云函数

实践教程

腾讯云

文档目录

实践教程

实践教程概述

最近更新时间：2025-01-21 18:21:42

根据云函数的特点，我们推荐您这样使用：

- **以无状态的风格编写函数代码，确保您的代码不会进行任何状态维护。本地存储和内存结果都是可能丢失的，应当使用 COS、Redis/Memcached 等服务缓存中间信息并落地最终计算结果。**
- **在执行方法外实例化任何可能复用的对象，例如数据库连接等。**
- **请务必在已上传的 ZIP 中设置对您的文件的 `+rx` （可读及执行）权限，以确保代码能够执行。**
- **在代码中尽可能多地使用 log/print 语句，给调试工作带来充足的信息。**

腾讯云云函数可结合众多云上产品，构建丰富的解决方案。如下表所示：

合作产品	解决方案
Serverless Cloud Framework	SSR 框架迁移
	Serverless HTTP 服务
	静态网站托管
	接入 Serverless DB
	部署流式转码应用
API 网关 APIGW	提供 API 服务
	快速构建文字识别小工具
	实现 Web 静态页面托管
	基于 Websocket 搭建匿名聊天室
	处理多文件上传
	实现自定义邀请函
小程序云开发 TCB	在小程序云开发中实现函数互调及邮件发送
实时音视频 TRTC	输入在线媒体流
	实现单流录制
	实现混流录制
对象存储 COS	实时音视频转码
	日志分析写数据库
	获取图片并创建缩略图
	实现身份证文字识别
	数据入湖解决方案
	自定义计算文件哈希值
	实现自定义转码
	实现 MapReduce
消息队列 CKafka	消息数据自定义处理并投递至 COS
	消息转储至 Elasticsearch

日志服务 CLS	消息转储至消息队列 Ckafka
	消息转储至对象存储 COS
负载均衡 CLB	快速部署 Web 服务
视频处理 MPS	视频任务回调备份 COS
	视频任务回调通知工具
内容分发网络 CDN	定时预热刷新
腾讯云数据仓库 TCHouse-P	导入 Ckafka 数据到腾讯云数据仓库 TCHouse-P
云点播 VOD	接收事件通知
短信 SMS	实现短信验证码功能
Elasticsearch Service	快速构建搜索服务
定时任务	页面内容定时采集
	定时拨测并通过邮件发送告警
	定时备份数据库到 COS

欢迎订阅 Serverless 技术专栏 了解更多最新实践和腾讯云 Serverless 动态。

说明：
在下文的具体实践中，大都通过模板函数的形式来部署函数。您可自行下载代码来分析学习，模板函数和代码均支持下载操作。

业务开发相关实践
函数并发管理实践

最近更新时间：2024-06-12 16:01:51

操作场景

腾讯云 Serverless 云函数已上线并发管理能力升级版，该升级版提供3个维度的并发额度管理功能。通过该功能，可以获得更强的函数并发管理控制权限，无需再等待申请云函数配额即可自行根据业务需求快速调整。

> **说明**
> 目前 函数并发 管理功能已上线，函数可以配置 最大独占配额 。

未使用并发管理功能前，一个云函数默认具有300的并发数量上限。针对使用量较小的低频业务，300并发值已足够满足业务使用。但遇到业务量上涨、支撑大型运营活动等大并发场景，开发者则需提交工单申请提升函数并发额度，该解决方案存在以下3种情况：

- 业务每遇到一次大并发，就需要联系腾讯云申请提升云函数配额，时效性弱。
- 申请等待周期时会导致上涨的业务部分有损。
- 在评估量级时，可能会因评估不足而未通过，导致需要再次申请，效率较低。

本文将为您全方位介绍并发管理功能，并针对多种使用场景提供配置建议。

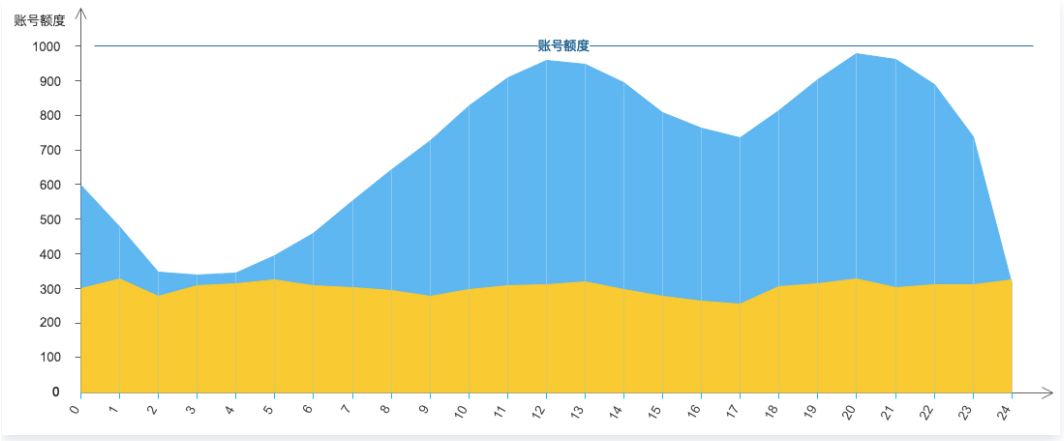
功能介绍

并发能力升级

相对于原有的函数默认固定并发值，云函数新上线的并发管理能力，有以下方面优化：

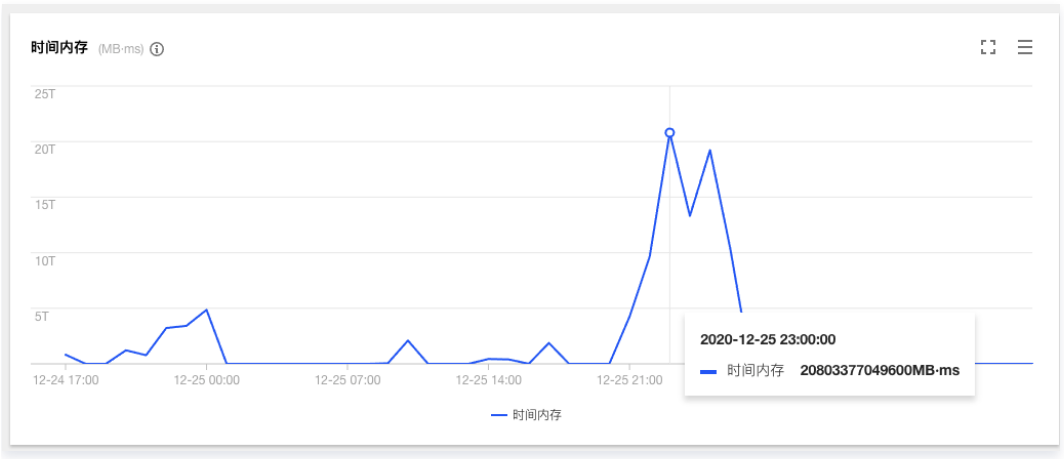
- 开放单一函数的并发调整，用户可自行控制并发数。
- 并发额度由单一的函数维度转移到账号维度。
- 账号默认具有一定的并发额度，由账号下的函数所共享，无需单独为其中一个大并发函数申请额度。

通过并发管理能力，当前账号维度的并发配额为128000MB，即可支持1000个128MB配置的函数实例同时运行。用户无需再申请提升配额，即可获得比调整前更高的一个并发额度，用于支撑上涨的业务。云函数额度变化如下图所示：



内存和并发额度的关系

当前并发额度按内存进行计算，配置内存大的函数，并发运行时占用的额度多，而配置内存小的函数在多并发运行时占用的额度小。
由于函数服务的资源用量计费项和函数的配置内存强相关，如需通过内存进行额度控制，建议您针对业务选择适合的函数内存，可针对整体支出进行有效控制。内存变化如下图所示：



实践建议

并发使用场景建议

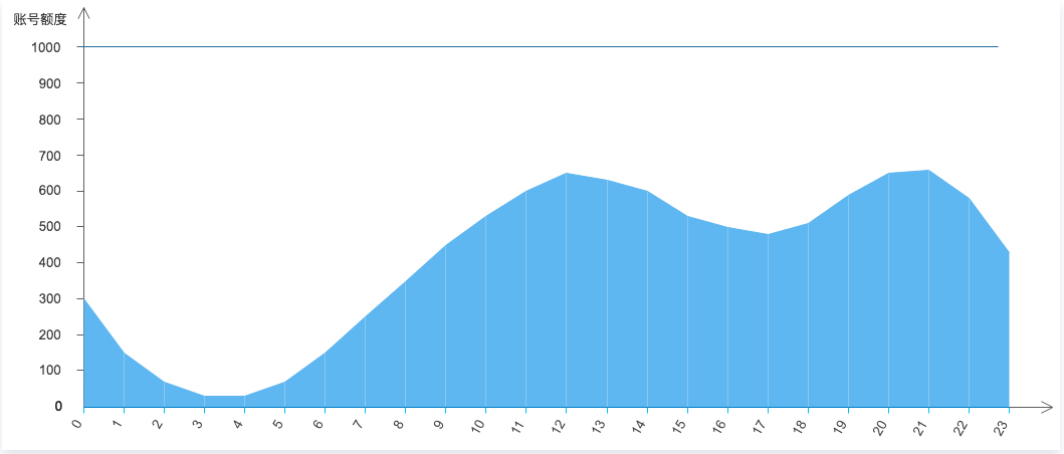
一个账号下有多个业务同时使用云函数进行支撑时，云函数的并发配额则需要进行按需调度。根据不同的业务特性来进行合理的设置。本文分别从不同的业务类型、特点及要求进行分析：

业务类型	业务特点	业务要求
前端业务	存在波峰波谷	保障用户体验，要求加载速度快，可以有一定的错误容忍度
数据处理业务	平稳运行波动不大	不能接收延迟、波动或失败
运维任务	定时任务，偶尔运行一次	无需投入过多关注，运行正常即可
视频处理业务	并发量不大但计算量复杂、计算时间长	接受按需调度，失败时能自动运行即可

根据不同的业务特性、容错额度、业务波动情况、时延要求，即可按照不同的场景进行不同的配置。上述几项业务中，并发配置建议如下：

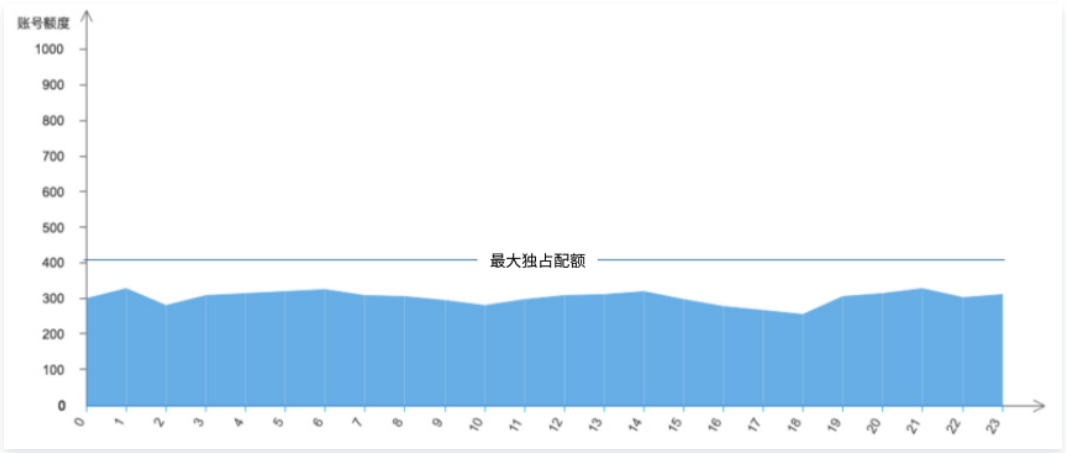
前端业务

针对要求加载速度快、存在波峰波谷的前端业务，可以为函数配置一定量的预置额度。例如，按最大使用量的60%来设置，但同时不配置函数的最大独占配额，确保在高峰到来时能充分利用总配额。云函数额度变化如下图所示：



