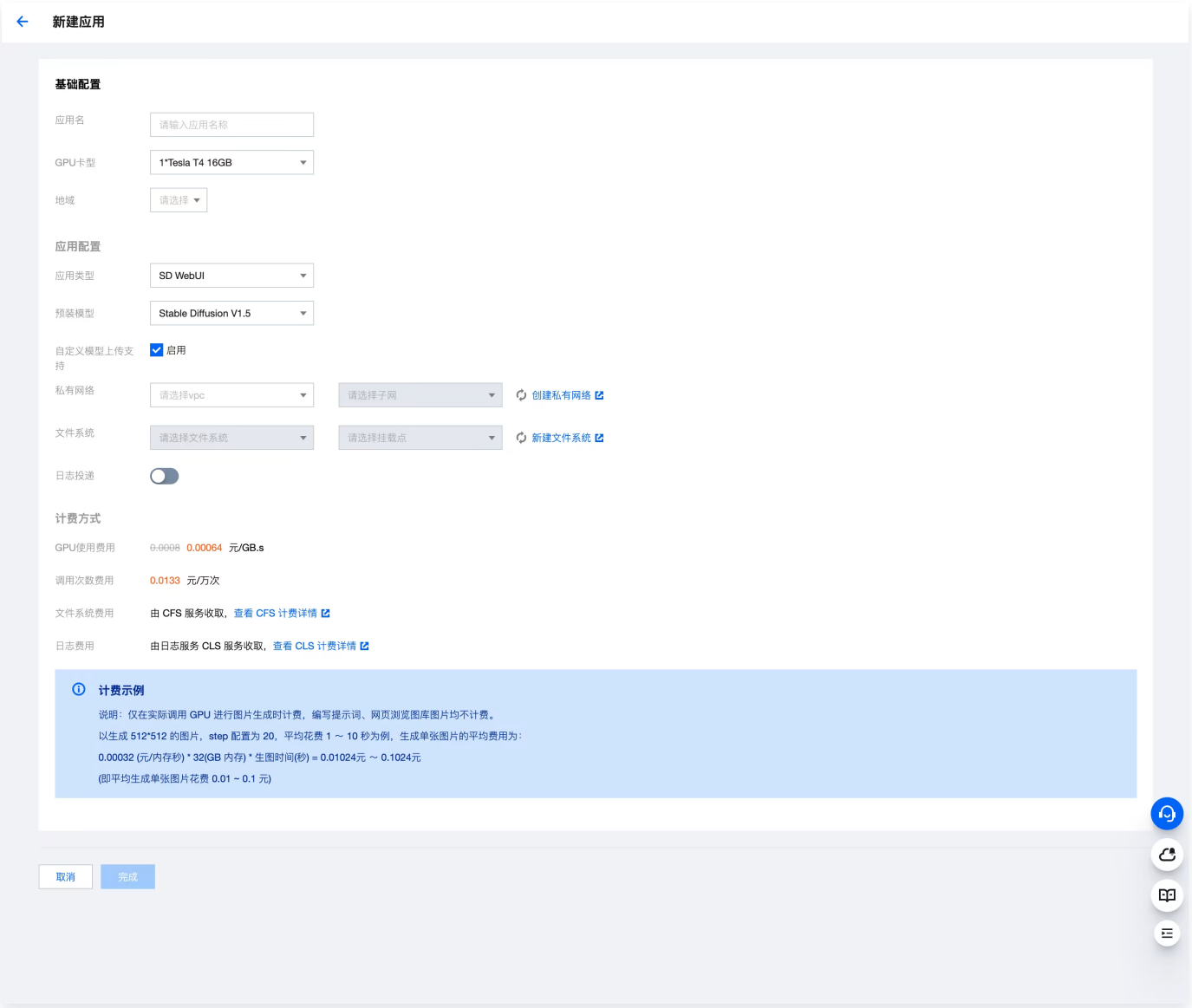
1. 配置部署账号权限。参考 账号和权限配置 。
2. 配置 运行角色 权限。
3. 全景录制需要和其他服务关联使用，需要添加以下策略：QcloudRedisFullAccess、QcloudVPCFullAccess、QcloudCFSFullAcces、QcloudSCFFullAccess、QcloudCOSFullAccess、QcloudAccessForScfRole 权限。详情请参见 策略绑定 。

操作步骤

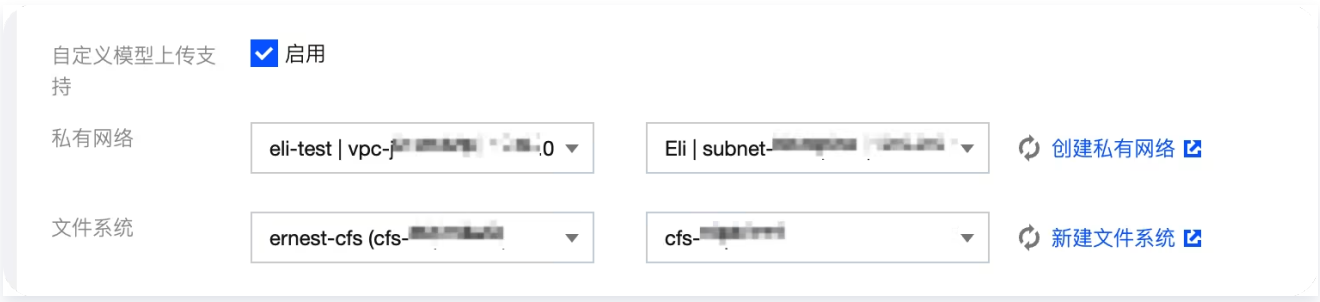
创建应用

1. 登录 Serverless 控制台，选择左侧导航栏中的 Serverless 应用 。
2. 在 **Serverless 应用**页面，单击**新建应用**。
3. 在**新建应用**页面，根据页面相关信息提示进行配置。
   - **创建方式**：选择**应用市场**。
   - **模糊搜索**：输入“AI”进行搜索，选择 Stable Diffusion AI 绘画自定义模型版。如下图所示：

4. 单击**下一步**，根据页面相关信息提示进行配置。如下图所示：



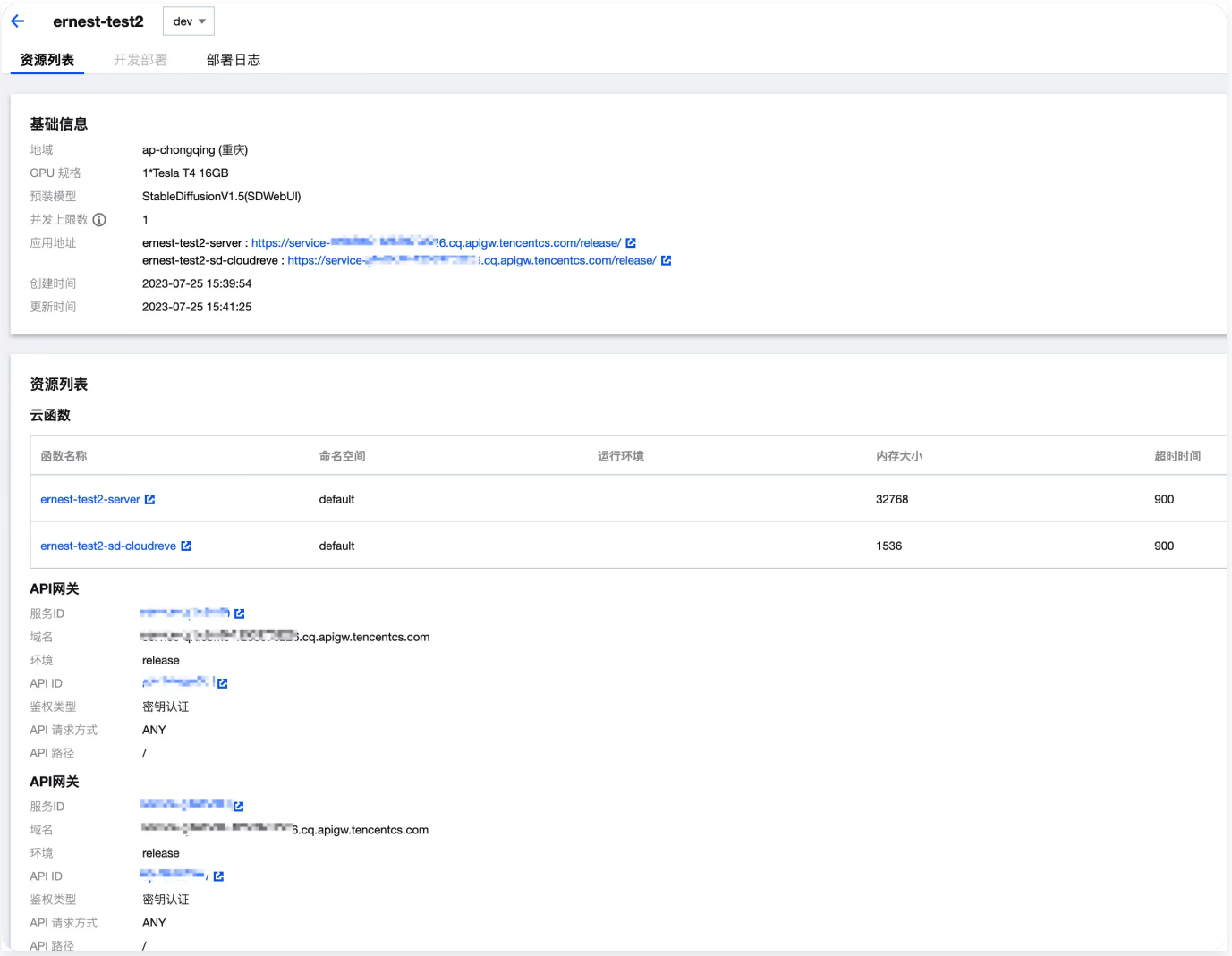
- **应用名**：例如 “sd-test”。
- **地域**：例如 “北京”。
- **应用类型**：SD WebUI 或 SD API，前者提供可视化界面，后者提供 API 以便于集成到业务系统中。
- **自定义模型上传支持**：启用后，需进行私有网络 VPC 和文件系统 CFS 的配置，具体创建流程请参见 VPC 创建 、CFS 创建 。如下图所示：



- **日志投递**：启用后将对 Stable Diffusion 运行过程中产生的日志投递到日志服务 CLS 中，以便排障。

5. 单击**完成**即可完成应用创建、函数创建以及 API 网关触发器创建。

如需根据业务场景修改函数配置，可通过 **Serverless 应用 > 资源列表 > 函数详情**进行修改。如下图所示：



使用应用

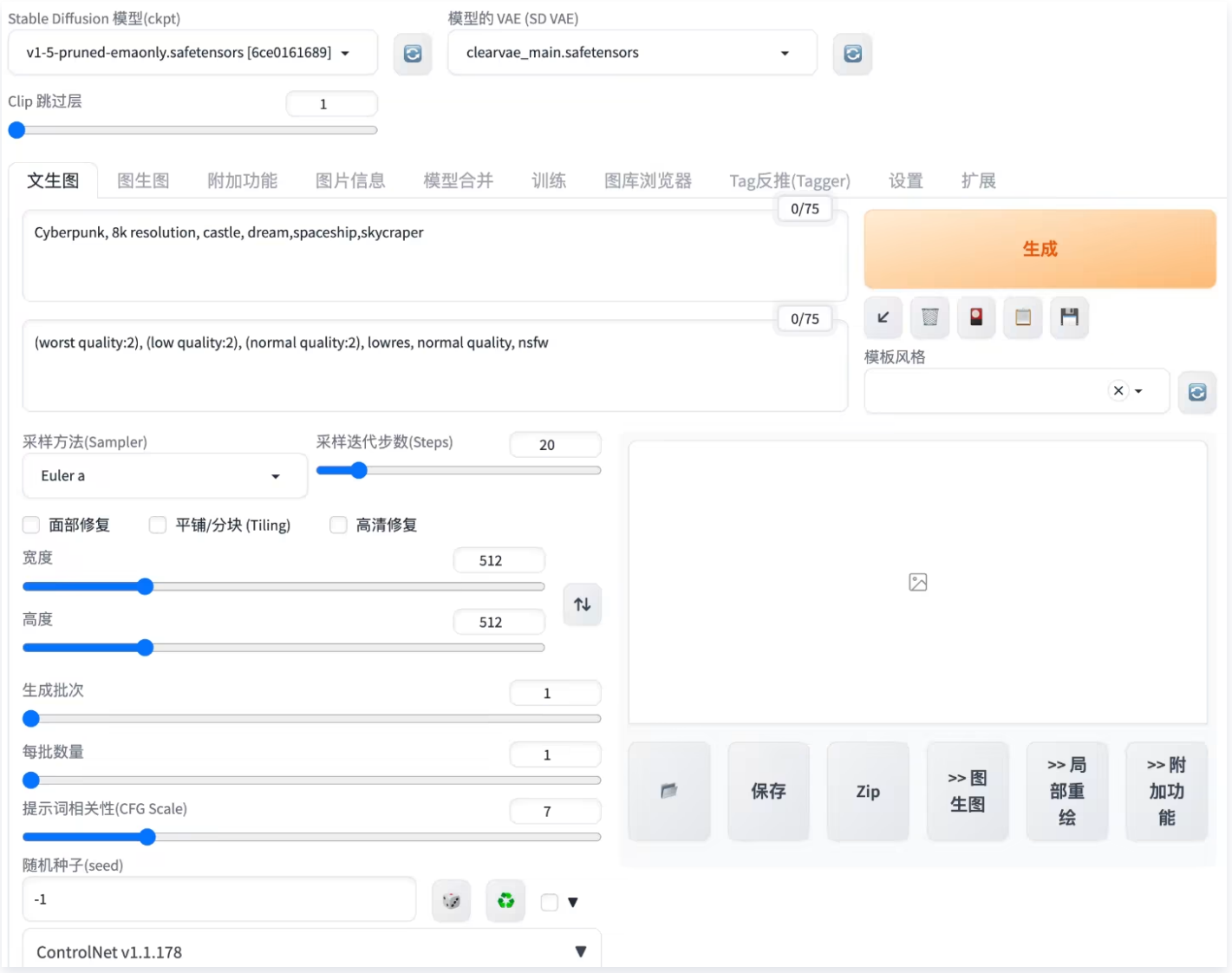
在创建成功的应用详情页面中，**基础信息 > 应用地址**展示两个 URL，分别是：

- stable diffusion WebUI 的访问地址，以 -server 结尾；如果创建时选择的是 API，则该地址为 API 的访问地址。
- stable diffusion 自定义模型的管理地址，以 -cloudreve 结尾。