数据处理业务

针对波动不大但是容错低的数据处理业务，可以为函数配置一定量的最大独占配额，确保其他业务不会使用共享额度导致数据处理业务的问题。云函数额度变化如下图所示：

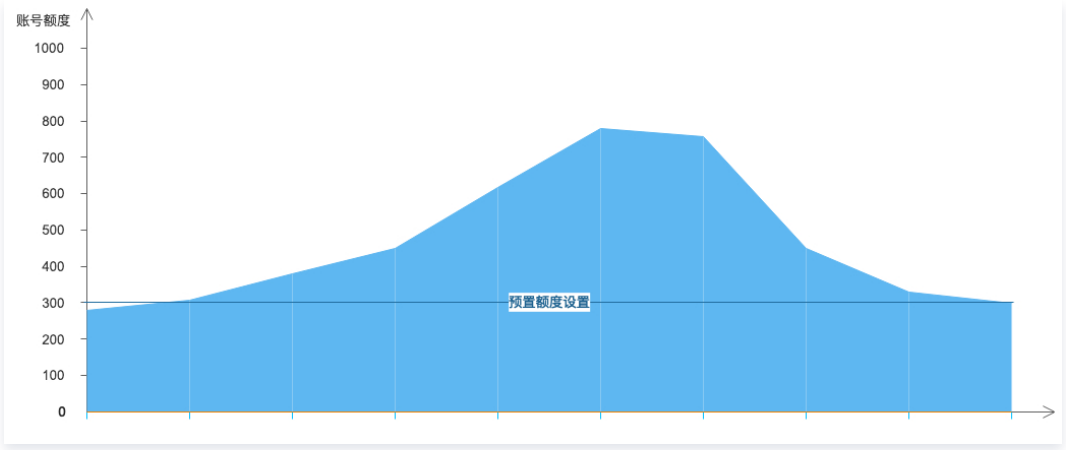


运维任务

针对要求不高、定时运行的运维任务，可以不做任何配置，使用账号维度的配额处理即可。

视频处理业务

针对计算量大且按需排队处理任务的视频处理业务，可以配置一定的预置额度，让业务跑满并发额度，充分利用和控制好计算资源。云函数额度变化如下图所示：



最大独占配额处理

账号级别的额度由账号下的多个函数之间共享，在多个函数同时运行的情况下，因为流量突增、业务上涨导致并发增高的函数在占用完全部空闲额度后，可能会与平稳运行的函数产生冲突。

针对上述场景，有以下两种解决方案：

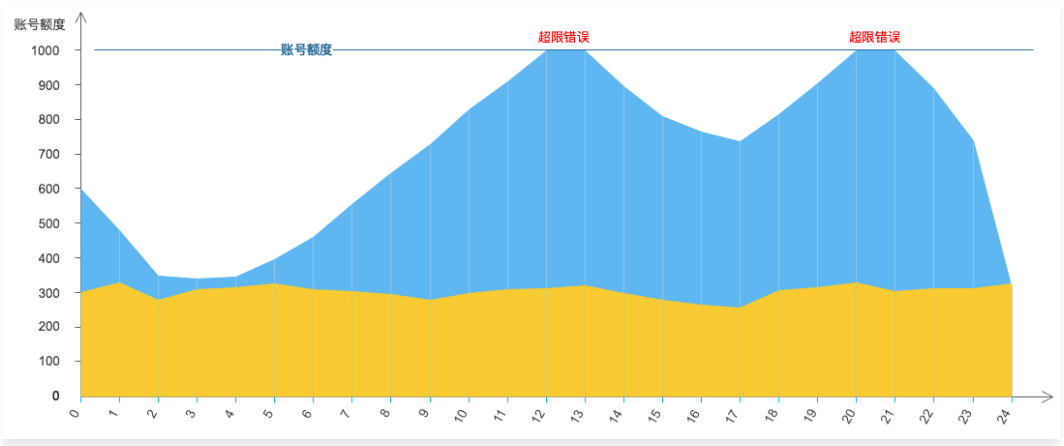
- 解决方案1（推荐）：最大独占配额配置，通过将部分额度分配给具体函数，来保障具体函数的运行可靠性。
- 解决方案2：通过购买更高规格的 套餐包 来获得更高的并发额度，以满足业务量上涨带来的并发上涨。

最大独占配额配置示例

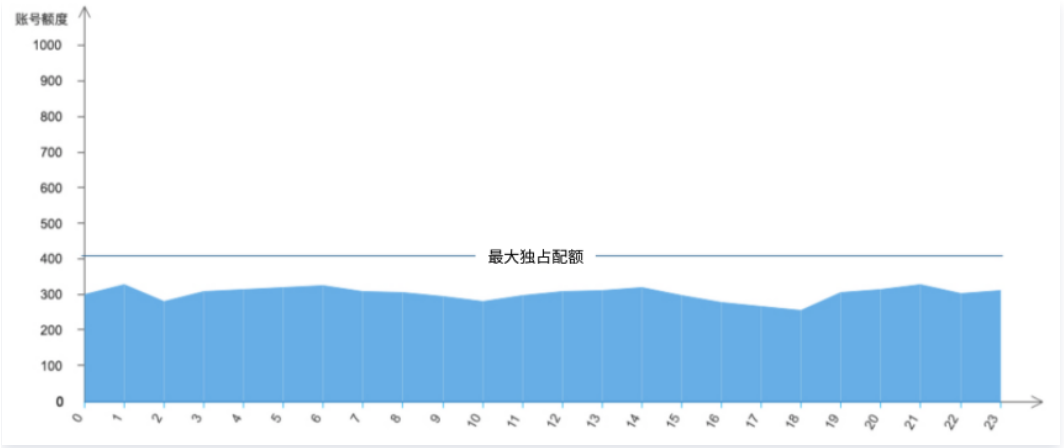
本文以解决方案1（最大独占配额配置）为例详细介绍如何使用最大独占配额：

示例场景：在同个账号下，函数 A 提供 H5 页面用于秒杀的运营活动，函数 B 在进行后台的流式数据处理。在 B 函数启动300并发进行业务处理时，运营活动会受限于 A 函数，最多仅能运行700并发。而在此时函数 A 的业务压力下，如果 B 函数也面临业务量上涨，将无配额可用导致无法启动更多实例。云函数额度变化

如下图所示：



示例配置：在上述示例中，如需保证 B 函数数据处理流程的可靠性，可以为 B 函数设置到350的最大独占配额。此时，该额度将从账号维度划给 B 函数单独使用，而运营活动所使用的 A 函数将仅可以最大使用650并发的额度。函数 B 在设置到350的最大独占配额额度后，在业务持续上升后，最大也仅可使用到配置的额度，即使账号维度的额度仍然有剩余，B 函数也无法使用。云函数额度变化如下图所示：

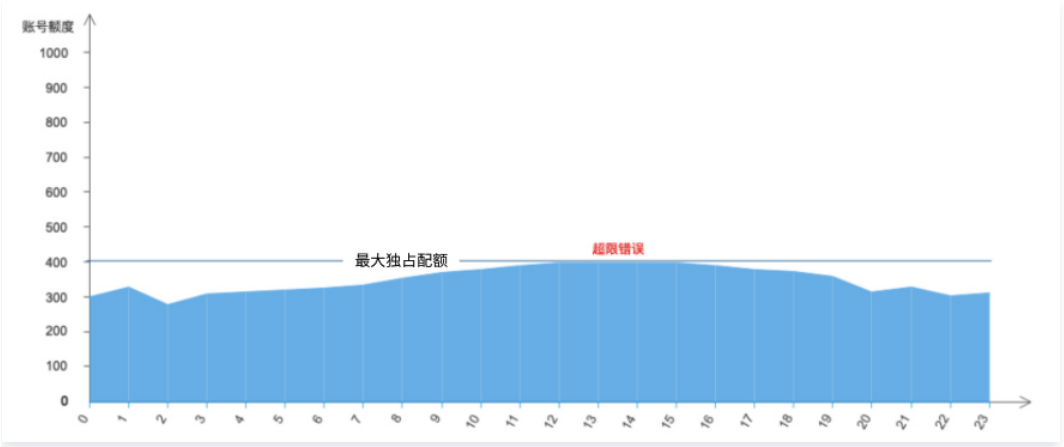


通过最大独占配额的配置：

- 可以保障函数正常运行，避免多个函数共享额度时，由于其他业务的函数占用导致本函数无法运行产生损失。
- 定义了函数的运行额度上限。

最大独占配额配置建议：针对函数的并发管理控制，可以基于业务来进行更精细的控制，您可参考以下3点建议进行配置：

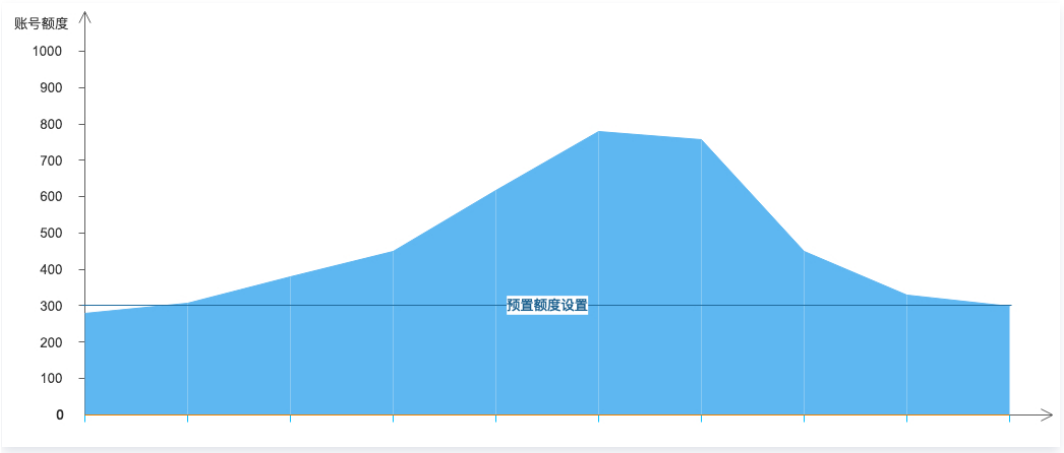
- 针对开发测试阶段的函数，由于请求量小、无业务压力、并发极少，可以不配置最大独占配额而仅仅使用账号维度的共享额度。
- 针对稳定运行的函数，通常可以确定并发量，浮动范围不会很大，可以为该函数配置有一点余量的最大独占配额，来保障函数的额度不受共享的影响。
- 针对运营活动、有可能突增并发的函数，可以利用账号维度的高额度，来充分利用和支撑业务爆发。



相关说明

当前预置额度设置在函数的版本上，是从账号维度的并发配额或函数上的预置额度扣减而出。

通过配置预置额度可以预设启动所需量的并发实例，完成实例的初始化并等待事件发生。针对函数的请求将不会有冷启动时间，可直接在已完成准备、初始化完成的实例中得到运行。
针对时延敏感的业务（例如前端的 SSR 页面响应）、或是初始化时间较长的业务（例如 AI 推理的模型加载过程），通过为函数配置上一定的预置可以保障业务更好的运行。
同时，预置的配额并非实例并发的上限，在业务量上涨到超过已经预置的实例可以承载的最大量时，云函数仍会根据函数的预置额度或者账号配额，拉起更多的实例来支持业务运行。云函数额度变化如下图所示：



最大独占配额其他用法

对业务的限制或关停同样可以通过最大独占配额来配置。在突发紧急事项时，例如漏洞攻击、循环调用失控等，为避免出现重大损失，可以通过配置最大独占配额，将额度控制在极小的值以避免运行失控，甚至可以配置为0关闭函数的运行，详情请参见 最大独占配额 。配置如下图所示：

并发高性能架构最佳实践

最近更新时间：2024-09-02 16:59:41

并发是云函数在某个时刻同时处理的请求数。在业务其他服务可以支撑的情况下，您可以通过简单的配置实现云函数从几个并发到数以万计并发的拓展。

应用场景

高 QPS 短运行时长

使用云函数进行简单的数据、文件处理，例如对象存储触发云函数进行信息上报、对文件处理等。此类场景下单次请求运行时间较短。

重计算长时间运行

使用云函数进行音视频转码、数据处理、AI 推理等场景，由于模型加载等操作导致函数初始化时间长、函数运行的时间长。包括 Java 运行环境的初始化时间较长。

异步消息处理

使用云函数做异步消息处理的场景较多，例如全景录制场景，腾讯云自研的全景录制，主打所见即所得的录制理念；TDMQ 函数触发场景，可最大程度的衔接消息队列两端的数据上下游，帮助用户实现 Serverless 体系下的异步事件解耦和削峰填谷的能力等。

产品优势

通过综合使用最大独占配额、预置并发等能力，您可以灵活调配多个函数间的资源用量情况，并按需预热函数。

共享配额

在未进行各项配置的情况下，各函数默认共享使用账号额度。如果有某个函数产生了突增业务调用，可以充分利用空闲未使用的额度，来保证突增不会引起函数的调用并发超限。

并发保障

如果有某个函数的业务功能比较敏感或关键，需要尽力保障请求的成功率，此时可以使用最大独占配额。最大独占配额可以给到函数独享额度，确保并发的可靠性，不会由于多函数的争抢导致的调用并发超限。

预置并发

在函数对冷启动敏感、或代码初始化流程耗时较长、加载库较多的情况下，可以通过设置具体函数版本的预置并发，预先启动函数实例来保障运行。

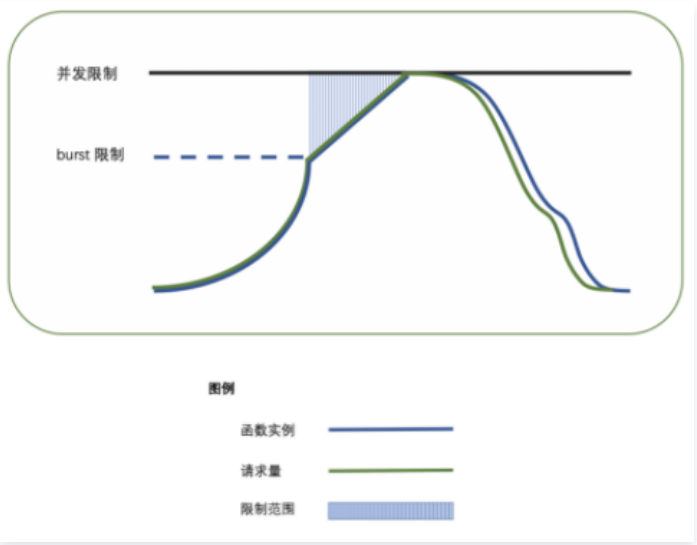
并发扩容原理

并发实例复用与回收与并发扩容原理详情见 并发概述 。

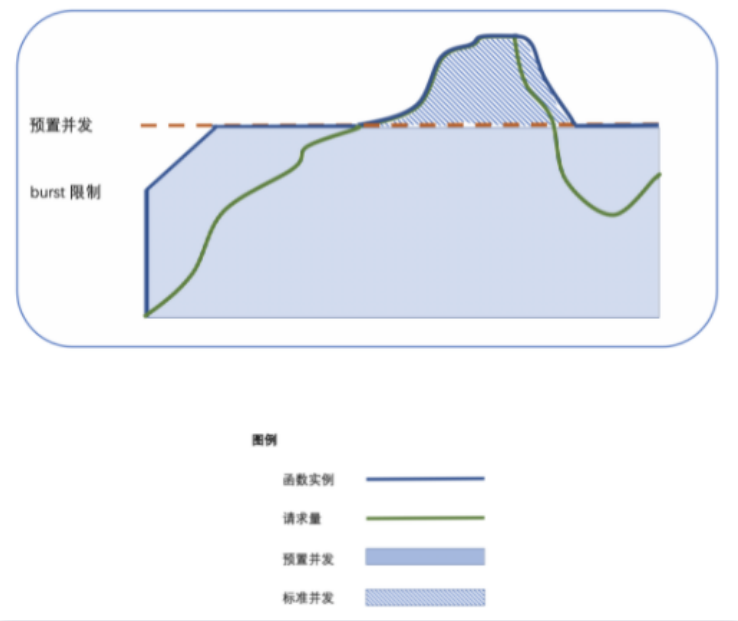
示例

例如，广州地域的账号默认并发额度可以支撑128MB函数的1000个并发实例。有大量请求到来时，第一分钟可以从0个并发实例启动到500个并发实例。如果还有请求需要处理，第二分钟可以从500个并发实例启动到1000个并发实例。
下图模拟了业务流量峰值时函数并发处理的具体场景。随着业务请求量不断增加，没有可用并发实例处理新的请求时，函数将启动新的并发实例进行扩容。当达到弹性并发的扩容速度限制时，函数扩容速度逐渐变慢，新的请求将会受到限制尝试重试。接着函数会继续扩容，最终达到函数区域对账户的并发限制。最终在满足

业务需求后，请求量逐步减少，函数未使用的并发实例会逐步停止。



预置并发支持并发实例按配置预先启动，同时云函数平台不会主动回收这些实例，会尽可能地保障有相应数量的可以处理请求的并发实例。您可通过此功能，为函数的指定版本设定预置并发额度。通过配置预置并发，可预先进行计算资源的准备，降低冷启动、运行环境初始化及业务代码初始化引起的耗时。下图模拟了函数在处理业务流量峰值的预置并发的真实情况。



场景压测

场景1：高 QPS 短运行时长

此类场景下，QPS 较高单次请求运行时间较短，业务在冷启动的一两秒会有并发高峰。接下来我们通过测试观察流量逐步切换或配置预置并发是否能够缓解冷启动并发高峰。

业务情况

业务信息	指标
函数初始化时长	业务没有初始化的操作
业务运行时长	5ms
QPS	峰值约10w左右

压测任务

我们计划分为三个压测任务，分别对应：完全冷启动、逐步切流量、配置预置并发。
每次压测任务都需要从冷启动开始，不能有热实例在；无其他函数影响。

压测目标

高 QPS 的业务在冷启动的一两秒会有并发高峰，通过流量逐步切换或配置预置并发可以缓解冷启动并发高峰。

压测配置

函数配置	a. 内存：128M，未开启异步执行和状态追踪，关闭日志投递 b. 并发配额：4000 × 128M c. duration：5ms d. burst：2000
测试工具	使用 go-wrk 工具，直接调用 RegionInvoke 接口。

1.完全冷启动

客户端2k并发，调用2k × 5k次，统计并发执行情况以及冷启动并发情况。
性能表现函数并发执行个数如下图所示，在3分钟内完成了整个并发扩缩容的流程。

函数冷启动数据如下图所示，并发可以瞬时启动，在1分钟内达到了我们设置的 burst 为2000冷启动限制。

该场景下平均 QPS 达到 6w左右，并发可以在1分钟内瞬时启动，burst 冷启动限制正常，在3分钟内完成了这个并发扩缩容的全过程。

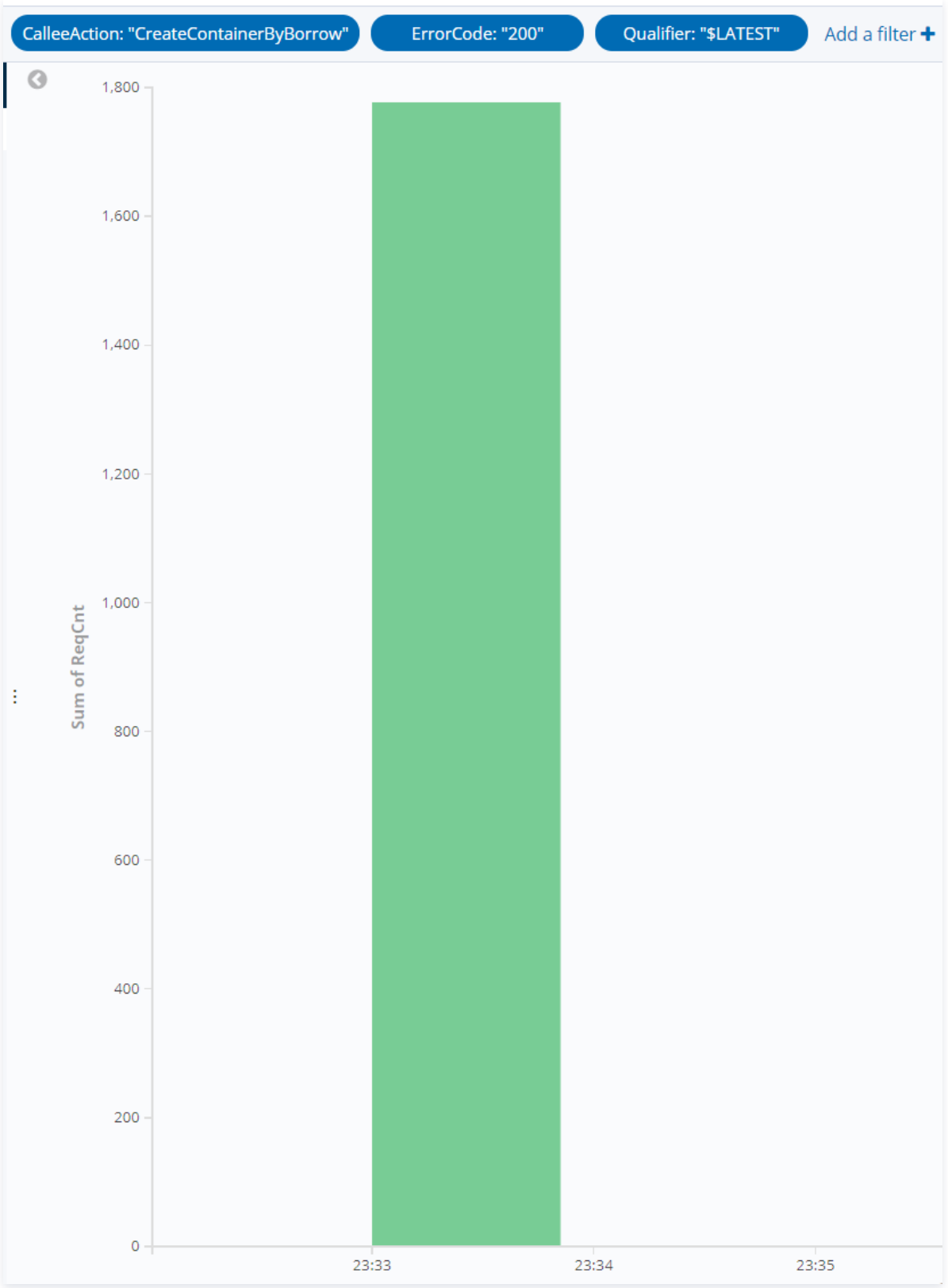
```
[root@VM_2_188_centos /data/home/pigpigzhu/test_pt]# ./invoke.sh -I 1253970226 -u 3473058547 -U 100019716422 -F /data/p_lizhili/config.json -c 200 -N 5000 -S Scf.ap-chengdu-pt-1.Regio
nInvoke.OldHttpObj -P Test -R lam-92m8en2h
function lam-92m8en2h already in ap-chengdu-pt-2 cluster.
use polaris
log file path:/data/log/pt/perf_test_m
Summary:
startTime: 2021-09-01 16:07:58 endTime: 2021-09-01 16:10:42
timestamp start: 1630483678, end: 1630483842

Function: pressure-test-qps-1[lam-92m8en2h] pigpigzhu

Client:
Nodes: 10
Request total: 10000000
Avg QPS: 60975.61
CostTime total: 309932549.00ms
Avg cost time: 30.99ms
CostTime P95: 85.00     P75: 29.00      P50: 20.00      P25: 16.00
Avg duration time: 6.24ms
Concurrency: 200
Requests: 5000
Sleep: 0s
[root@VM_2_188_centos /data/home/pigpigzhu/test_pt]#
```