如下图所示：

Stable Diffusion webui 的使用

访问上述提供的 WebUI URL，即可打开 WebUI 的页面开始使用，如下图所示：



Stable Diffusion API 的使用

访问上述提供的 API URL，在 URL 后边添加 docs 路径，即可打开 API 文档页面，如下图所示：

使用指南详情见 文档 。

Stable Diffusion API 的 URL 认证与安全配置

Stable Diffusion 应用创建后，会通过 API 网关服务，自动创建一个开通了公网访问权限的 API 服务，方便开箱即用。为了保护您的 API，避免恶意访问、未授权访问、应用漏洞、黑客攻击等导致的数据损失、资产损失，API 网关提供了多种 API 认证方式和 API 防护策略。目前 API 网关主要有 应用认证 、OAuth2.0认证 、EIAM 认证（历史功能，建议使用应用认证）三种方式。如业务上对 API 有鉴权访问的需求，可通过 API 网关控制台对该 API 进行鉴权配置，详细请参见 API 网关-认证与安全 。

Stable Diffusion API 集成数据万象内容审核服务提高安全性

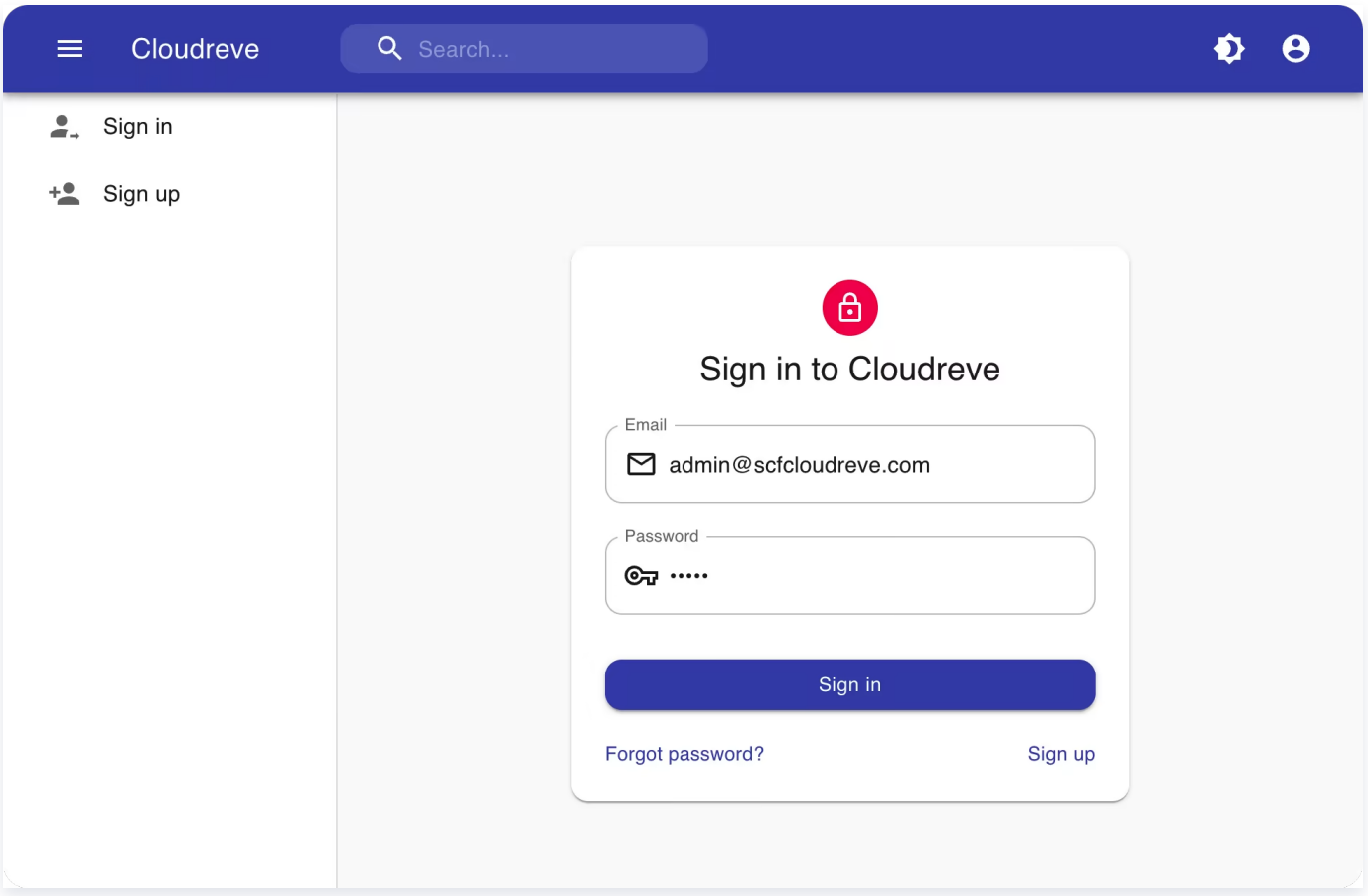
在业务中使用 Stable Diffusion API 生成图片，由于生成内容具有一定的不可控性，存在涉黄、违法违规等安全风险。建议通过集成内容审核服务，降低风险。详细的集成使用指南请参见 Stable Diffusion AI 绘画审核 。

上传并管理 Stable Diffusion 自定义模型

访问上述提供的自定义模型管理 URL，打开管理页面，需要管理员账号密码进行登录：

- 用户名：admin@scfcloudreve.com
- 密码：admin

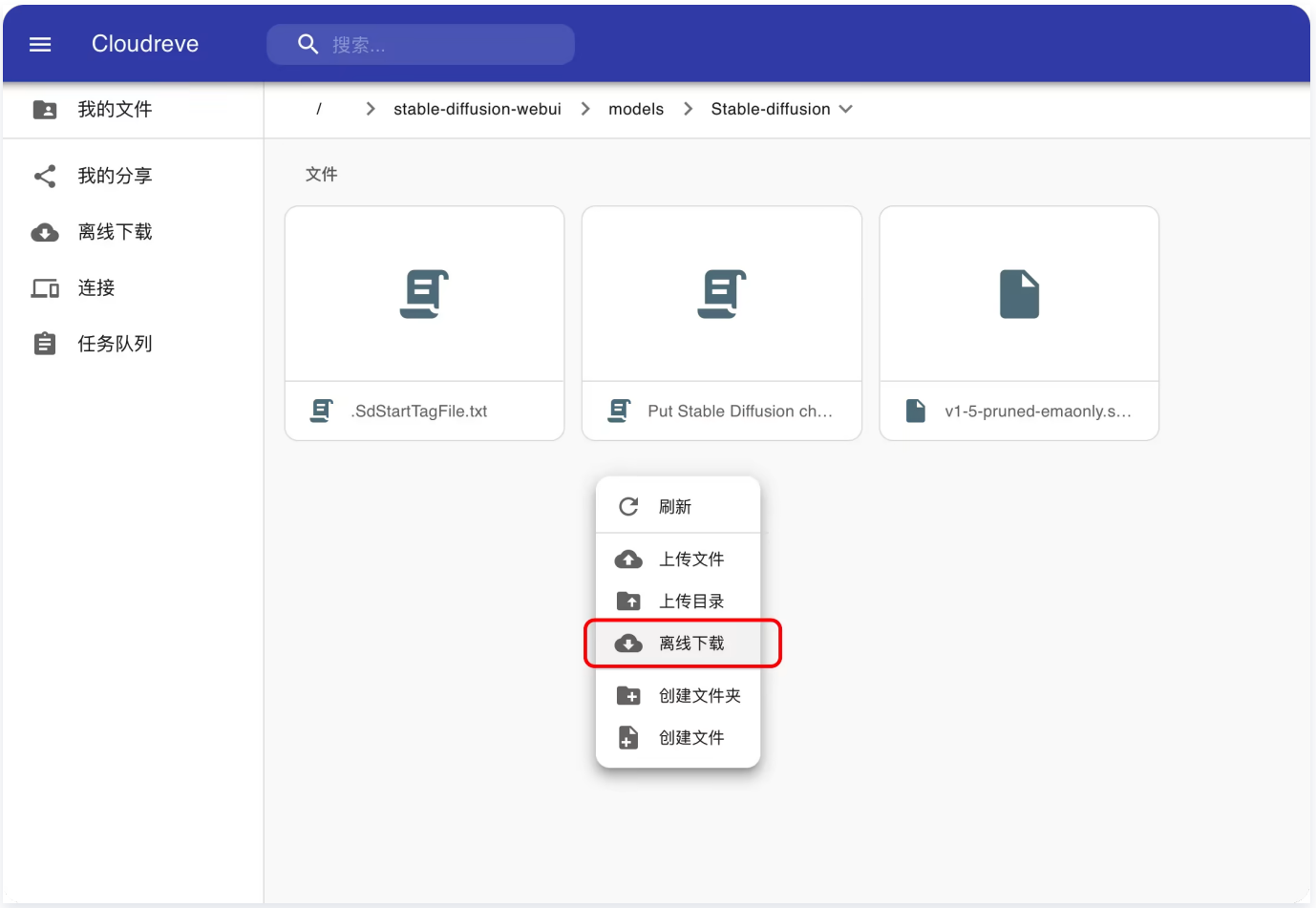
登录成功后，可在设置页面自行更新密码。



管理页面使用的是 cloudreve 开源网盘系统，实际操作的文件放在应用创建时指定的 CFS 文件系统中。

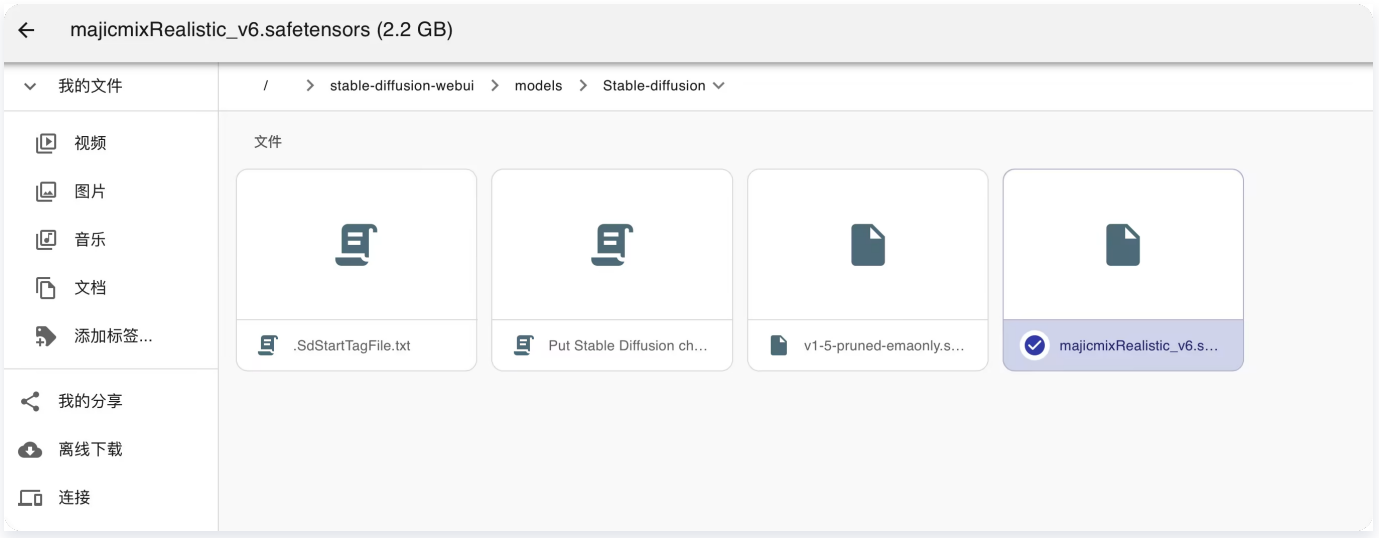
由于模型文件体积较大，受限于网关的限制，当前仅支持“离线下载/remote download”方式提交自定义模型，操作步骤如下：

1. 在文件页面，进入 "/stable-diffusion-webui/models/Stable-diffusion/" 目录。
2. 右键菜单中选择**离线下载/remote download**。如下图所示：