2.发布新版本逐步切流量

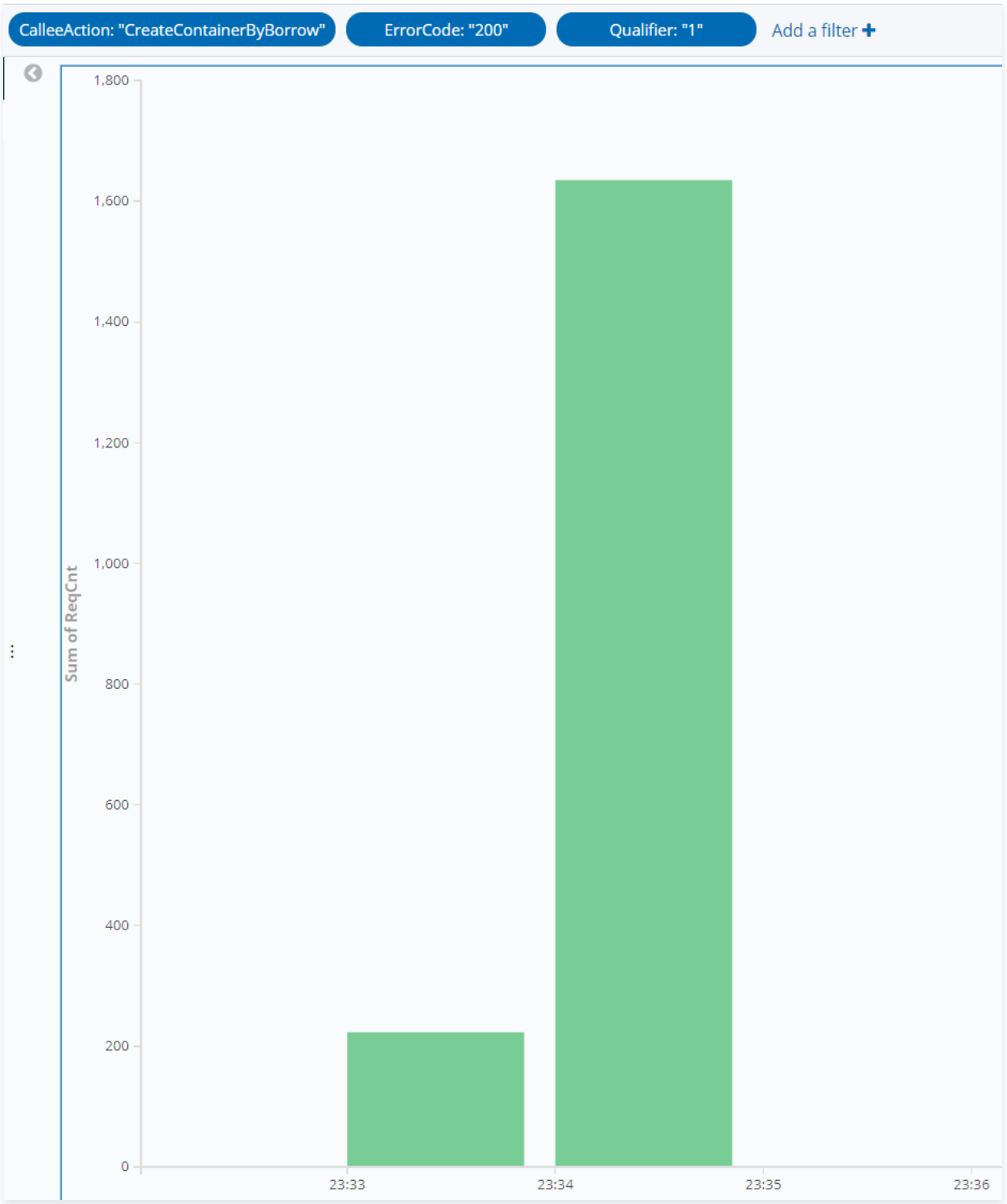
客户端2k并发，调用2k × 5k次。新版本1分钟内请求从0并发提升1k、再到2k并发，老版本并发由2000降低到1k、再到0并发，统计新老版本并发次数以及冷启动并发情况。性能表现并发折线图旧版本如下图所示：

并发折线图新版本如下图所示：

冷启动并发折线图旧版本如下图所示：



冷启动并发折线图新版本如下图所示：



从上述数据可以看到，通过逐步发版本切流量的方式可以降低并发的突然高峰。

3.配置预置并发

该函数预置2000个并发。预置启动成功后，2000个并发调用重复5k次，统计并发次数以及冷启动并发情况。性能表现并发折线图如下所示：

冷启动并发如下所示：

可以看到整个区间冷启动数为0。函数请求次数如下图所示：

从上述数据可以看到，通过配置预置实例可以使得并发冷启动数为0，推荐客户配置预置并发来保障性能、避免初始化时间过长的问题。

结论

综上，在高 QPS 短运行时间的场景，使用流量逐步切换可以缓解冷启动并发高峰，同时通过预置并发解决客户认为初始化过程（包含函数冷启动）时间过长的问题。

场景2：重计算长时间运行

业务情况

业务信息	指标
函数初始化时长	10s
业务运行时长	2mins
QPS	约20

压测目标

长时间计算任务的平均 QPS 不高，但由于计算时间长，大量实例处于运行状态，导致云函数的并发很高。此压测场景希望看到云函数在处理大量较长时间运行任务的时候，任务的调度和处理速度。

压测配置

函数配置	a. 内存: 128M 开启异步执行和状态追踪，关闭日志投递 b. 并发配额：2000 × 128M c. duration： 2min d. burst：2000
测试工具	使用 ab 工具，模拟 cos 服务的消息，通过函数的 cos 触发器调用函数。

压测任务

每次压测任务都需要从冷启动开始，不能有热实例在；无其他函数影响。

长初始化、长运行时间场景、每个并发1个消息

2000并发，每个并发1个消息，累计投递2000个请求。统计从第一个请求到达的开始时间、最后一个请求返回的结束时间，以及请求总处理时间的分布情况。性能表现并发折线图如下所示：

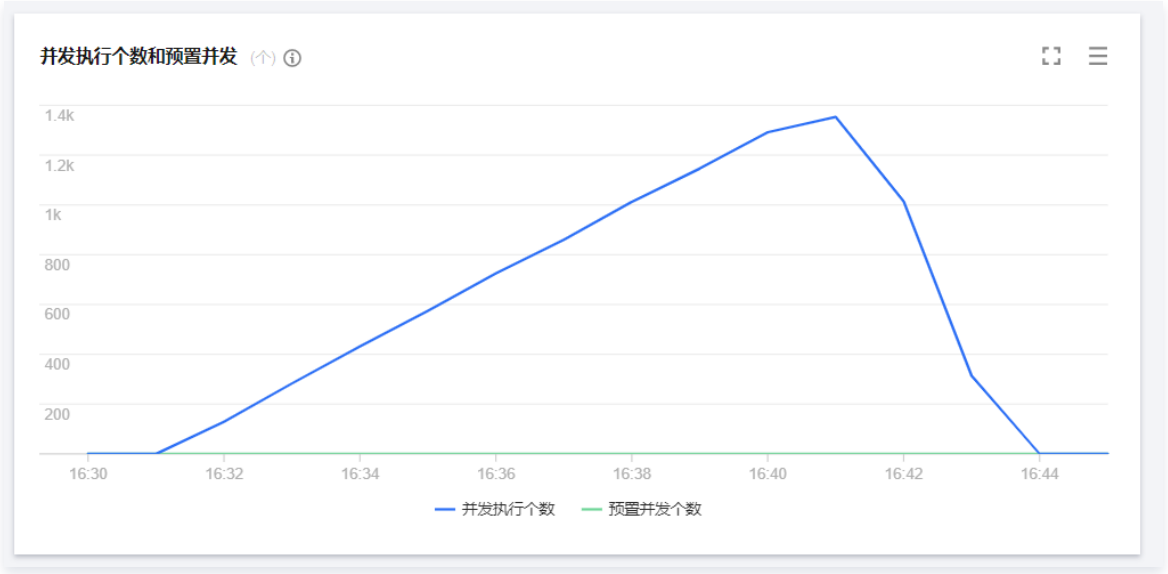
冷启动折线图如下所示：

函数请求次数如下所示：

冷启动个数为何并不是同一分钟内冷启动完成？
a. 异步场景，请求次数并不是一起到达 RegionInvoke，而是通过异步触发器的 worker 调用 RegionInvoke。
b. 因为函数 duration 为2分钟，初始化10秒钟，总体耗时在3分钟内，所以随着函数请求次数的不断提高，但是前面启动的容器还没有释放，需要不停的启动容器来完成请求，函数请求次数每分钟增长的速率基本和冷启动的个数吻合。
c. 函数并发到达最高点时冷启动开始下降，基本吻合符合正常。

长初始化、长运行时间场景 、每个并发2个消息

2000并发，每个并发2个消息，累计投递4000个请求。统计从第一个请求到达的开始时间、最后一个请求返回的结束时间，以及请求总处理时间的分布情况。性能表现并发折线图如下所示：



冷启动折线图如下所示：

函数请求次数如下所示：

结果分析

4000消息更能印证2000消息时的结论，前两分钟因为之前的函数没有释放，所以冷启动数和函数请求数是一致的，后续冷启动数和函数请求数增长基本保持一致。

结论

长初始化、长运行时间的业务能够稳定运行，系统能够随着请求数瞬时的完成扩缩容。

场景3：异步消息处理

业务背景

使用云函数做异步消息处理的场景较多，这里取运行100ms作为平均值。

业务信息	指标
函数初始化时长	0
业务运行时长	100ms

压测目标

异步消息的消费情况与生产相关，我们假设已经有大量的消息堆积，查看云函数的消费情况。其中会包括消费过程中受到并发限制而后重试的情况，您可以通过灵活调整函数的保留并发，从而控制云函数的消费速度，也就是对下游后台的压力。

压测配置

函数配置	a. 内存: 128M　开启异步执行和状态追踪，关闭日志投递 b. 并发配额：X × 128M c. duration：100ms d. burst：2000
测试工具	使用 ab 工具，模拟 cos 服务的消息，通过函数的 cos 触发器调用函数。