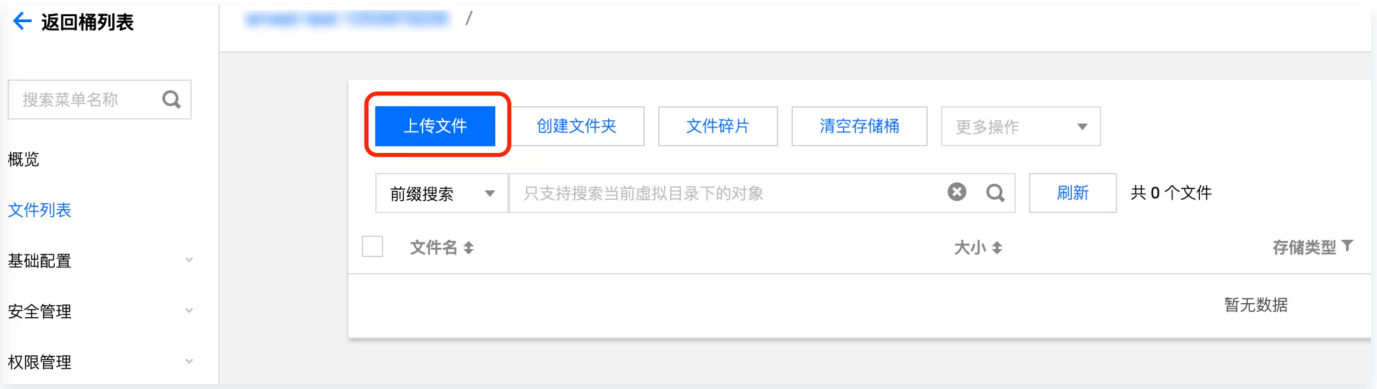


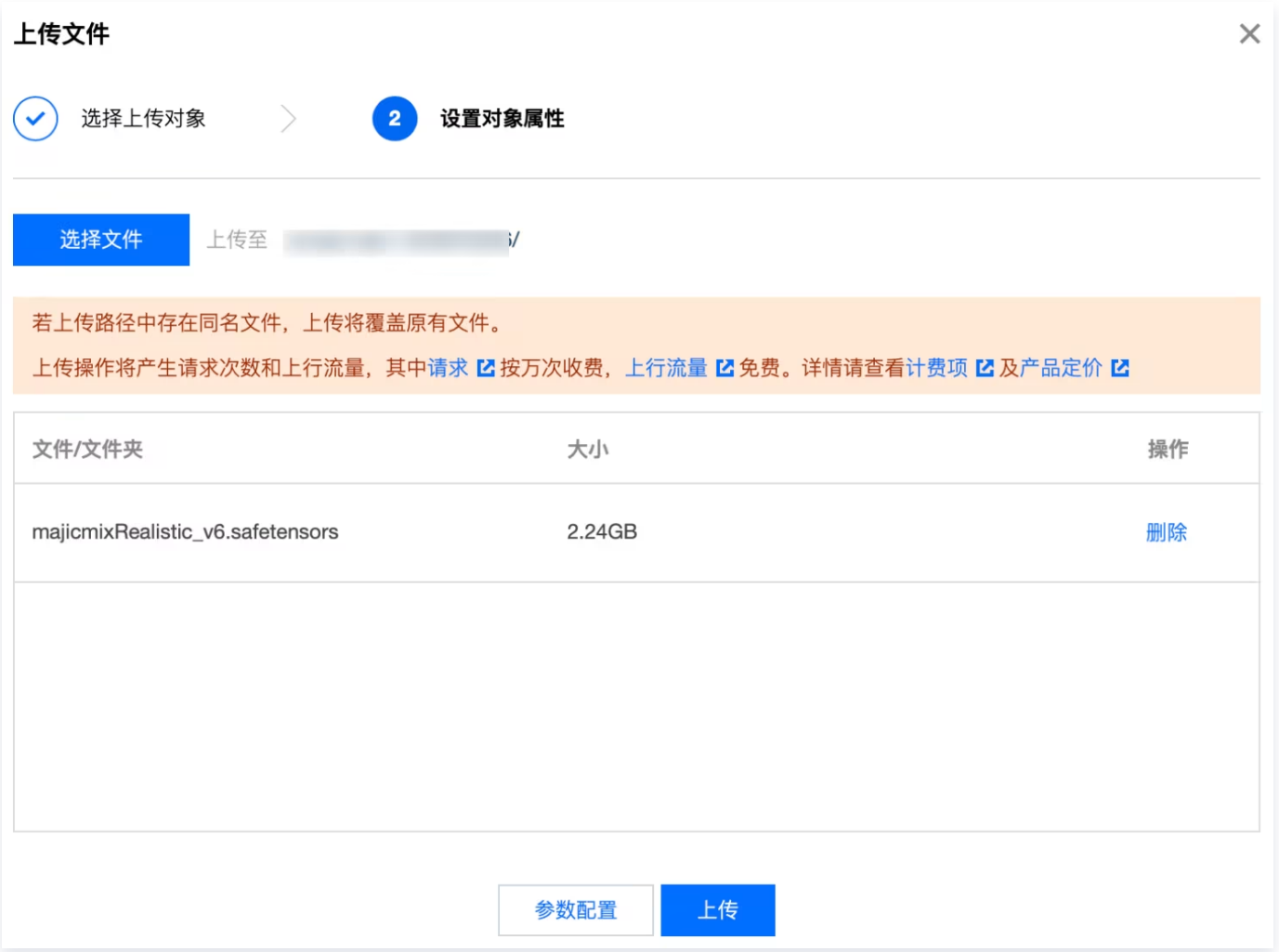
3. 在**离线下载**页面，填写可访问的模型文件下载链接（如应用创建在北京、广州等国内地域，则应提供境内可访问的链接，否则会导致下载失败），单击**创建任务**。如下图所示：

4. 进入离线下载页面，稍等片刻，等待模型文件下载完成后，进入 Stable Diffusion WebUI，在 Stable Diffusion 模型（ckpt）中选择新模型使用。如下图所示：

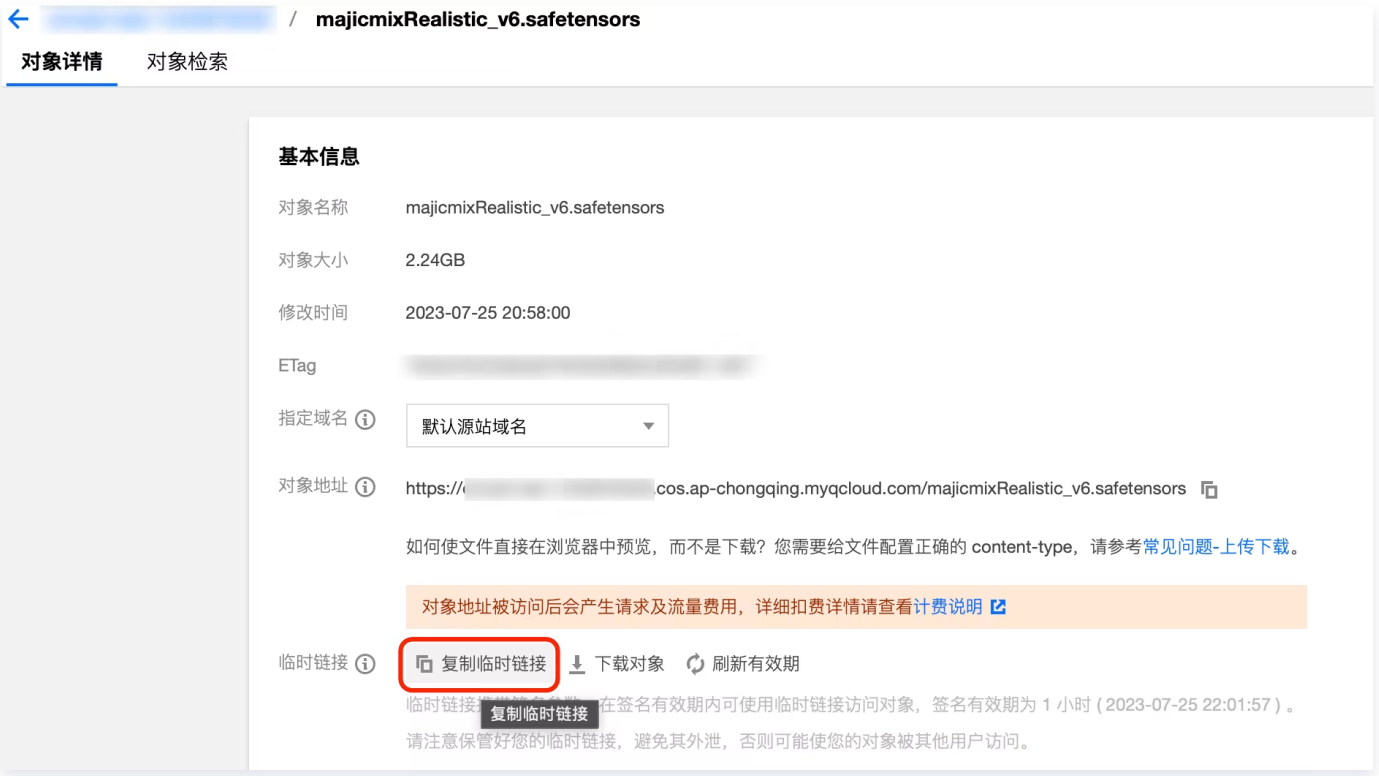


5. 如果您在本地有自定义模型想要直接上传，目前可以通过先将模型文件上传到 COS 对象存储的存储桶中，在对象详情页面，单击**获取临时链接**复制临时下载链接，然后在管理页面使用离线下载到您的应用中。如下图所示：


返回桶列表
搜索菜单名称
概览
文件列表
基础配置
安全管理
权限管理
上传文件
创建文件夹
文件碎片
清空存储桶
更多操作
前缀搜索
只支持搜索当前虚拟目录下的对象
刷新
共 0 个文件
文件名
大小
存储类型
暂无数据


上传文件
选择上传对象
2
设置对象属性
选择文件
上传至
若上传路径中存在同名文件，上传将覆盖原有文件。
上传操作将产生请求次数和上行流量，其中请求 按万次收费，上行流量 免费。详情请查看计费项 及产品定价
文件/文件夹
大小
操作
majicmixRealistic_v6.safetensors
2.24GB
删除
参数配置
上传


任务已完成（总共 1 个，成功 1 个，失败 0 个）
刷新或关闭浏览器将取消当前所有任务，并清除任务记录。若您有进行中的任务，请勿刷新或关闭浏览器。
上传文件
创建文件夹
文件碎片
清空存储桶
更多操作
在线编辑器
前缀搜索
只支持搜索当前虚拟目录下的对象
刷新
共 1 个文件
每页 100 个对象
文件名
大小
存储类型
修改时间
操作
majicmixRealistic_v6.safetensors
2.24GB
标准存储
2023-07-25 20:58:00
详情
预览
下载
更多



常见问题

计费说明

计费类型

Stable Diffusion AI 绘画应用的 GPU 使用费用仅支持按量计费（后付费）模式。套餐包、资源包、以及免费额度里均不包含 GPU使用费用。

定价

- GPU 使用费用 0.00021694元/GBs（内存*秒）。
- 调用次数费用 0.0133元/万次。
- 预置并发闲置费用 0.000045元/GBs。
- 文件系统费用由 CFS 服务收取，详情见 CFS 计费详情 。
- 日志费用由日志服务 CLS 服务收取，详情见 CLS 计费详情 。

计费示例

在实际调用 GPU 进行图片生成、模型加载、插件安装等任务时计费，编写提示词、网页浏览图库图片等前端页面上的操作不计费。
以第一次打开 webui，加载自定义模型（平均耗时20秒~100秒）后生成一张512大小的图片（10s）为例，共计30秒~110秒，该过程的费用为：0.00021694 (元/内存秒) * 32(GB 内存) * 30~110生图时间(秒) = 0.2元~0.7元。模型越大，该过程费用越多。

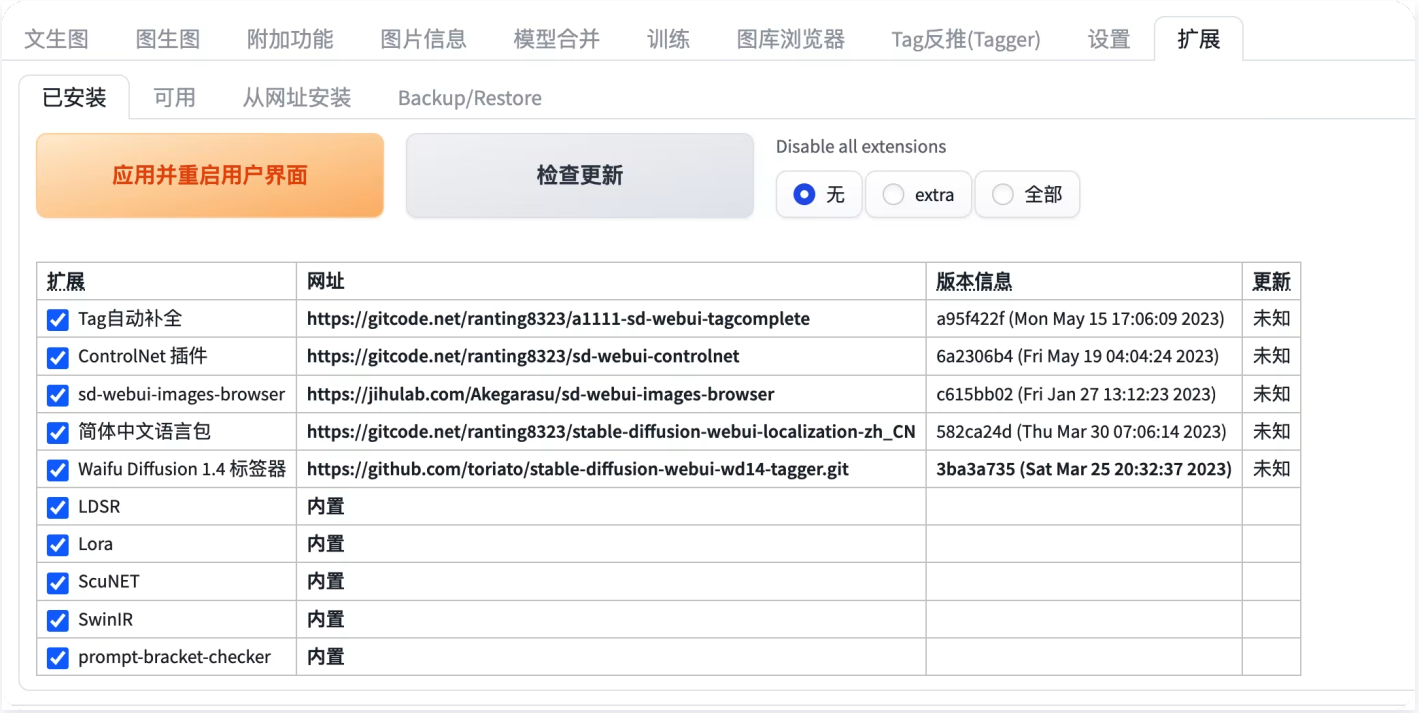
模型加载完后的平均生图的费用，以生成 512*512 的图片，step 配置为20，平均花费5~10秒为例，生成单张图片的平均费用为：0.00021694 (元/内存秒) * 32(GB 内存) * 5~10 生图时间(秒) = 0.03元~0.07元（即大量生图的平均生成单张图片花费 0.05~0.1元，该费用会随着图片分辨率、steps、controlnet 等插件步骤的不同而增加，和生成图片的时间成正比。）

并发说明

默认情况下，同一账户在同一地域下仅允许创建两个 AI 绘画应用，单个应用支持最大并发一个生成图片任务。如果您需要更大的并发能力，请 提交工单 联系我们。

Stable Diffusion WebUI 插件安装

在 WebUI 的扩展/extensions 页面来管理和安装 Stable Diffusion 插件，如下图所示：



扩展	网址	版本信息	更新
Tag自动补全	https://gitcode.net/ranting8323/a1111-sd-webui-tagcomplete	a95f422f (Mon May 15 17:06:09 2023)	未知
ControlNet 插件	https://gitcode.net/ranting8323/sd-webui-controlnet	6a2306b4 (Fri May 19 04:04:24 2023)	未知
sd-webui-images-browser	https://jihulab.com/Akegarasu/sd-webui-images-browser	c615bb02 (Fri Jan 27 13:12:23 2023)	未知
简体中文语言包	https://gitcode.net/ranting8323/stable-diffusion-webui-localization-zh_CN	582ca24d (Thu Mar 30 07:06:14 2023)	未知
Waifu Diffusion 1.4 标签器	https://github.com/toriato/stable-diffusion-webui-wd14-tagger.git	3ba3a735 (Sat Mar 25 20:32:37 2023)	未知
LDSR	内置		
Lora	内置		
ScuNET	内置		
SwinIR	内置		
prompt-bracket-checker	内置		

插件安装的网络问题

为了解决插件安装时可能遇到的网络问题，您可以尝试以下三种方式：

- 使用国内的插件托管地址：由于国内访问 GitHub 可能存在网络不稳定的问题，您可以尝试使用国内的插件托管地址，如 gitcode、coding 等。
- 使用代理加速：如果您仍然希望使用 GitHub 的插件地址，但遇到网络问题，可以考虑给 GitHub 插件地址加上 proxy 代理加速。
- 前往新加坡地域创建应用。