压测任务

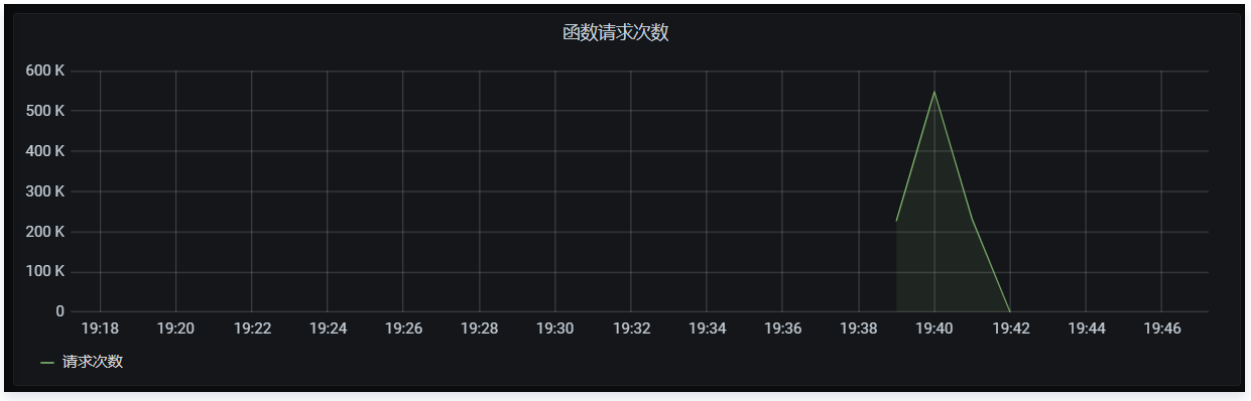
每次压测任务都需要从冷启动开始，不能有热实例在。

异步消息1k并发

函数保留 2k 并发，投递100w条消息进行消费，记录总消费时间、消息处理的分布情况，并发监控。性能表现并发折线图如下所示：

冷启动折线图如下所示：

函数请求次数如下所示：

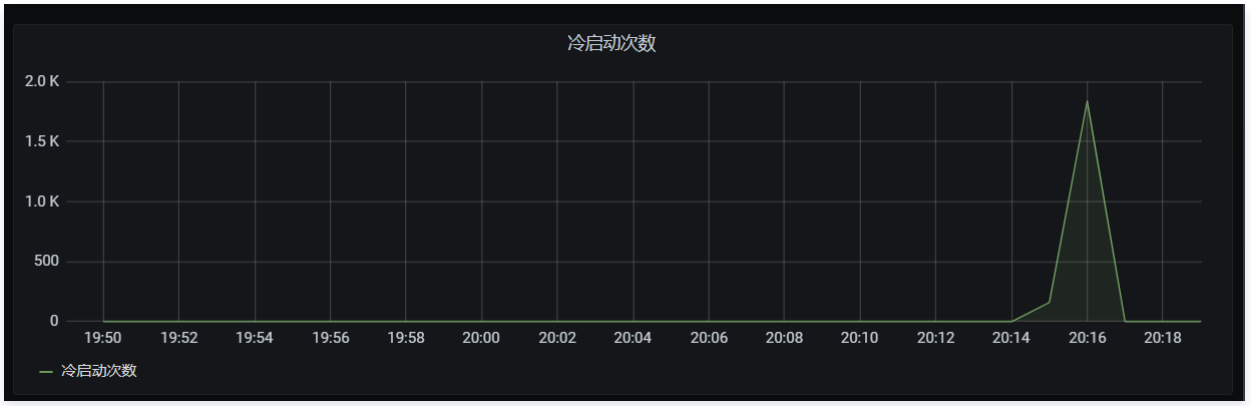


共计3分钟，处理完所有消息。

异步消息2k并发

函数保留 2k 并发，投递100w条消息进行消费，记录总消费时间、消息处理的分布情况，并发监控。性能表现并发折线图如下所示：

冷启动折线图如下所示：

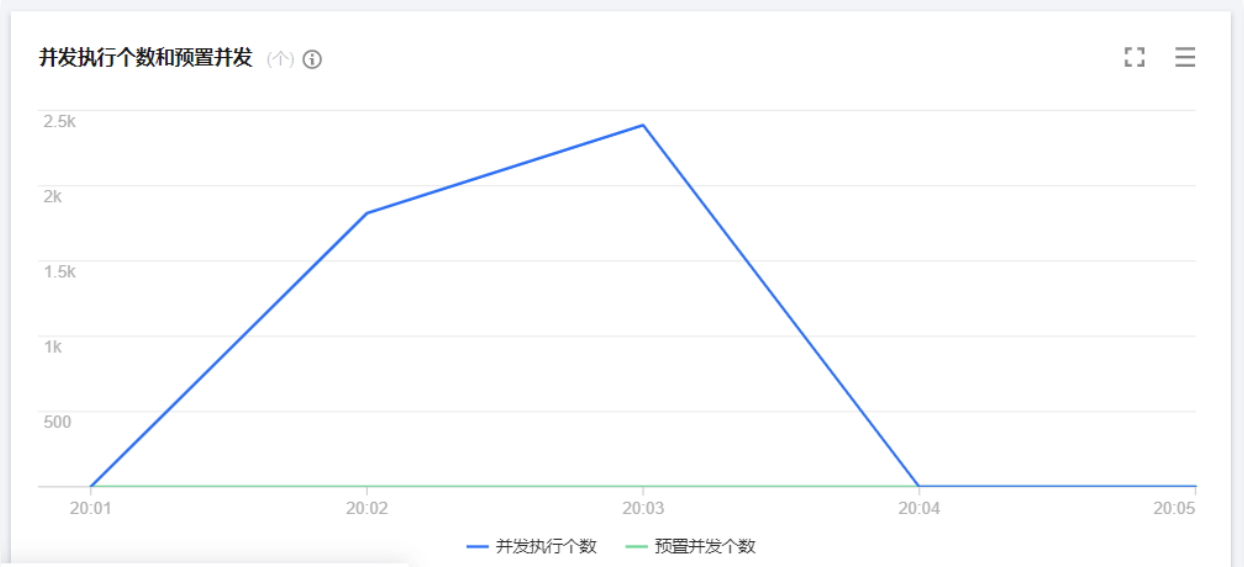


函数请求次数如下所示：

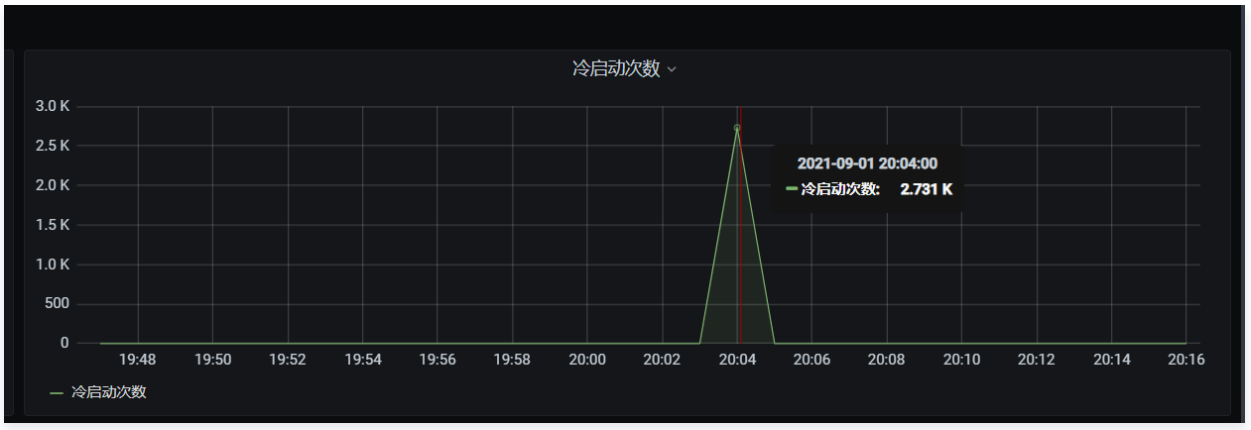
共计2分钟，处理完所有消息。

异步消息4k并发

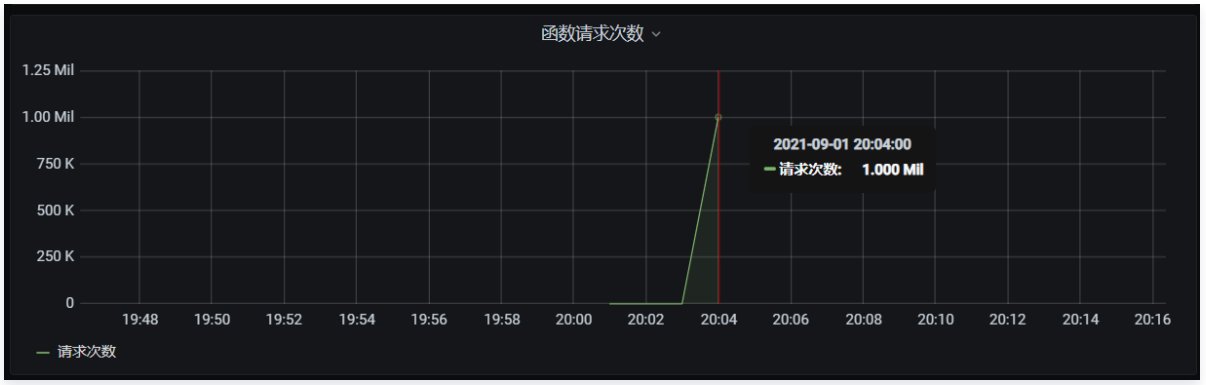
函数保留 4k 并发，投递100w条消息进行消费，记录总消费时间、消息处理的分布情况，并发监控。性能表现并发折线图如下所示：



冷启动折线图如下所示：



Dashboard 上函数冷启动数据，该冷启动有 burst 超限。
函数请求次数如下所示：



共计1分钟，处理完所有消息。

结果分析

1. 1000并发配额提升到2000并发配额时，能看到函数的处理速度，包括函数并发等明显提升，所以在大量异步消息时提高并发配额可以提升消息处理速度。
2. 2000并发配额提升到4000时，消息处理速度和函数并发量也有提升，但总体处理时间也在2分钟内，说明100w条异步消息时，4000肯定也能提升，但2000配额基本已满足场景需求。
3. 4000并发配额时会有 burst 超限的问题，因为并发配额是4000，burst 是2000，所以大量消息时肯定会有超限的情况发生。

结论

大量异步消息场景，提升函数并发配额，能明显提升消息处理速度，符合预期。
您可以灵活控制函数并发，进而控制异步消息的消费速度。

注意事项

预置并发功能只对已经配置并启动、但未使用的实例收取少量闲置费用，**对于已配置且在使用的实例不收取额外费用**。即只有当前版本的预置数量大于版本并发数时，会收取超出部分的闲置费用。未超出的部分不额外收费。详情见 计费概述 。

在云函数中使用自定义字体

最近更新时间：2023-03-20 17:41:11

应用场景

通过在云函数中使用 Puppeteer，可按需完成针对特定网页的截图、保存、录屏、生成 PDF 等操作。该功能延续了云函数按需拉起的特性，在需要时才去实际启动实例执行，无需使用虚拟机或容器去持续运行服务，方便封装为通用能力。
云函数的运行环境目前仅内置了单一字体，本文提供在云函数中使用自定义字体的最佳实践来解决单一字体不能满足定制化需求的问题。本文以 Node.js 16.13 使用文泉驿正黑体为例，介绍如何在 SCF 中使用自定义字体。

前提条件

需要准备好对应自定义字体文件。如文泉驿正黑体字体文件 WenQuanZhengHei-1.ttf 。

整体流程

1. 将字体文件放在函数代码的指定目录下。
2. 通过设置字体文件配置文件 `fonts.config` ，使其可以加载 1 中指定目录下的字体文件。
3. 通过设置 SCF 环境变量 `XDG_CONFIG_HOME` ，指定字体配置文件的加载路径。

操作步骤

1. 在函数代码根目录下创建文件夹 `fonts` ，将提前准备好的字体文件放在该目录下。
2. 在函数代码根目录下创建文件夹 `fontconfig` ，在该目录下创建字体配置文件 `fonts.conf` ， `fonts.conf` 内容如下：

> **注意**
> 第 27 行中的 `/var/user/fonts` 即为步骤 1 中创建的 fonts 文件夹在 SCF 环境下的路径。

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<!-- /etc/fonts/fonts.conf file to configure system font access -->
<fontconfig>

<!--
    DO NOT EDIT THIS FILE.
    IT WILL BE REPLACED WHEN FONTCONFIG IS UPDATED.
    LOCAL CHANGES BELONG IN 'local.conf'.

    The intent of this standard configuration file is to be adequate for
    most environments.  If you have a reasonably normal environment and
    have found problems with this configuration, they are probably
    things that others will also want fixed.  Please submit any
    problems to the fontconfig bugzilla system located at fontconfig.org

    Note that the normal 'make install' procedure for fontconfig is to
    replace any existing fonts.conf file with the new version.  Place
    any local customizations in local.conf which this file references.