部署静态网站

最近更新时间：2025-10-29 14:47:52

操作场景

腾讯云 Website 静态网站组件通过使用 Tencent Serverless Cloud Framework，基于云上 Serverless 服务（如对象存储等），实现“0”配置，便捷开发，极速部署您的静态网站，Website 静态网站组件支持丰富的配置扩展，如自定义域名和 CDN 加速等。提供了目前最易用、低成本并且弹性伸缩的静态站点开发和托管能力。

特性介绍：

- **按需付费**：按照请求的使用量进行收费，没有请求时无需付费。
- **“0”配置**：只需要关心项目代码，之后部署即可，Serverless Cloud Framework 会搞定所有配置。
- **极速部署**：仅需几秒，部署您的静态网站。
- **实时日志**：通过实时日志的输出查看业务状态，便于直接在云端开发应用。
- **便捷协作**：通过云端的状态信息和部署日志，方便进行多人协作开发。
- **CDN 加速，SSL 证书配置和自定义域名**：支持配置 CDN 加速，支持自定义域名及 HTTPS 访问。

操作步骤

1. 安装

通过 npm 安装最新版本的 Serverless Cloud Framework，详情见 安装 Serverless Cloud Framework 。

2. 创建

创建并进入一个全新目录：

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

通过如下命令和模板链接，快速创建一个静态网站托管应用：

```
$ scf init  website-starter
$ cd website-starter
```

下载完毕后，目录结构如下所示：

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

在 `src` 目录中既可以托管简单的 html 文件，也可以托管完整的 React/Vue 的应用。

3. 部署

在 `serverless.yml` 文件下的目录中运行如下命令进行静态网站的部署。部署完毕后，您可以在命令行的输出中查看到您静态网站的 URL 地址，点击地址即可访问网站托管的链接。

```
$ scf deploy
```

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。
如果希望查看更多部署过程的信息，可以通过 `scf deploy --debug` 命令查看部署过程中的实时日志信息，`scf` 是 `serverless` 命令的缩写。

4. 配置

静态网站组件支持“0”配置部署，也就是可以直接通过配置文件中的默认值进行部署。但您依然可以修改更多可选配置来进一步开发该静态网站项目。
以下是静态网站 Website 组件的 `serverless.yml` 部分配置说明：

```
# serverless.yml

component: website # (必填) 引用 component 的名称，当前用到的是 tencent-website 组件
name: websiteDemo # (必填) 该 website 组件创建的实例名称

app: website-starter-xxx # (可选) 该 website 应用名称
stage: dev # (可选) 用于区分环境信息，默认值是 dev

inputs:
  src:
    src: ./src  # 部署项目的目录路径
    # dist: ./dist # build 完成后输出目录，如果配置 hook， 此参数必填
    # hook: npm run build # hook 脚本
    index: index.html
    websitePath: ./
  region: ap-guangzhou
  bucketName: my-bucket
  protocol: http
  hosts:
    - host: abc.com
      https:
```

```
        switch: on
        http2: on
        certInfo:
          certId: 'abc'
```

查看 全量配置及配置说明 >>
当您根据该配置文件更新配置字段后，再次运行 `scf deploy` 或者 `serverless` 就可以更新配置到云端。

5. 开发调试

部署了静态网站应用后，可以通过开发调试能力对该项目进行二次开发，从而开发一个生产应用。在本地修改和更新代码后，不需要每次都运行 `scf deploy` 命令来反复部署。您可以直接通过 `scf dev` 命令对本地代码的改动进行检测和自动上传。
可以通过在 `serverless.yml` 文件所在的目录下运行 `scf dev` 命令开启开发调试能力。
`scf dev` 同时支持实时输出云端日志，每次部署完毕后，对项目进行访问，即可在命令行中实时输出调用日志，便于查看业务情况和排障。

6. 查看状态

在 `serverless.yml` 文件所在的目录下，通过如下命令查看部署状态：

```
$ scf info
```

7. 移除

在 serverless.yml 文件所在的目录下，通过以下命令移除部署的静态网站 Website 服务。移除该应用时，只删除云函数相关的配置、代码。关联的其他云资源（如 COS、CLS 等），平台均不会关联删除，您可以前往对应产品控制台删除，避免不必要的计费。

```
$ scf remove
```

和部署类似，支持通过 `scf remove --debug` 命令查看移除过程中的实时日志信息（ `scf` 是 `serverless-cloud-framework` 命令的缩写）。

账号配置

当前默认支持 CLI 扫描二维码登录，如您希望配置持久的环境变量/密钥信息，也可以本地创建 `.env` 文件：

```
$ touch .env # 腾讯云的配置信息
```

在 `.env` 文件中配置腾讯云的 SecretId 和 SecretKey 信息并保存：

说明：

- 如果没有腾讯云账号，请先 注册新账号 。
- 如果已有腾讯云账号，可以在 API 密钥管理 中获取 SecretId 和 SecretKey。

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

部署 Hexo 博客

最近更新时间：2024-02-29 14:26:41

操作场景

该任务指导您通过 Serverless Website 组件，快速构建一个 Serverless Hexo 站点。

前提条件

- 已安装 Node.js 。（**2020年9月1日起，Serverless 组件不再支持 Node.js10.0 以下版本，请注意升级**）
- 已安装 Git 。

如您未安装上述应用程序，可以参见 Hexo 安装说明 。

操作步骤

1. 安装

1. 通过 npm 安装最新版本的 Serverless Cloud Framework，详情请参见 安装 Serverless Cloud Framework 。
2. 通过 npm 安装 Hexo：

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

3. 安装 Hexo 完成后，请执行下列命令，Hexo 将会在指定文件夹中新建所需要的文件。

```
$ hexo init hexo    # 生成 Hexo 目录
$ cd hexo
$ npm install
```

新建完成后，指定文件夹的目录如下：

```
.
├── _config.yml
├── package.json
├── scaffolds
├── source
|   ├── _drafts
|   └── _posts
└── themes
```

4. 安装完成后，可以通过 `hexo g` 命令生成静态页面：

```
$ hexo g    # generate
```

> **说明：**
> 如果希望在本地查看效果，也可以运行下列命令，通过浏览器访问 `localhost:4000` 查看页面效果。
>
> ```
> $ hexo s    # server
> ```

2. 配置

在 `hexo` 目录下，创建 `serverless.yml` 文件：

```
$ touch serverless.yml
```

在 `serverless.yml` 文件中进行如下配置：

```
# serverless.yml

component: website # (必填) 引用 component 的名称，当前用到的是 tencent-website 组件
name: hexodemo # (必填) 该 website 组件创建的实例名称

app: websiteApp # (可选) 该 website 应用名称
stage: dev # (可选) 用于区分环境信息，默认值是 dev

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

```
```

配置完成后，文件目录如下：

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

3. 部署

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

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

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


25s › hexodemo › Success
```

访问命令行输出的 Website URL，即可查看您的 Serverless Hexo 站点。

> ⚠ **注意：**
> 如果希望更新 Hexo 站点中的文章，需要在本地重新运行 `hexo g` 进行生成静态页面，再运行 `serverless` 更新到页面。

4. 移除

通过以下命令移除 Hexo 网站：

> **注意：**
> 在 serverless.yml 文件所在的目录下，通过以下命令移除部署的静态网站 Website 服务。移除该应用时，只删除云函数相关的配置、代码。关联的其他云资源（如 COS、CLS 等），平台均不会关联删除，您可以前往对应产品控制台删除，避免不必要的计费。

```
$ scf remove --debug

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

  6s » myWebsite » done
```

账号配置（可选）

当前默认支持 CLI 扫描二维码登录，如您希望配置持久的环境变量/密钥信息，也可以本地创建 `.env` 文件：

```
$ touch .env # 腾讯云的配置信息
```

在 `.env` 文件中配置腾讯云的 SecretId 和 SecretKey 信息并保存：

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

> **说明：**
> - 如果没有腾讯云账号，请先 注册新账号 。
> - 如果已有腾讯云账号，可以在 API 密钥管理 中获取 SecretId 和 SecretKey。

部署融合媒体转码应用

最近更新时间：2023-08-29 15:25:42

应用场景

适配多种终端设备

利用综合转码能力，适配 PC、TV 以及移动终端等多平台播放，带来更好的视觉体验。

支持多种网络环境

不同网络带宽的用户可通过智能融合媒体转码能力选择最佳码率，流畅播放。

视频快速审核

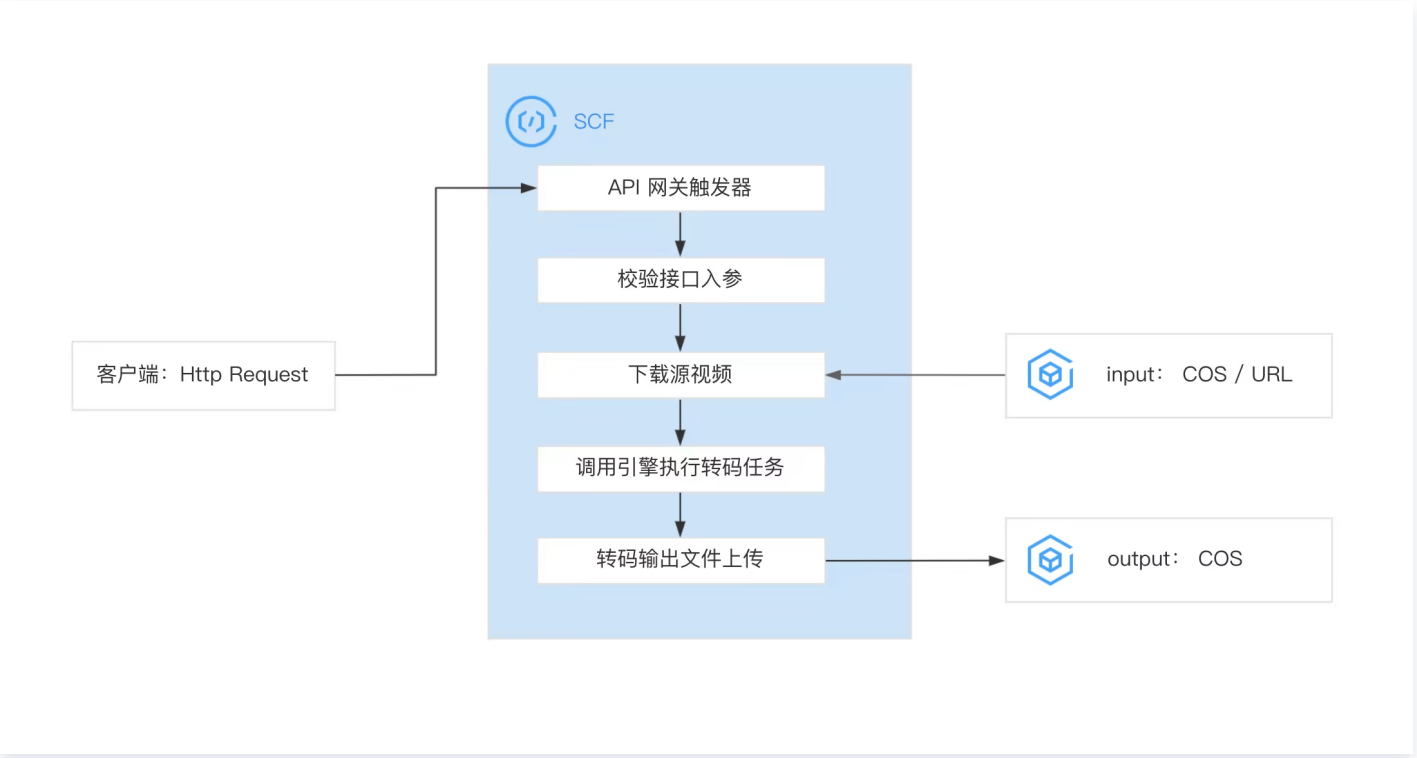
视频在生产、发布的各个环节都有送审、评审需求，在保证相同画质质量的前提下，可通过提高视频压缩率、调整视频码率等方式降低带宽消耗，压缩成本，提升效率。

提升短视频用户体验

支持 K 歌等场景下的画质增强，综合利用多种视频加清技术进行处理，提升画质，进一步提升歌曲点唱率。

架构原理

腾讯多媒体实验室自研的智慧融合媒体处理技术，支持对包括视频、音乐、文字、图片等融合媒体内容进行理解、分析、处理、编创。可满足广电、文旅、教育、电商、政企、泛互等各行业场景的融合媒体处理需求。
通过 Serverless 应用和腾讯多媒体实验室智慧融合媒体（Intelligent Convergence Media）深度整合，您可快速部署一个基于 COS + API + SCF + ICM 的开箱即用、灵活便捷的高效视频转码应用。通过 API 网关进行事件触发，将用户原始视频进行转码处理，转码完成后上传到用户对应的 COS 平台进行存储。具体流程如下图所示：



应用优势

- **存储多样化**
  视频源文件支持 URL 和 COS 两种类型，满足客户源视频存在于不同位置的存储需求，避免资源冗余和浪费。
- **开箱即用**
  提供转码应用，开箱即用，无需用户进行代码开发，零运维，弹性伸缩、按需付费。
- **算力可配置**
  可根据视频大小，选择不同规格的函数计算资源，可显示支持多核、大内存、高 IO 等定制化需求。
- **高压缩率**
  依托腾讯多媒体融合编解码引擎，可使同等画质下视频压缩率较传统压缩率平均提升50%。

应用资源

转码应用部署后，将为您创建以下资源：

- **云函数**：读取 COS 文件，使用腾讯多媒体融合编解码引擎转码后流式输出回 COS 中，并将转码过程的实时日志输出到 CLS。
- **API 网关**：通过 API 网关触发器进行事件触发。
- **CLS 日志**：存储转码过程的实时日志。CLS 日志可能会产生一定计费，详情请参见 CLS 计费规则 。