    Keith Packard
-->

<!-- Font directory list -->

    <dir>/usr/share/fonts</dir>
    <dir>/var/user/fonts</dir>
    <dir>/usr/share/X11/fonts/Type1</dir> <dir>/usr/share/X11/fonts/TTF</dir>
<dir>/usr/local/share/fonts</dir>
    <dir prefix="xdg">fonts</dir>
```

```
    <!-- the following element will be removed in the future -->
    <dir>~/.fonts</dir>

<!--
  Accept deprecated 'mono' alias, replacing it with 'monospace'
-->
    <match target="pattern">
        <test qual="any" name="family">
            <string>mono</string>
        </test>
        <edit name="family" mode="assign" binding="same">
            <string>monospace</string>
        </edit>
    </match>

<!--
  Accept alternate 'sans serif' spelling, replacing it with 'sans-serif'
-->
    <match target="pattern">
        <test qual="any" name="family">
            <string>sans serif</string>
        </test>
        <edit name="family" mode="assign" binding="same">
            <string>sans-serif</string>
        </edit>
    </match>

<!--
  Accept deprecated 'sans' alias, replacing it with 'sans-serif'
-->
    <match target="pattern">
        <test qual="any" name="family">
            <string>sans</string>
        </test>
        <edit name="family" mode="assign" binding="same">
            <string>sans-serif</string>
        </edit>
    </match>

<!--
  Load local system customization file
-->
    <include ignore_missing="yes">/etc/fonts/conf.d</include>

<!-- Font cache directory list -->

    <cachedir>/var/cache/fontconfig</cachedir>
    <cachedir prefix="xdg">fontconfig</cachedir>
    <!-- the following element will be removed in the future -->
    <cachedir>~/.fontconfig</cachedir>

    <config>
<!--
  These are the default Unicode chars that are expected to be blank
  in fonts.  All other blank chars are assumed to be broken and
  won't appear in the resulting charsets
 -->
        <blank>
            <int>0x0020</int>   <!-- SPACE -->
            <int>0x00A0</int>   <!-- NO-BREAK SPACE -->
            <int>0x00AD</int>   <!-- SOFT HYPHEN -->
```

```
            <int>0x034F</int>   <!-- COMBINING GRAPHEME JOINER -->
            <int>0x0600</int>   <!-- ARABIC NUMBER SIGN -->
            <int>0x0601</int>   <!-- ARABIC SIGN SANAH -->
            <int>0x0602</int>   <!-- ARABIC FOOTNOTE MARKER -->
            <int>0x0603</int>   <!-- ARABIC SIGN SAFHA -->
            <int>0x06DD</int>   <!-- ARABIC END OF AYAH -->
            <int>0x070F</int>   <!-- SYRIAC ABBREVIATION MARK -->
            <int>0x115F</int>   <!-- HANGUL CHOSEONG FILLER -->
            <int>0x1160</int>   <!-- HANGUL JUNGSEONG FILLER -->
            <int>0x1680</int>   <!-- OGHAM SPACE MARK -->
            <int>0x17B4</int>   <!-- KHMER VOWEL INHERENT AQ -->
            <int>0x17B5</int>   <!-- KHMER VOWEL INHERENT AA -->
            <int>0x180E</int>   <!-- MONGOLIAN VOWEL SEPARATOR -->
            <int>0x2000</int>   <!-- EN QUAD -->
            <int>0x2001</int>   <!-- EM QUAD -->
            <int>0x2002</int>   <!-- EN SPACE -->
            <int>0x2003</int>   <!-- EM SPACE -->
            <int>0x2004</int>   <!-- THREE-PER-EM SPACE -->
            <int>0x2005</int>   <!-- FOUR-PER-EM SPACE -->
            <int>0x2006</int>   <!-- SIX-PER-EM SPACE -->
            <int>0x2007</int>   <!-- FIGURE SPACE -->
            <int>0x2008</int>   <!-- PUNCTUATION SPACE -->
            <int>0x2009</int>   <!-- THIN SPACE -->
            <int>0x200A</int>   <!-- HAIR SPACE -->
            <int>0x200B</int>   <!-- ZERO WIDTH SPACE -->
            <int>0x200C</int>   <!-- ZERO WIDTH NON-JOINER -->
            <int>0x200D</int>   <!-- ZERO WIDTH JOINER -->
            <int>0x200E</int>   <!-- LEFT-TO-RIGHT MARK -->
            <int>0x200F</int>   <!-- RIGHT-TO-LEFT MARK -->
            <int>0x2028</int>   <!-- LINE SEPARATOR -->
            <int>0x2029</int>   <!-- PARAGRAPH SEPARATOR -->
            <int>0x202A</int>   <!-- LEFT-TO-RIGHT EMBEDDING -->
            <int>0x202B</int>   <!-- RIGHT-TO-LEFT EMBEDDING -->
            <int>0x202C</int>   <!-- POP DIRECTIONAL FORMATTING -->
            <int>0x202D</int>   <!-- LEFT-TO-RIGHT OVERRIDE -->
            <int>0x202E</int>   <!-- RIGHT-TO-LEFT OVERRIDE -->
            <int>0x202F</int>   <!-- NARROW NO-BREAK SPACE -->
            <int>0x205F</int>   <!-- MEDIUM MATHEMATICAL SPACE -->
            <int>0x2060</int>   <!-- WORD JOINER -->
            <int>0x2061</int>   <!-- FUNCTION APPLICATION -->
            <int>0x2062</int>   <!-- INVISIBLE TIMES -->
            <int>0x2063</int>   <!-- INVISIBLE SEPARATOR -->
            <int>0x206A</int>   <!-- INHIBIT SYMMETRIC SWAPPING -->
            <int>0x206B</int>   <!-- ACTIVATE SYMMETRIC SWAPPING -->
            <int>0x206C</int>   <!-- INHIBIT ARABIC FORM SHAPING -->
            <int>0x206D</int>   <!-- ACTIVATE ARABIC FORM SHAPING -->
            <int>0x206E</int>   <!-- NATIONAL DIGIT SHAPES -->
            <int>0x206F</int>   <!-- NOMINAL DIGIT SHAPES -->
            <int>0x2800</int>   <!-- BRAILLE PATTERN BLANK -->
            <int>0x3000</int>   <!-- IDEOGRAPHIC SPACE -->
            <int>0x3164</int>   <!-- HANGUL FILLER -->
            <int>0xFEFF</int>   <!-- ZERO WIDTH NO-BREAK SPACE -->
            <int>0xFFA0</int>   <!-- HALFWIDTH HANGUL FILLER -->
            <int>0xFFF9</int>   <!-- INTERLINEAR ANNOTATION ANCHOR -->
            <int>0xFFFA</int>   <!-- INTERLINEAR ANNOTATION SEPARATOR -->
            <int>0xFFFB</int>   <!-- INTERLINEAR ANNOTATION TERMINATOR -->
        </blank>
<!--
  Rescan configuration every 30 seconds when FcFontSetList is called
 -->
        <rescan>
```

```
            <int>30</int>
        </rescan>
    </config>

</fontconfig>
```

步骤 1、2 完成后，函数代码目录结构如下：

```
├── 函数代码文件（如 index.js、app.js）
├── fonts
|   └── WenQuanZhengHei-1.ttf
├── fontconfig
|   └── fonts.conf
```

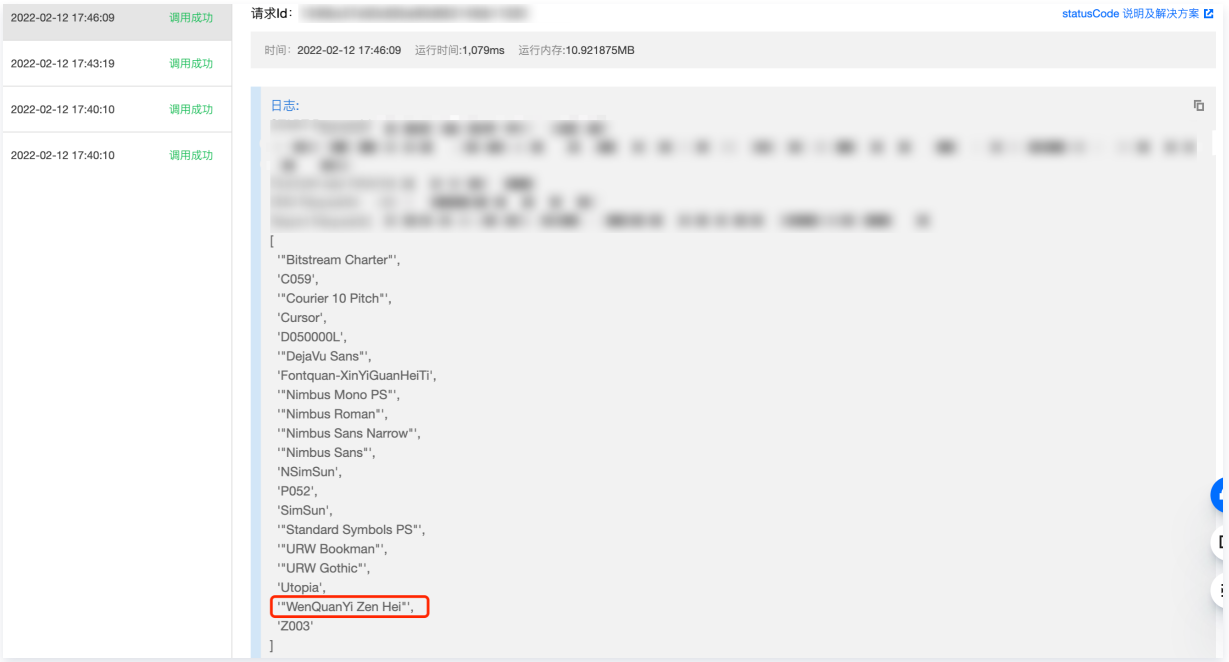
3. 将函数代码打包为 zip 格式，选择“本地上传 zip 包”，创建函数或者更新函数代码，上传完成后单击**部署**完成云端函数代码的更新。

> **注意**
> 函数代码打包时是将代码根目录下全部文件进行打包，即上述文件结构中的全部文件，不需要打包外层文件夹。

4. 编辑函数环境变量，添加环境变量后，单击**保存**完成配置更新。
   - key：`XDG_CONFIG_HOME`
   - value：`/var/user`

5. 验证字体是否添加成功。Node.js 环境下，使用 `font-list` 库，可以打印出环境中支持的字体。完成上述配置后，可以看到环境中已经成功加载了文泉驿正黑体。

```
var fontList = require('font-list')
console.log(await fontList.getFonts()) //打印环境中支持的字体
```



镜像部署函数也可以采用如下方式实现自定义字体。以在镜像内包装 Chrome、Puppeteer 以及自身所需的字体文件，构筑成镜像，并使用此镜像来部署函数为例。操作步骤如下：

1. 编写 dockerfile

下文向您提供一个简单的、通过 alpine 基础镜像来构筑包含了 chrome 和 puppeteer 的镜像构建文件，文件可以命名为 `mypuppeteer.dockerfile` 。示例如下：

```
FROM alpine

# Installs latest Chromium (92) package.
RUN apk add --no-cache \
      chromium \
      nss \
      freetype \
      harfbuzz \
      ca-certificates \
      ttf-freefont \
      nodejs \
      yarn


# Tell Puppeteer to skip installing Chrome. We'll be using the installed package.
ENV PUPPETEER_SKIP_CHROMIUM_DOWNLOAD=true \
    PUPPETEER_EXECUTABLE_PATH=/usr/bin/chromium-browser

# Puppeteer v10.0.0 works with Chromium 92.
RUN yarn add puppeteer@10.0.0

# 添加 cjk 字体以支持中文
COPY NotoSansCJK.ttc  /usr/share/fonts/TTF
```

注意

- 代码中所使用的 字体文件，需要下载放置到 docker file 所在的相同目录。
- 本文仅提供示例，您可以根据实际情况选择字体文件，调整文件名及 docker file 中对应字段。

2. 镜像构建

通过如下命令可以完成镜像构建，将构建出名称为 `mypuppeteer:v1` 的镜像：