前提条件

1. 配置部署账号权限。参考 账号和权限配置 。
2. 配置 运行角色 权限。

操作步骤

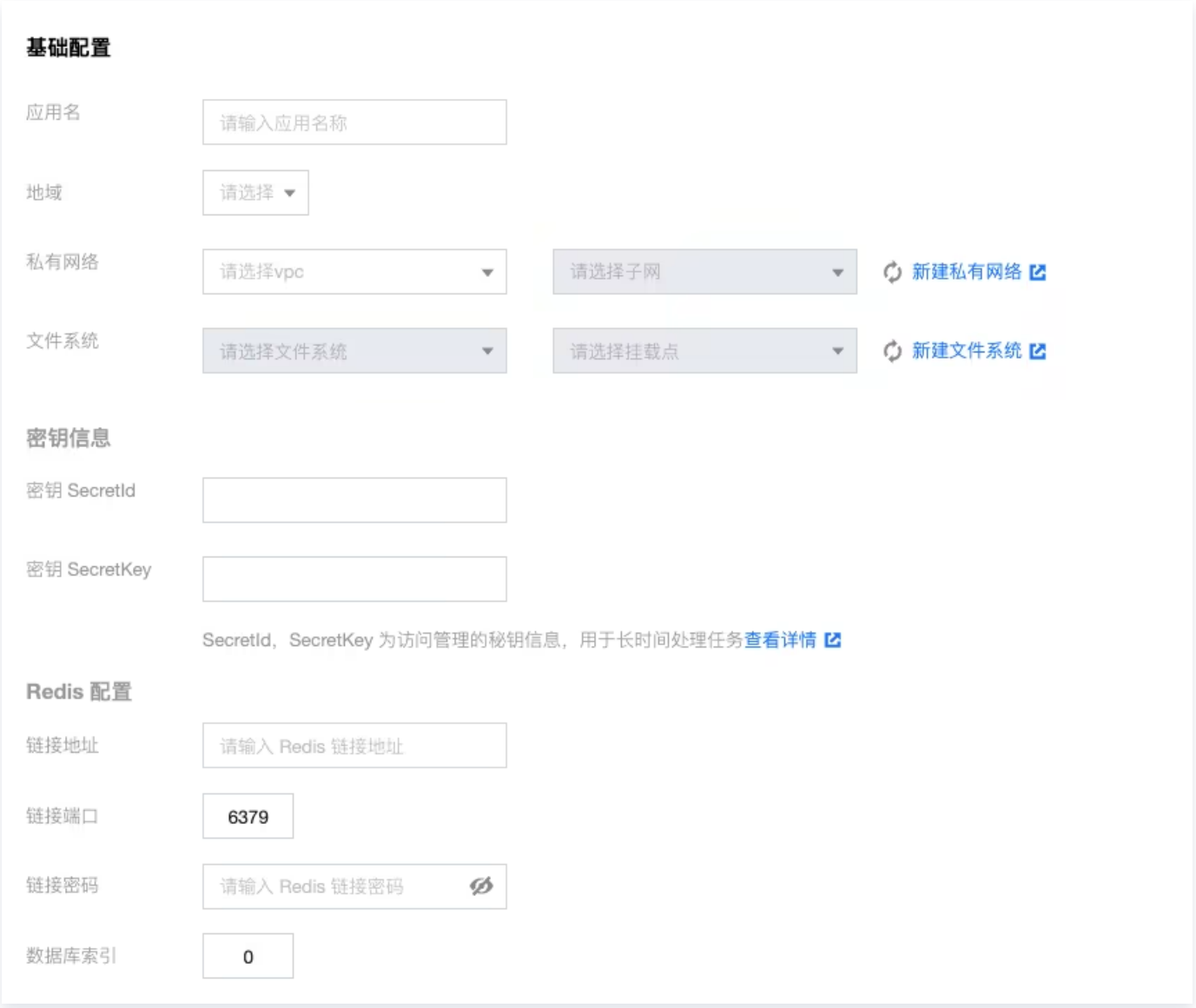
创建依赖资源

创建的过程中，需要依赖以下相关组件，请提前创建。具体创建流程请参见 VPC 创建 、CFS 创建 、COS 创建 、Redis 创建 、SecretId/SecretKey 创建 。

创建融合媒体转码应用

1. 登录 Serverless 控制台，选择左侧导航栏中的 Serverless 应用 。
2. 在 “Serverless 应用” 页面，单击**新建应用**。
3. 在新建应用页面，根据页面相关信息提示进行配置。如下图所示：

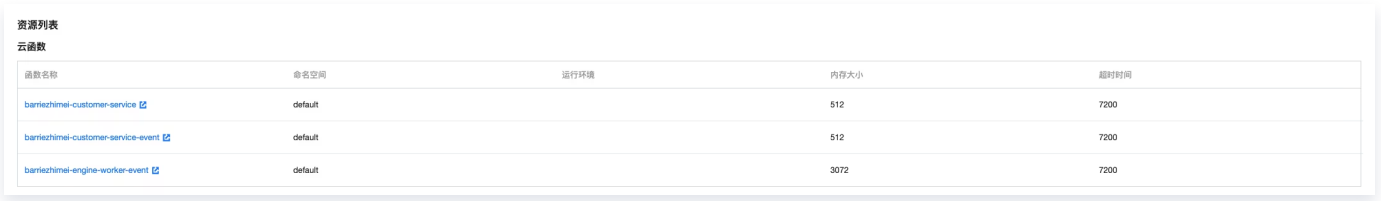
   - **创建方式**：选择**应用市场**。
   - **模糊搜索**：输入“转码”进行搜索，选择**快速部署一个融合媒体转码应用**。
     单击模板中的**查看详情**，即可在弹出的“模板详情”窗口中查看相关信息，支持下载操作。
4. 单击**下一步**，根据页面相关信息提示进行配置。如下图所示：



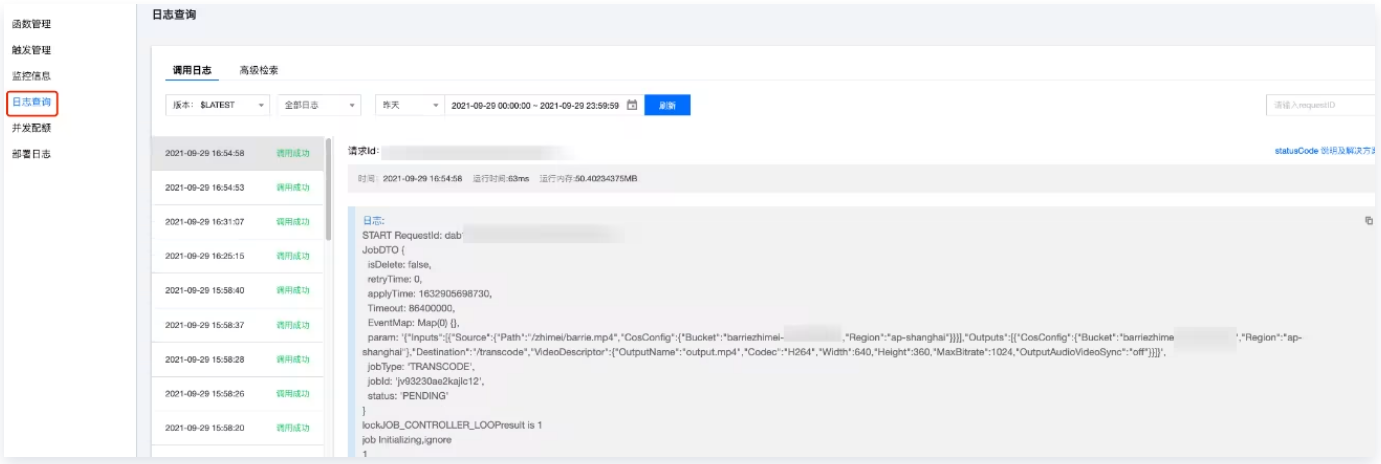
- **应用名**：例如，“transcode-app”。
- **地域**：例如：“成都”。
- **私有网络**：按需选择。
- **文件系统**：按需选择。
- **密钥 SecretId**：按需选择。
- **密钥 SecreKey**：按需选择。
- **链接地址**：按需选择。
- **链接密码**：按需选择。
- **数据库索引**：按需选择。

5. 单击**完成**即可完成应用创建、函数创建以及 API 网关触发器创建。
   如需根据业务场景修改函数配置，您可以在 **Serverless 应用 > 应用名称 > 资源列表** 中进行修改。如下图所

示：



6. 在“函数详情”页面中选择**日志查询**，可以查看到打印出的转码日志信息。如下图所示：



7. 进入 COS 控制台，查询转码结果 。

> **说明**
>
> - 转码应用可能需要依赖云函数长时间运行能力，详情请参见 异步执行 。
> - 融合媒体处理后台冷启动有一定时间，为避免请求超时，API 网关触发器的后端超时建议设置较大数值。

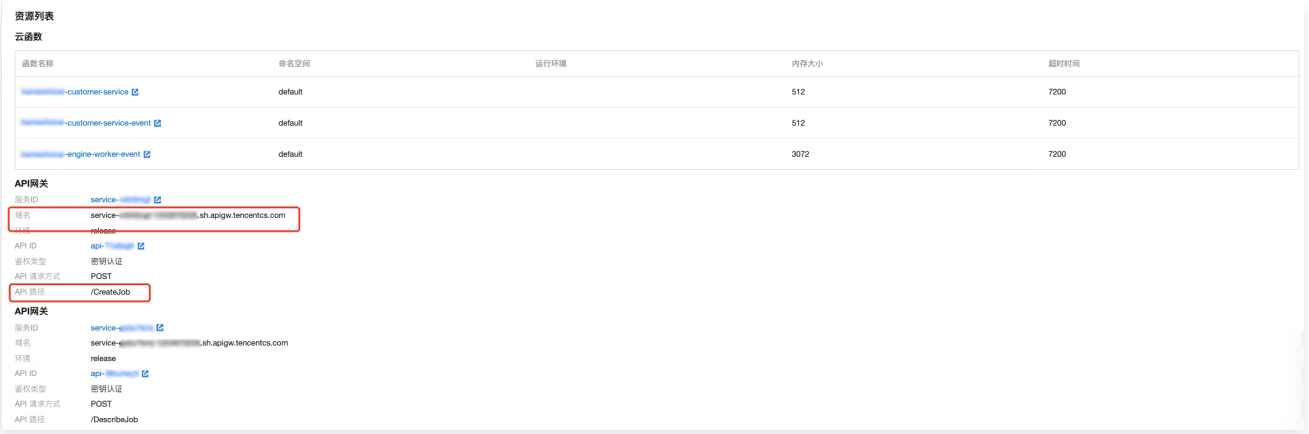
接口使用说明

接口请求路径

请求URL

1. 登录 Serverless 控制台，选择左侧导航栏中的 Serverless 应用 。
2. 在 Serverless 应用列表页，单击应用名称。

3. 在应用详情页，获取触发器访问入口。如下图所示：



请求方式

POST
Content-Type: application/json

接口鉴权方式

融合转码服务接口由云函数实现，采用了 API 网关触发器方式对外提供服务，鉴权采用应用认证方式（ApiAppKey 和 ApiAppSecret），详情请参见 应用鉴权 。

创建融合媒体转码任务接口

通过此接口可以发起转码任务，在接口参数中指定视频输入源、输出配置以及CFS等参数。

请求参数说明

参数名	类型	必选	描述
Action	String	是	本产品固定 CreateTransCodeJob
CreateTransCodeJobRequest	Object	是	包含 Inputs、Outputs
Inputs	Input[]	是	转码文件的获取来源，目前应用仅处理数组的第一项
Outputs	Output[]	是	转码文件的输出配置，目前应用仅处理数组的第一项

参数详情如下：

Input

参数名	类型	必选	描述
Url	String	否	待转码视频的 Url 地址，支持视频格式（mp4，avi，mkv，mov）
Source	Object	是	视频转码配置，用于转码

> **说明**
> Url 和 Source 必填一个，同时都有，优先使用 Url 。

Source 说明：

参数名	类型	必选	描述
Path	String	否	COS 路径，支持视频格式（mp4，avi，mkv，mov），例如 /video/1.mp4
CosConfig	CosConfig	是	COS 存储配置，用于存储转码后的文件

Output

参数名	类型	必选	描述
Destination	String	否	COS 路径（不含文件名），例如/video
VideoDescriptor	VideoDescriptor	是	视频转码配置，用于转码
CosConfig	CosConfig	是	COS 存储配置，用于存储转码后的文件

VideoDescriptor

参数名	类型	必选	描述
OutputName	String	是	输出文件名，后缀代表封装格式（mp4，avi，mkv，mov），例如 test.mp4
Codec	String	是	输出编码格式（H264，H265，VP9）
Width	Int	否	输出视频宽，正整数，取值范围[0，7680]，默认0（输入视频宽度），必须为偶数
Height	Int	否	输出视频高，正整数，取值范围[0，7680]，默认0（输入视频高度），必须为偶数
MaxBitrate	Int	否	输出视频最大码率，范围[10，200000]，单位Kbps，默认输入视频码率
OutputAudioVideoSync	String	是	是否进行音频和视频帧率同步，取值范围为 on off

CosConfig

参数名	类型	必选	描述
Bucket	String	是	COS 桶名称，例如 test-12345678
Region	String	是	COS 所在的地域，例如 ap-guangzhou

说明：

- 转码函数运行时需要读取 COS 资源进行转码，并将转码后的资源写回 COS。CosConfig 将使用模板创建时填入的 SecretId 和 SecretKey 信息，需要给已提供的SecretId 和 SecretKey配置一个授权 COS 全读写的运行角色。更多参考 函数运行角色 。
- 转码输出桶与应用建议配置在同一区域，因为跨区域配置转码应用稳定性及效率都会降低，并且会产生跨区流量费用。

返回参数说明

参数名	类型	描述

CreateTransCodeJobRequest	Object	异步调用固定返回格式，指示请求是否成功
job	Object	-
id	String	任务 id，可以通过任务查询接口查询任务当前状态
status	String	任务状态。PENDING 队列中；RUNNING 运行中；TERMINATED 已终止；SUCCESS 成功；FAIL 失败

请求示例

输入源为 Url

```
{
    "Action": "CreateTransCodeJob",
    "CreateTransCodeJobRequest": {
        "Inputs": [{
            "Url": "https://tencent"
        }],
        "Outputs": [{
            "CosConfig": {
                "Bucket": "media",
                "Region": "ap-guangzhou"
            },
            "Destination": "/transcode",
            "VideoDescriptor": {
                "OutputName": "output.mp4",
                "Codec": "H264" ,
                "Width":640,
                "Height" :360,
                "MaxBitrate": 1024,
                "OutputAudioVideoSync": "off"
            }
        }]
}
}
```

输入源为 COS

```
{
    "Action": "CreateTransCodeJob",
    "CreateTransCodeJobRequest": {
        "Inputs": [{
          "Source": {
            "Path":"/enhance.mp4",
            "CosConfig":{
              "Bucket": "media",
              "Region": "ap-guangzhou"
            }
          }
        }],
        "Outputs": [{
            "CosConfig": {
                "Bucket": "media",
                "Region": "ap-guangzhou"
            },
            "Destination": "/transcode",
            "VideoDescriptor": {
                "OutputName": "output.mp4",
                "Codec": "H264" ,
                "Width":640,
                "Height" :360,
                "MaxBitrate": 1024,
                "OutputAudioVideoSync": "off"
            }
        }]
}
}
```