```
docker build -t mypuppeteer:v1 -f mypuppeteer.dockerfile .
```

3. 本地测试

本地镜像构建完成后，可以使用如下 `test.js` 脚本快速测试验证效果。您可通过脚本针对网页截图并保存。

```
const puppeteer = require('puppeteer');
(async () => {
  const browser = await puppeteer.launch({
    executablePath: '/usr/bin/chromium-browser',
    args: ['--no-sandbox','--disable-setuid-sandbox','--ignore-certificate-errors'],
    defaultViewport: {
        width: 1920,
        height: 1080,
        deviceScaleFactor: 3,
      },
  });
  const page = await browser.newPage();
  await page.goto('https://www.baidu.com');
  await page.screenshot({path: '/home/test.png'});
  await browser.close();
})();
```

执行以下命令运行镜像，将测试脚本目录挂载到容器内并运行，同时截屏文件也将生成在此目录下。

```
docker run -it --rm -v $(pwd):/home mypuppeteer:v1 node /home/test.js
```

您可以通过查看是否输出截屏文件，以验证字体文件是否生效。

4. 后续操作

可以运行 chrome、puppeteer 的镜像构建完成后，您可以基于此镜像之上构建函数运行环境，并通过将镜像 push 到腾讯云的镜像仓库，利用云函数的自定义镜像部署能力构筑自身所需的业务。
云函数的自定义镜像部署的说明及使用方法详情见 使用镜像部署函数功能说明 。

常见问题

报错 “font-list 模块找不到”怎么处理?

您可以按以下步骤安装相关模块：

```
cd src
```

```
npm install font-list
```

SpringBoot + SCF 实现待办应用

最近更新时间：2023-03-17 11:29:42

操作场景

Spring Boot 是由 Pivotal 团队提供的框架，用来简化新 Spring 应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。
本文将介绍如何通过 SCF 使用 SpringBoot 搭建一个待办应用。SCF 提供事件触发的 事件函数 和 HTTP 请求触发的 Web 函数 两种函数类型，在 SpringBoot 场景下推荐使用 Web 函数。

前提条件

请参考 云函数 JAVA 开发指南 准备开发环境和工具。

操作步骤

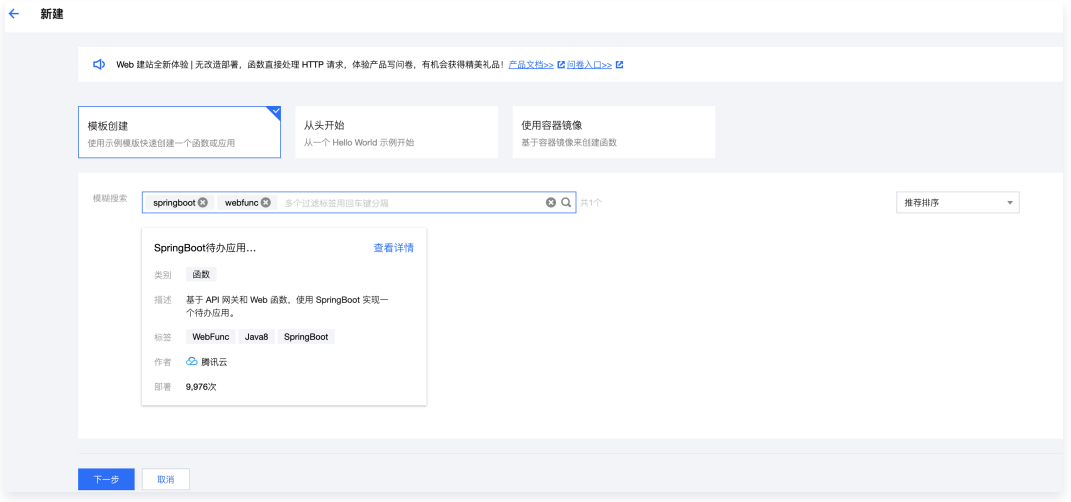
使用 Web 函数

SCF 提供模板函数，按照如下流程操作可使用 Web 函数快速创建一个待办应用并体验待办事项的增删改查功能。

> **注意**
> 本模板仅作为示例提供，待办事项数据实际存储在实例缓存中，不作为持久化存储。

创建函数

1. 登录 Serverless 控制台。
2. 在**函数服务**页中单击**新建**。
3. 在**新建**页中，选择**模板创建**，并搜索关键词 `springboot` 、 `webfunc` 。在查询结果中选择 **SpringBoot 待办应用**并单击**下一步**。如下图所示：

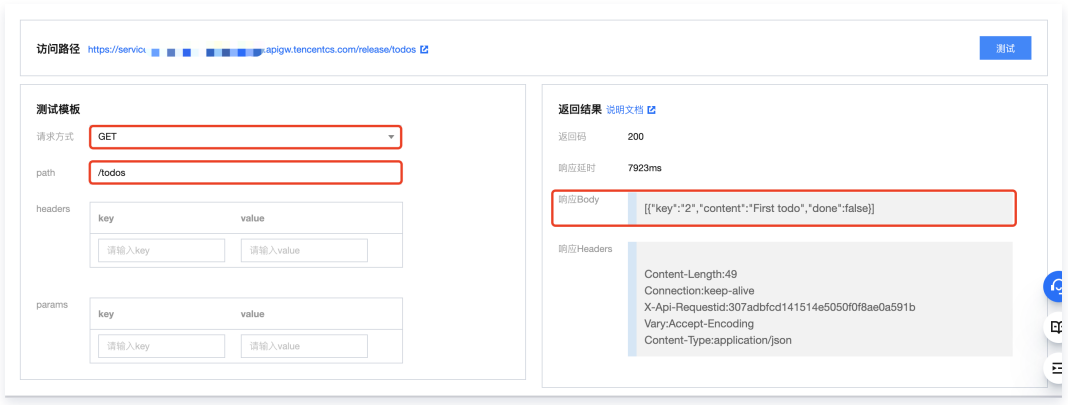


4. 保持默认配置，单击**完成**，完成函数创建。

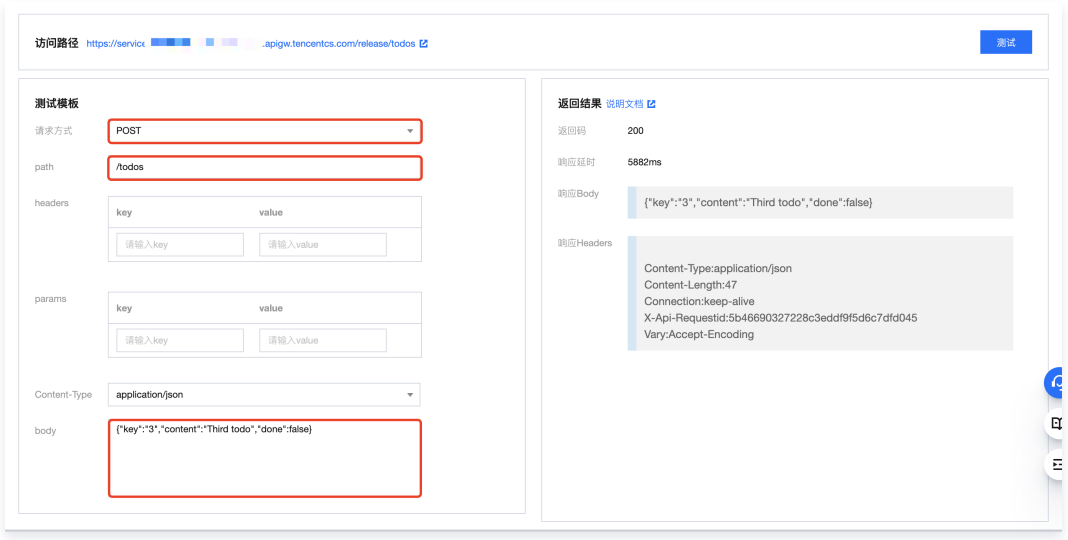
测试函数

在**函数代码**页签，按照如下流程操作，通过测试模板发起模拟请求体验待办应用增删改查功能：

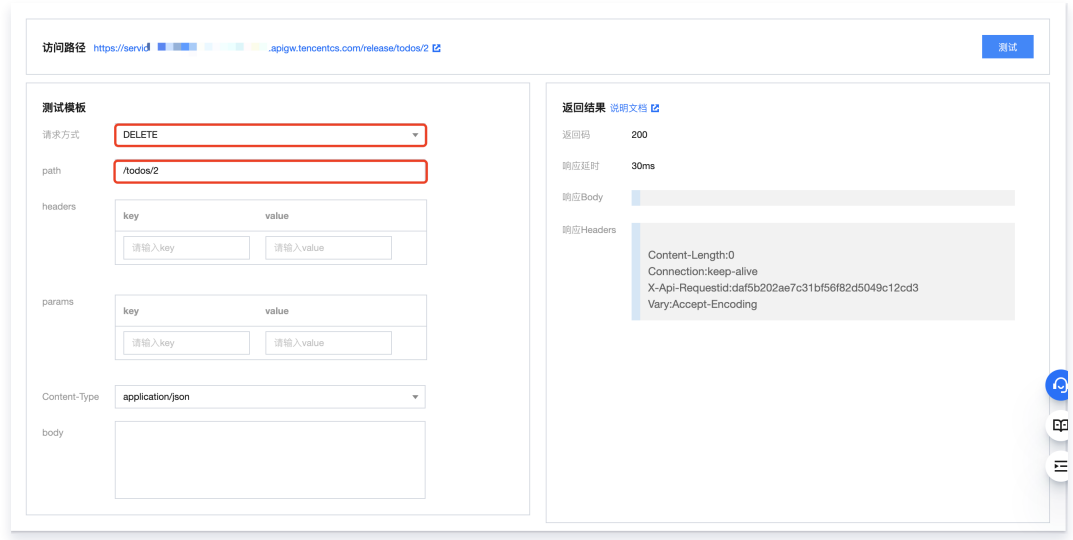
- 查询待办列表：
  请求方式选择 GET, path 填写 `/todos` ，单击**测试**后，在响应 Body 中可以查看到当前的待办事项。如下图所示：



- **增加待办事项：**
  请求方式选择 POST，path 填写 `/todos`，body 填写 `{"key":"3","content":"Third todo","done": false}`，单击测试增加一个待办事项。如下图所示：

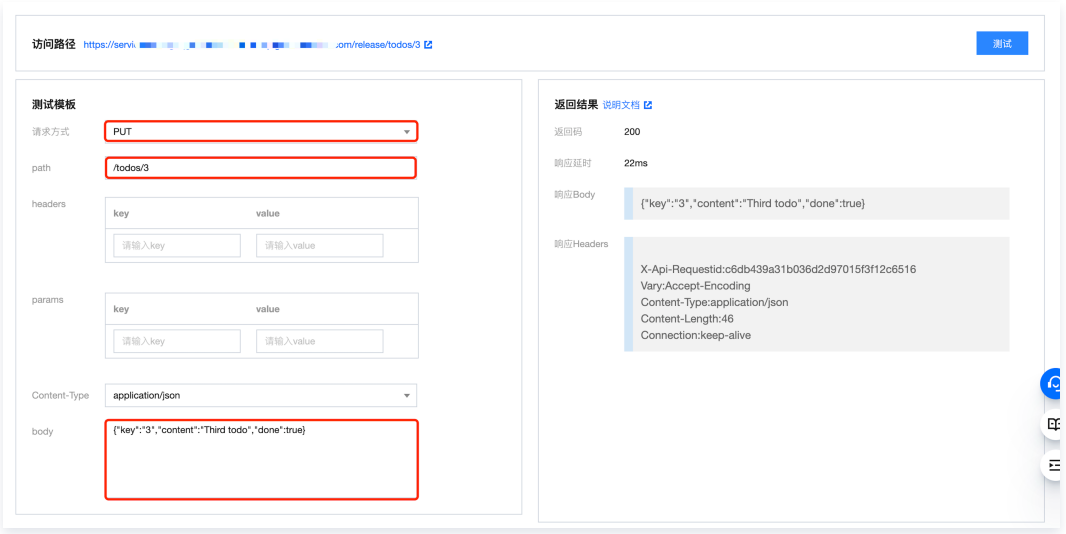


- **删除待办事项：**
  请求方式选择 DELETE，以删除 key 为 2 的待办事项为例，path 填写 `/todos/2`，单击测试。如下图所示：



- **修改待办事项：**
  请求方式选择 PUT，以将 key 为 3 的待办事项由未完成改为完成为例，path 填写 `/todos/3`，body 填写

`{"key":"3","content":"Third todo","done": true}`，单击**测试**。如下图所示：

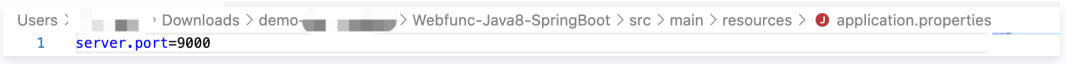


代码示例

在 创建函数 步骤中，您也可以根据业务需求修改函数模板。在**模板选择**页面，单击模板卡片右上角的**查看详情**，在展开的页面中单击**下载模板函数**即可获取模板函数源码。

原生 SpringBoot 项目迁移到 Web 函数需要执行如下步骤：

1. 确保 Spring 监听端口为 9000（SCF Web 函数指定监听端口）。



2. 编译 JAR 包。
   下载代码之后，在目录 Webfunc-Java8-SpringBoot 下运行编译命令：