响应示例

```
{
    "CreateJobResponse": {
```

```
        "job": {
            "id": "ecjsxdesrprc9f8235",
            "state": "PENDING"
        }
    }
}
```

查询融合媒体转码任务状态接口

此接口可以通过创建融合媒体任务得到的任务 id 来查询任务状态。

请求参数说明

参数名	类型	必选	描述
Action	String	是	本产品固定 DescribeJob
DescribeJobRequest	Object	是	包含 id
DescribeJobRequest	Input[]	是	转码文件的获取来源，目前应用仅处理数组的第一项
id	String	是	创建融合媒体任务得到的任务 id

响应说明

参数名	类型	必选	描述
RequestId	String	是	请求唯一 RequestId
DescribeJobResponse	Object	是	包含job 对象
job	Object	是	任务状态结构体
id	String	是	任务 id
state	String	是	任务状态。PENDING 队列中；RUNNING 运行中；TERMINATED 已终止；SUCCESS 成功；FAIL 失败
ApplyParam	String	是	已经提交的任务参数
applyTime	String	是	提交时间

startTime	String	是	开始时间
endTime	String	是	任务结束时间
EngineRequestId	String	是	引擎 RequestId，可以查询实际的引擎运行情况
ErrorMessage	String	是	若引擎运行失败，可以看到根据错误信息判断错误来源

请求示例

```
{
  "Action": "DescribeJob",
  "DescribeJobRequest": {
    "id": "ecjsxdesrprc9f8235"
  }
}
```

响应示例

```
{
  "RequestId": "f0mn4teo68pjvn4eujawwcpz52vsiplf",
  "DescribeJobResponse": {
    "job": {
      "id": "g2ads4bgms1wwqv244",
      "state": "SUCCESS",
      "ApplyParam": "{}",
      "applyTime": "1630929363223",
      "startTime": "1630932002511",
      "endTime": "1630965618361",
      "EngineRequestId": "f8344d37-10c1-4a33-95eb-920337dacb8a"
    }
  }
}
```

取消融合媒体转码任务状态接口

此接口可以通过创建融合媒体任务得到的任务 id 来取消转码任务，只有 PENDING 状态任务可取消。

请求参数说明

参数名	类型	必选	描述
Action	String	是	本产品固定 CancelJob
CancelJobRequest	Object	是	包含 id
id	String	是	创建融合媒体任务得到的任务 id

响应参数说明

参数名	类型	必选	描述
RequestId	String	是	请求唯一 id
Result	Object	是	包含 IsSuccess
IsSuccess	Boolean	是	取消任务是否成功，取值范围为 true false

请求示例

```
{
  "Action": "CancelJob",
  "CancelJobRequest": {
    "id": "9rv9zx15idweviq2"
  }
}
```

响应示例

```
{
  "RequestId": "yqsrrgh6t1097k6cqd8cpnw5y26xkone",
  "Result": {
    "IsSuccess": true
  }
}
```

中止融合媒体转码任务状态接口

此接口可以通过创建融合媒体任务得到的任务 id 来中止转码任务，只有 RUNNING 状态任务可中止，此操作为尝试中止，最终是否中止以结果为准。

请求参数说明

参数名	类型	必选	描述
Action	String	是	本产品固定 TerminateJob
TerminateJobRequest	Object	是	包含 id
id	String	是	创建融合媒体任务得到的任务 id

响应参数说明

参数名	类型	必选	描述
RequestId	String	是	请求唯一 id
Result	Object	是	包含 IsSuccess
IsSuccess	Boolean	是	中止任务是否成功，取值范围为 true false

请求示例

```
{
    "Action": "TerminateJob",
    "TerminateJobRequest": {
        "id": "m2y3isjca9ew9w54"
    }
}
```

响应示例

```
{
  "RequestId": "yqsrrgh6t1097k6cqd8cpnw5y26xkone",
  "Result": {
    "IsSuccess": true
  }
}
```

部署互动直播间语音识别服务

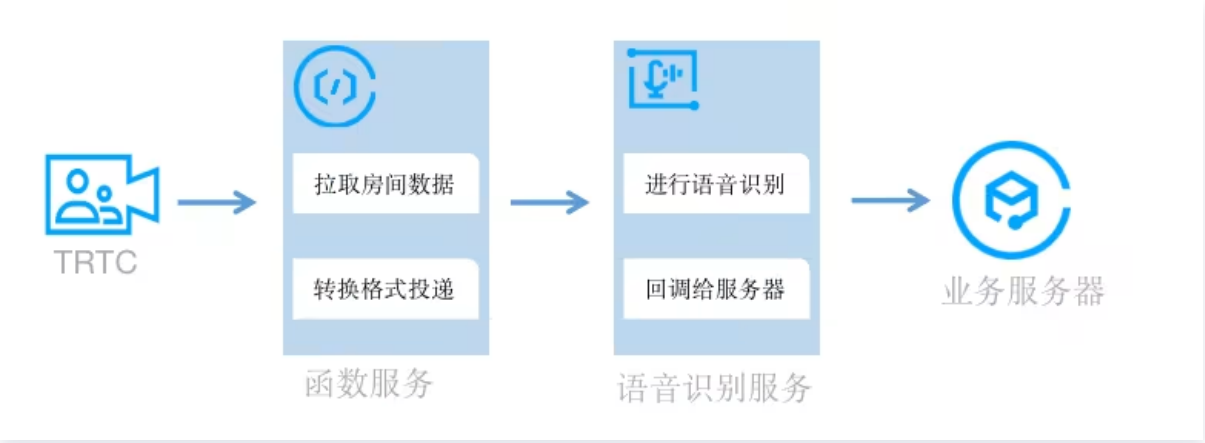
最近更新时间：2022-12-22 11:24:24

使用场景

- **你画我猜**：可以实时拉取房间内某个用户的音频进行实时识别，转换成文本之后回调给客户的业务服务器，进行业务逻辑判断
- **语音审核**：和业务关联比较多的语音审核，可以采取该接口将数据流投递到语音识别接口进行语音识别，然后进行关键词过滤。
- **实时字幕**：可以通过该接口实时识别房间音频数据，形成文本，在前端做呈现。

架构原理

具体流程如下图所示：



应用优势

- **实时返回**：可以将 trtc 房间的语音数据实时识别返回，快速高效。
- **流程简单**：深度融合 trtc 和 asr，数据流完全打通，不需要复杂接入流程。
- **使用灵活**：数据返回给业务服务器之后，可以和业务逻辑实时关联。

注意事项

- 一般情况下，语音识别的处理时间较长是由于部署函数时开启了 异步执行 。
- 目前识别的结果将下发至业务服务器。暂不支持 websocket 的形式，无法下发至客户端。
- 默认的鉴权形式是 应用鉴权 ，测试时可更改为 无鉴权 。

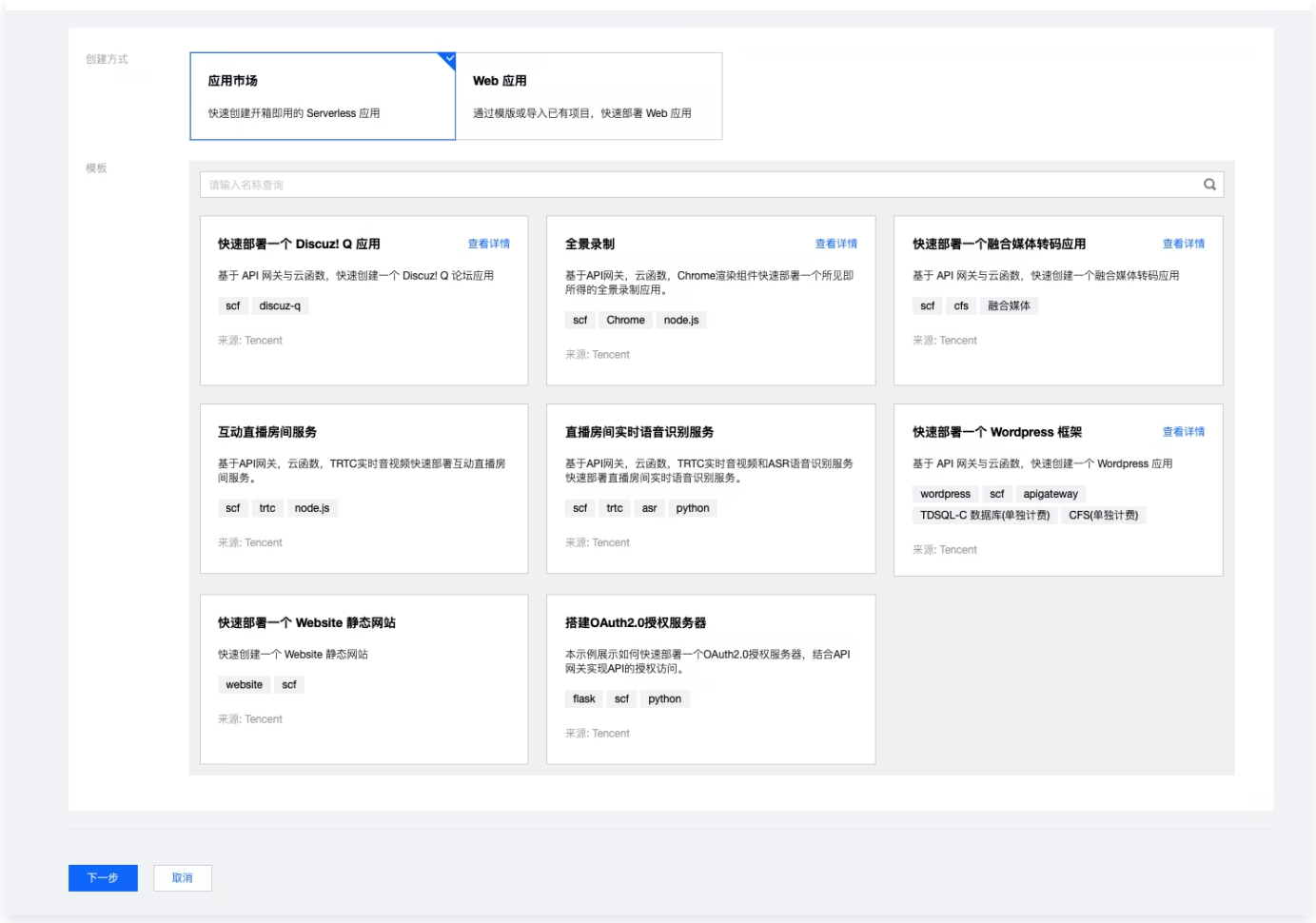
使用流程

1. 开通服务

您需要开启腾讯云语音识别服务，操作详情见 开通语音识别服务 。

2. 部署函数服务

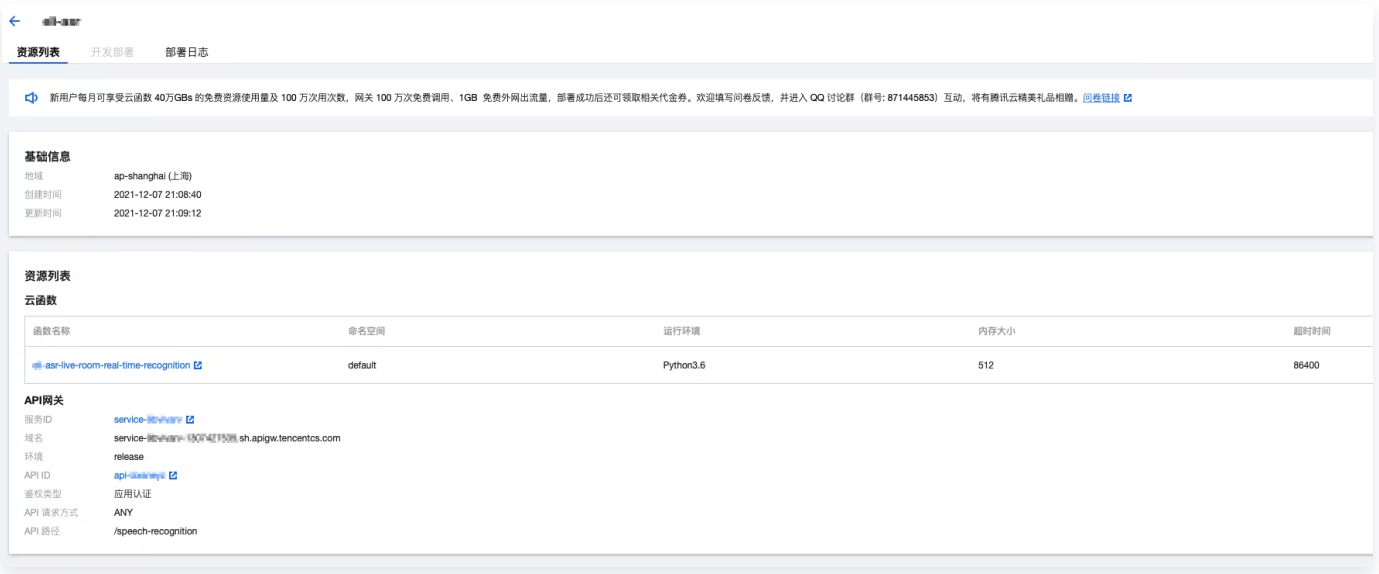
1. 登录 Serverless 应用控制台 。
2. 单击**新建应用**，进入“新建应用”页面。如下图所示：



3. 选择**直播房间实时语音识别服务**，并进行**基础配置**。如下图所示：

- **应用名**：自定义名称。
- **地域**：根据实际情况进行选择。
- **密钥信息**：您可在 API密钥管理 中查看腾讯云账号密钥信息。

4. 单击**完成**即可。已创建的应用详情界面如下图所示：



5. 单击**云函数** > **函数名称**进入函数详情页，在“触发管理”中获取访问路径。

3. 语音识别启动接口

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

请求参数：

参数	类型	必填	说明
SdkAppId	Int	是	应用 ID，用于区分不同 TRTC 应用。
RoomId	Int	否	整型房间号 ID，用于在一个 TRTC 应用中唯一标识一个房间。
StrRoomId	String	否	字符串房间号 ID，RoomId 与 StrRoomId 必须配置一项，如果 RoomId 与 StrRoomId 同时配置，使用 RoomId。
UserId	String	是	录制用户 ID，用于在一个 TRTC 应用中唯一标识一个用户。
UserSig	String	是	录制用户签名，用于对一个用户进行登录鉴权认证。
Callback	String	否	录制结束后的回调地址，并使用 POST 方式进行回调。

请求示例：

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

识别结果回调接口

回调参数说明：

参数	类型	必填	说明
SdkAppId	Int	是	应用 ID。
RoomId	int	是	整型房间 ID。
UserId	String	是	识别的用户 ID。
StrRoomId	String	是	字符串房间 ID。
Result	Array	是	语音识别结果 [{},{},{},{}]
Status	String	是	当前用户语言识别状态，normal/finished

Result 为数组类型，元素封装为 JSON 对象，封装格式如下：

参数名称	类型	必选	描述
Voice	String	是	当前一句话文本结果，编码为 UTF8。
Index	Integer	是	当前一句话结果在整个音频流中的序号，从 0 开始逐句递增。
StartTime	Integer	是	当前一句话结果在整个音频流中的起始时间。
EndTime	Integer	是	当前一句话结果在整个音频流中的结束时间。
Message	String	是	识别任务的执行结果。例如，识别结束，识别中，识别失败等。

结果示例：

```
{
    "RequestID": "95941e2c85898384a95b81c2a5******",
    "SdkAppId": 1400000000,
    "RoomId": 43474,
    "UserId": "user_55952145",
    "Status": "recognizing/finished",
    "Result": [{
        "Voice": "实时语音识别",
        "Index": 0,
        "StartTime": 0,
        "EndTime": 1024,
        "Message": "success"
    }]
}
```

部署互动直播房间服务

最近更新时间：2023-07-10 18:06:02

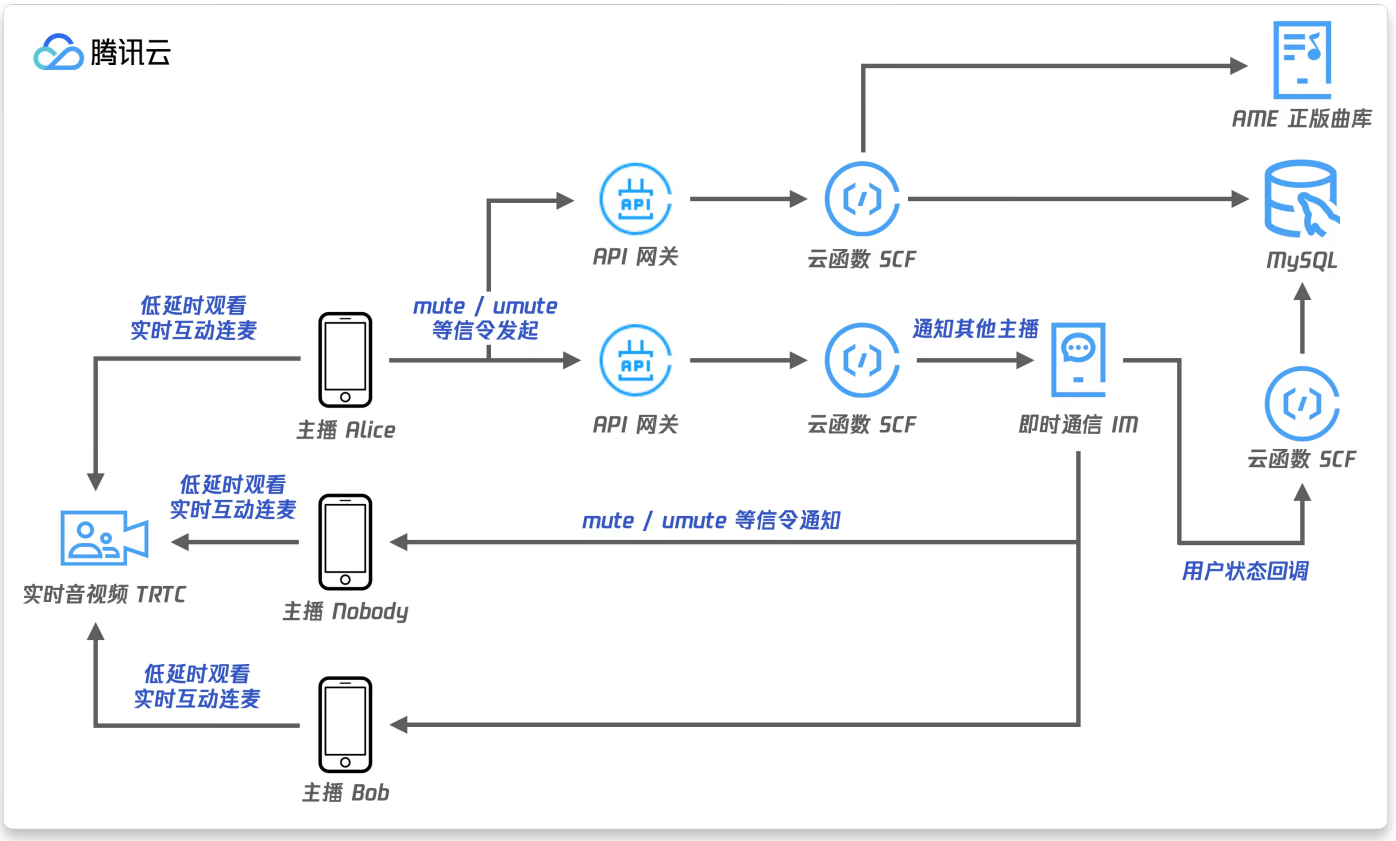
应用介绍

互动直播房间服务是一套业务房间后台服务，可以实现房间列表、房间成员列表、房间用户状态同步等功能。通过 Serverless 方式进行快速部署，配合实时音视频 TRTC、直播、IM 等音视频能力，可以快速搭建社交泛娱乐行业各种场景。

使用场景

腾讯云在 秀场直播/电商直播 提供端云一体的低代码解决方案，通过云函数模板 LiveRoom 创建业务后台、配合移动直播 SDK、即时通信 IM 低门槛快速实现登录、注册、开播、连麦一整套在线直播业务场景。

架构原理



应用优势

- **快速部署**：采取云函数模板一键部署，快速实现互动直播房间服务。
- **定制灵活**：根据需要选择对应功能，灵活方便。
- **使用方便**：通过 api 接口的形式直接调用，快速高效。

应用资源

- **TRTC**：提供音视频服务。
- **IM**：提供消息服务。
- **Redis**：存储用户状态。
- **数据库**：存储用户信息。
- **VPC**：保障网络通畅。

注意事项

- 所有资源要在同一个 VPC 内部，无需使用 VPC 的资源最好在同一个区域。
- 该服务是 aPaas 的后台服务，需要结合 aPaas 的客户端代码一起使用。

操作步骤

前提条件

您已 注册腾讯云账号，并完成 实名认证。

步骤1：新建相关云资源

1. **新建 TRTC 服务**
   登录 实时音视频控制台，新建一个 TRTC 应用，并记录 SDKAppID 信息。操作详情见 新建应用。

   > **说明**
   > TRTC 与 IM 采用同一个 SDKAPPID，在 TRTC 控制台新建应用后会同步在 IM 控制台新建一个应用并开通体验版。

2. **新建 VPC 服务**
   登录 私有网络控制台，新建私有网络。操作详情见 新建私有网络。
3. **新建数据库服务**
   本文的数据库服务以 Mysql 为例。登录 MySQL 购买页，根据实际需求选择各项配置后进行购买。操作详情见 创建 MySQL 实例。
4. **新建 Redis 服务**
   登录 Redis 控制台，新建 Redis。操作详情见 创建 Redis 实例。

   > **说明**
   > **通过命令行**创建互动直播房间服务时，默认创建 TD SQL。因此您无需新建数据库服务、Redis 服务和 VPC。

步骤2：进行权限授权

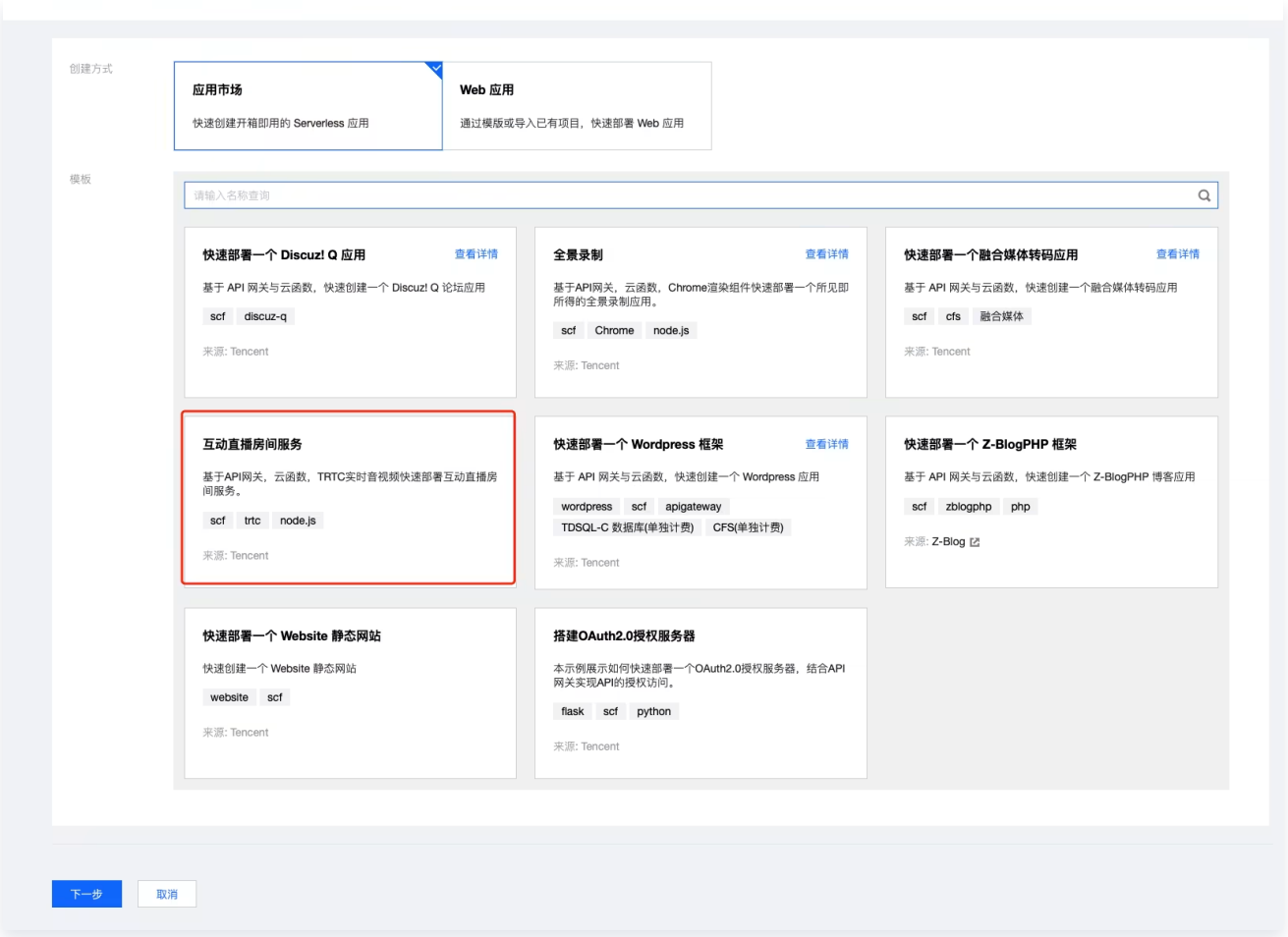
1. 配置部署账号权限。详情见 账号和权限配置 。
2. 配置 运行角色 权限，角色名称为 SCF_QcsRole。
3. 互动直播房间服务要和其他服务关联使用，需要添加以下策略：QcloudRedisFullAccess、QcloudVPCFullAccess、QcloudSCFFullAccess、QcloudAccessForScfRole 权限。详情请参见 策略绑定 。

步骤3：新建互动直播房间服务

1. 通过以下两种方式部署服务

方案1：通过控制台创建

1. 登录 Serverless 应用控制台 。
2. 单击新建应用 ，进入项目创建页面。
3. 选择**互动直播房间服务**，单击**下一步**。如下图所示：



4. 填写应用基本信息。如下图所示：

- **应用名**：新建的应用名称，这里可以自定义。
- **地域**：服务的区域，可以根据实际情况选择
- **私有网络**：VPC 名称。
- **SDKappid**：TRTC 的应用名称，可以在 TRTC 控制台获取。
- **密钥**：TRTC 的密钥，可以在跑通 demo 界面获取到。
- **数据库地址**：用来存储用户信息，这里一般使用 mysql，和 mysql 的地址保持一致即可。

- **数据库名称**：指数据库应用名，保持一致即可。
- **redis 地址**：redis 数据库的访问地址，该数据用来保存用户状态。

5. 单击**完成**，将为您自动部署应用。

方案2：通过命令行创建

通过命令行创建互动直播房间服务时，默认创建 TD SQL。
基于 腾讯云云函数 环境，小直播提供了一套部署简单、定制性更加灵活的**开源服务**，只需要四步即刻快速跑通“小直播”后台程序：

1. **下载源码：**

```
git clone https://github.com/TencentServerlessApps/live-room.git
```

> **说明**
> 您也可以直接进入我们的 GitHub 仓库 进行下载。

2. **部署准备：**
安装最新版本的 Serverless Cloud Framework，详情见 安装 Serverless Cloud Framework 。

```
npm i -g serverless-cloud-framework
npm install
```

> **说明**
> Node 安装方法请参见 Node.js，在 Windows 下请使用 Administrator 权限启动 `Node.js command prompt`，不支持 PowerShell。

3. **开始部署服务：**

3.1 在 `live-room` 目录下，创建环境变量文件 `.env`，并输入如下内容：

```
TRTC_TIM_APPID=xxxxxxxxxxxxxxx  // 此处即上一章节中的记录的SDKAppID;
TRTC_TIM_SECRET=xxxxxxxxxxxxxxx // 此处即上一章节中的记录的SecretKey
```

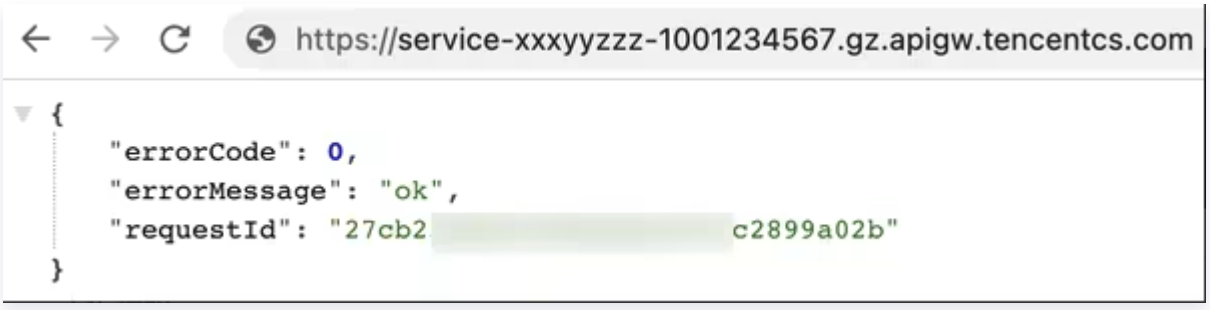
3.2 创建成功后，启动发布，需要扫码授权或配置 本地密钥授权：

```
npm install
sls deploy
```

> **说明**
> Windows 用户，请使用 Administrator 权限启动 `Node.js command prompt`，否则扫码认证会失败。

2. 检查服务是否部署成功

应用部署成功后，您可以查看项目的部署日志。
从发布日志中获取 API 网关地址，例如 `https://service-xxxyyzzz-1001234567.gz.apigw.tencentcs.com`，在浏览器中直接打开您的网关地址。如下图所示，即恭喜您，后台服务部署成功！



> **说明**
> 默认 api 接口是免鉴权的，如果对于安全性有较高的要求，可以在网关控制台更改鉴权模式，详情见 选择鉴权方式，建议选择应用鉴权，鉴权算法详情见 应用鉴权 。

步骤4：apaas 最佳实践

详情见结合互动直播房间 快速跑通小直播 。

常见问题

如何查看云函数的日志?

查看云函数的日志，请参见 日志检索教程 。

为何网关返回的是 `SystemError (99): Invalid TRTC config`？

请检查是否正确配置了 TRTC 的 SdkAppId (TRTC_TIM_APPID) 和 Secret (TRTC_TIM_SECRET)。

为何网关和函数无法访问?

请确认是否开通服务，请确认是否账户欠费。

为何 Windows 无法发布云函数?

请使用系统管理员 (Administrator) 启动 `Node.js command prompt`，请不要用 PowerShell。

如何确认网关新建成功?

在函数详情页中获取 API 网关访问路径，若能在浏览器访问，则网关正常。API 网关访问路径如下所示：

```
# 链接地址需要为已创建的接口地址
https://service-xxxyyzzz-1001234567.gz.apigw.tencentcs.com/helloworld
```

如何确认函数新建成功?

在函数详情页中获取函数访问路径，若能在浏览器访问，则网关正常。函数访问路径如下所示：

```
# 链接地址需要为已创建的接口地址
https://service-xxxyyzzz-1001234567.gz.apigw.tencentcs.com
```

接口说明

详情见 接口说明 。

Token 登录

Token 登录可以实现客户端一定时间免登录。客户端可以将 Token 存储在本地，下次使用 Token 登录，避免每次都需要输验证码。
*接口地址：/base/v1/auth_users/user_login_token***
请求参数：用户信息，例如：

```
{
  // 用户 (user) 信息
  "userId": "xxx-1000019",
  "token": "dd243f4.......c32742af3bf6cb",
  "apaasUserId": "xxx-1000019" // 多租户
}
```

特别说明

- apaasAppId：若登录到默认租户，可以不传。若非默认租户，必须设置为指定租户。
- userId 和 token 是用户的账号（手机号或邮箱等）信息，后台会返回 apaasUserId。
- token 有效期是 30 天。

返回值：错误码和鉴权信息

- 若数据操作失败，返回 ServiceDBFailed (103)。
- 若 Token 已过期，返回 UserTokenExpired (203)。
- 若 Token 不正确，返回 UserTokenInvalid (204)。
- 返回新的 token 和 expire 过期 UTC 时间。

以下信息给 SDK 鉴权使用。

```
{
  "errorCode": 0,
  "errorMessage": "login done",
  "data": {
    "userId": "xxx-1000019",
    "sdkAppId": 1400544182,
    "sdkUserSig": "eJwtjMEKwjA.............bKC9S",
    "token": "9d0dk28r31........c327id192k9f",
    "expire": "2021-08-22 06:42:54",
    "phone": "+8615800001111",
    "email": "xxx@xxx.com",
    "name": "",  // 新用户用户名为空字符串
    "avatar": "https://xxx",
    "apaasAppId": "xxx", // 多租户
    "apaasUserId": "xxx-1000019" // 多租户
  }
}
```

具体错误码详情见 错误码 。

注销登录

客户端请求注销登录，会在数据库清除 session 和 token 等信息，但用户信息会保留。

*接口地址：/base/v1/auth_users/user_logout***

请求参数：用户信息，例如：

```
{
  "userId": "xxx-1000019",
  "token": "dd243f4.......c32742af3bf6cb",
  "apaasUserId": "xxx-1000019" // 多租户
}
```

返回值：错误码和鉴权信息

- 若数据操作失败，返回 ServiceDBFailed (103)。
- 若用户不存在，Token 过期，Token 不存在，都返回成功（已经成功注销）。

```
{
  "errorCode": 0,
  "errorMessage": "user logout ok"
}
```

具体错误码详情见 错误码 。

删除用户

客户端请求删除账号，在数据库清除用户相关的账户和临时登录信息。

*接口地址：/base/v1/auth_users/user_delete***

请求参数：用户信息，例如：

```
{
  "userId": "xxx-1000019",
  "token": "dd243f4.......c32742af3bf6cb",
  "apaasUserId": "xxx-1000019" // 多租户
}
```

返回值：错误码和鉴权信息

- 若数据操作失败，返回 ServiceDBFailed (103)。
- 若 Token 已过期，返回 UserTokenExpired (203)。
- 若 Token 不正确，返回 UserTokenInvalid (204)。
- 若用户不存在，返回成功（已经成功删除）。

```
{
  "errorCode": 0,
  "errorMessage": "user delete ok"
}
```

具体错误码详情见 错误码 。

修改用户信息

用户可以修改用户信息，例如设置用户名，或者绑定邮箱（手机登录），或设置手机号（邮箱登录）。新用户直接登录后，用户名为空，也可以用这个来判断用户是新用户。

客户端请求注销登录，用户的临时信息。

*接口地址：/base/v1/auth_users/user_update**

请求参数：用户信息，例如：

```
{
  "userId": "xxx-1000019",
  "token": "dd243f4.......c32742af3bf6cb",
  "name": "Alice", // 可选
  "tag": "im", // 可选
  "tag2": "mlvb", // 可选
  "avatar": "https://xxx", // 可选
  "apaasUserId": "xxx-1000019" // 多租户
}
```

特别说明

- name，用户名，昵称。限制 125 个字符。可以是中文。
- tag，用户标签，用户类型等，用作分类用户。
- tag2，用户标签 2，用户类型 2 等，用作分类用户。

返回值：错误码和鉴权信息

- 若数据操作失败，返回 ServiceDBFailed (103)。
- 若 Token 已过期，返回 UserTokenExpired (203)。
- 若 Token 不正确，返回 UserTokenInvalid (204)。
- 若 name/tag/tag2 全部为空，返回 UseInfoEmpty (205)。
- 若 name 和之前一致，返回成功（设置成功）。

```
{
  "errorCode": 0,
  "errorMessage": "update ok"
}
```

具体错误码详情见 错误码 。

获取用户信息

用户登录后，可以查询自己以及其他用户的信息。

*接口地址：/base/v1/auth_users/user_query**

请求参数：用户信息，例如：

```
{
```

```
  "userId": "xxx-1000019",
  "token": "dd243f4.......c32742af3bf6cb",
  "searchUserId": "200001", // 可选
  "searchPhone": "+8618500002222", // 可选
  "apaasUserId": "xxx-1000019" // 多租户
}
```

特别说明

- userId 和 token 用来鉴权。若 search 参数没有指定，则获取用户自己的详细信息。
- searchUserId：获取指定的用户的信息，按用户 ID（userId）查询。
- searchPhone：获取指定的用户的信息，按手机号（phone）查询。

返回值：错误码和鉴权信息

- 若数据操作失败，返回 ServiceDBFailed (103)。
- 若 Token 已过期，返回 UserTokenExpired (203)。
- 若 Token 不正确，返回 UserTokenInvalid (204)。

> ⚠ **注意**
> 由于涉及数据安全，该接口不返回 phone 和 email 字段。

```
{
  "errorCode": 0,
  "errorMessage": "query done",
  "data": {
    "userId": "xxx-1000019",
    "name": "",  // 新用户用户名为空字符串
    "avatar": "https://xxx",
    "apaasUserId": "xxx-1000019" // 多租户
  }
}
```

具体错误码详情见 错误码 。

心跳和保活

用户登录后，每隔 10 秒向服务器保活 (KeepAlive)。

- 用户若掉线，会根据不同的业务定义，更新相关的状态，例如 KTV 会在房主和主播掉线后，将房间设置为不可见。
- IM 用户状态回调，也会更新用户的在线状态。

*接口地址：/base/v1/auth_users/user_keepalive**
请求参数：用户信息，例如：

```
{
  "userId": "xxx-1000019",
  "token": "dd243f4.......c32742af3bf6cb",
  "apaasUserId": "xxx-1000019" // 多租户
}
```

特别说明
userId 和 token 用来鉴权。
返回值：错误码。

- 若数据操作失败，返回 ServiceDBFailed (103)。
- 若 Token 已过期，返回 UserTokenExpired (203)。
- 若 Token 不正确，返回 UserTokenInvalid (204)。

```
{
  "errorCode": 0,
  "errorMessage": "keep alive done"
}
```

具体错误码详情见 错误码 。

OAuth：用户签名注册

一般客户有自己的用户管理系统，包括用户注册和认证，在 Demo 中也提供了一种注册用户签名的方式，这样可以使用用户签名登录。
*接口地址：/base/v1/oauth/register**
请求参数：用户信息，例如：

```
{
  "username": "alice",
  "salt": "1e80a39bf9e1cb8e97cdaee2ace85281"
}
```

特别说明

- username：用户名，也可说是 ID 信息，邮箱，或者手机号等等。
- salt：使用 password 生成 salt，例如：md5 (username-password)
- 登录验证时，用 salt 作为 secret：signature=md5 (username-tag-ts-nonce-salt)

- 生成 salt 和 signature 的 nodejs 例子：

```
const md5 = require ('md5');
const username = 'user', password = 'xxx';
const salt = md5 (`${username}-${password}`);
const tag = '', ts = '', nonce = '';
const signature = md5 (`${username}-${tag}-${ts}-${nonce}-${salt}`);
```

返回值：错误码和鉴权信息
以下信息给 SDK 鉴权使用。

```
{
  "errorCode": 0,
  "errorMessage": "register ok"
}
```

具体错误码详情见 错误码 。

OAuth：用户签名登录

使用固定的 Secret，以及必要的用户信息，对消息签名后，就可以登录系统。
*接口地址：/base/v1/oauth/signature***
请求参数：用户信息，例如：

```
{
  "username": "alice",
  "signature": "b081b396acc4948019e124294c313396",
  "tag": "player", // 可选
  "ts": "1634543466000", // 可选
  "nonce": "o29dmek", // 可选
  "hash": "md5", // 可选
  "nickname": "Guest", // 可选
  "avatar": "http://tencent.com/xxx.png" // 可选
}
```

特别说明

- username：用户名，也可说是 ID 信息，邮箱，或者手机号等等。
- tag：签名的 secret 的标签，可以根据 tag 使用不同的 secret。
- ts：签名时的时间戳。例如：1634543466000

- nonce：签名使用的随机字符串，也可是自定义 hash。例如：o29dmek
- hash：签名使用的算法。例如：md5。
- signature：使用 secret 对这些参数签名，例如：md5 (username-tag-ts-nonce-secret)
- nickname：用户昵称，例如用来设置到 IM 的。

返回值：错误码和鉴权信息
以下信息给 SDK 鉴权使用。

```
{
  "errorCode": 0,
  "errorMessage": "signature ok",
  "data": {
    "userId": "xxx-1000019",
    "sdkAppId": 1400544182,
    "sdkUserSig": "eJwtjMEKwjA.............bKC9S",
    "token": "9d0dk28r31........c327id192k9f",
    "expire": "2021-08-22 06:42:54",
    "apaasAppId": "xxx", // 多租户
    "apaasUserId": "xxx-1000019" // 多租户
  }
}
```

具体错误码详情见 错误码 。

获取房间列表

客户端登录后，可以根据 ID 获取房间的详细数据。
*接口地址：/base/v1/rooms/query_room***
请求参数：用户信息和房间，例如：

```
{
  // 用户 (user) 信息
  "appId": "1400000212",
  "userId": "xxx-110016",
  "token": "ju121772Kd1n39",
  // 房间查询信息
  "category": "liveRoom", // 可选
  "roomIds": ["xxx-905449688"], // 可选
  "startUtc": "2021-09-12 16:00:00",  // 可选
  "endUtc": "2021-09-13 16:00:00",  // 可选
```

```
  // 分页查询信息
  "offset": 0,  // 可选
  "limit": 100,  // 可选
  // 多租户
  "apaasUserId": "xxx-110016"
}
```

特别说明

- appId：为 TRTC/TIM 的 sdkAppId。由于不同的 sdkAppId 之间的 category 可能重复，因此必须带 sdkAppId 参数。
- category，roomIds，startUtc/endUtc，查询参数。
- offset，查询的起始索引，默认 0。
- limit，查询的最多房间数目，默认 10，不超过 100。

返回值：错误码和房间 ID

```
{
  "errorCode":0,
  "errorMessage":"",
  "data": [
    {
      "roomId": "xxx-905449688",
      "category": "liveRoom",
      "title": "A live room",
      "ownBy": "xxx-110016",
```

错误码

错误码	描述	含义
98	SystemVerifyError	验证用户操作或参数错误。
99	SystemError	后端未知的系统错误。
100	ServiceUnavailable	后端依赖的服务不可用。
101	ServiceResponseInvalid	后端服务返回的结果不符合规范。
102	ServiceSmsFailed	SMS 发送短信服务失败。
103	ServiceDBFailed	DB 错误，SQL 语句执行失败。

104	ServiceUnknownError	服务未知错误。
200	UserCodeInvalid	用户登录失败，错误的验证码。
201	UserCodeExpired	用户登录失败，验证码已过期。
202	UserCodeConsumed	验证码已经使用过了，请重新获取。
203	UserTokenExpired	用户 Token 已经过期，请重新登录。
204	UserTokenInvalid	用户 Token 不正确，请重新登录。
205	UseInfoEmpty	用户的信息为空。
206	UserNotExists	用户不存在。
207	UserEmailInvalid	给用户发送邮件时参数错误。
208	UserQueryParams	用户查询时参数错误。
209	UserQueryLimit	用户查询数据过多，触发限流。
210	UserOAuthParams	用户 OAuth 参数异常。
211	UserAlreadyInRooms	用户已经在房间中。
2000	RoomCreateParams	新建房间时参数错误。
2001	RoomDestroyParams	删除房间时参数错误。
2002	RoomQueryParams	查询房间时参数错误。
2003	RoomLeaveParams	退出房间时参数错误。
2004	RoomNotExists	房间不存在。
2006	RoomsUserExists	用户已经在房间中。