```
gradle build
```

   编译完成后可在 `build/libs` 目录下获取到打包完成的 jar 包，选择后缀为 `-all` 的 jar 包。
3. 准备一个可执行文件 scf_bootstrap 用于启动 Web Server，文件内容可参考下文：

```
#!/bin/bash
/var/lang/java8/bin/java -Dserver.port=9000 -jar scf-springboot-java8-0.0.2-SNAPSHOT-all.jar
```

> **注意**
> 在 scf_bootstrap 文件所在目录执行 `chmod 755 scf_bootstrap` 来保证 scf_bootstrap 文件具有可执行权限。

4. 将 `scf_bootstrap` 文件与生成的 jar 包一起打包为 zip 部署到云函数。部署函数步骤如下：
   4.1 登录 Serverless 控制台 。
   4.2 在**函数服务**页中单击**新建**。
   4.3 在**新建**页中，选择**从头开始**。参考以下内容进行配置：
   - **函数类型**：web 函数
   - **运行环境**：Java8
   - **提交方法**：本地上传 zip 包
   - **函数代码**：单击上传选择打包好的 zip 文件

4.4 其他保持默认配置，单击**完成**即可完成函数创建。如下图所示：

使用事件函数

SCF 提供模板函数，按照如下流程操作可使用事件函数快速创建一个待办应用并体验待办事项的增删改查功能。

> **注意**
> 本模板仅作为示例提供，待办事项数据实际存储在实例缓存中，不作为持久化存储。

创建函数

1. 登录 Serverless 控制台 。
2. 在**函数服务**页中单击**新建**。

3. 在**新建**页中，选择**模板创建**，并搜索关键词 `springboot` 。在查询结果中选择 **SpringBoot** 并单击**下一步**。

4. 保持默认配置，单击**完成**，完成函数创建。

创建触发器

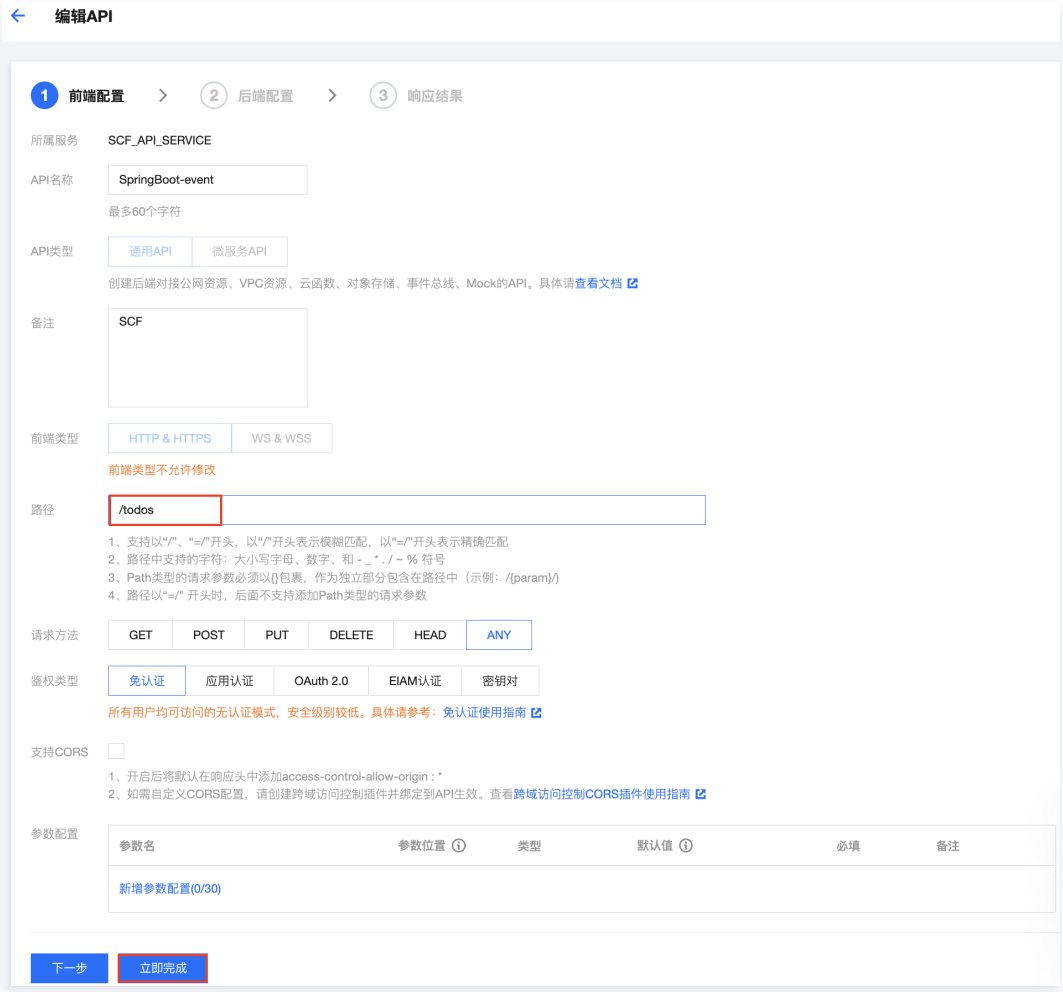
> **注意**
> 如果在创建函数过程中已经创建好 API 网关触发器，核对已有触发器与下文配置一致即可。

1. 函数创建完成后，在**触发管理**页签，单击**创建触发器**。

2. 在弹窗中进行触发器配置。参考以下内容进行选择，其余保持默认配置，单击**提交**。
   - 触发方式：API 网关触发
   - 集成响应：启用
3. 创建完成后需要调整 API 网关触发器的参数，单击 **API 服务名**跳转到 API 网关控制台进行下一步操作。如下图所示：

4. 在 API 网关控制台找到函数使用的 API，单击**编辑**。如下图所示：

5. 修改前端配置页面中的路径为 `/todos`，单击**立即完成**，并按照引导发布服务。如下图所示：

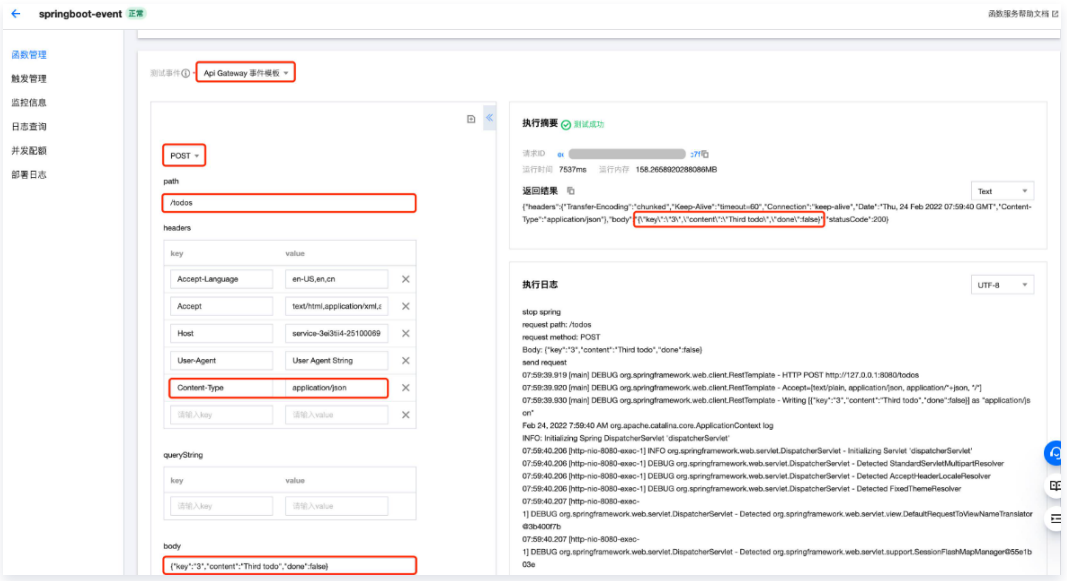


测试函数

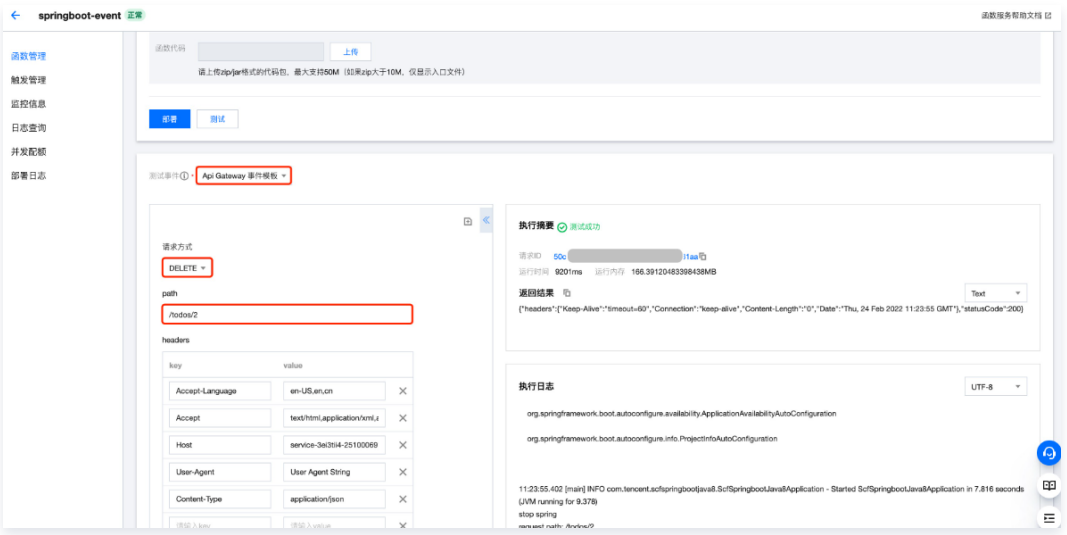
在**函数代码**页签，按照如下流程操作，通过 Api Gateway 事件模板发起模拟请求体验待办应用增删改查功能：

- 查询待办列表：
  请求方式选择 GET, path 填写 `/todos`，单击**测试**后，在响应 Body 中可以查看到当前的待办事项。如下图所示：

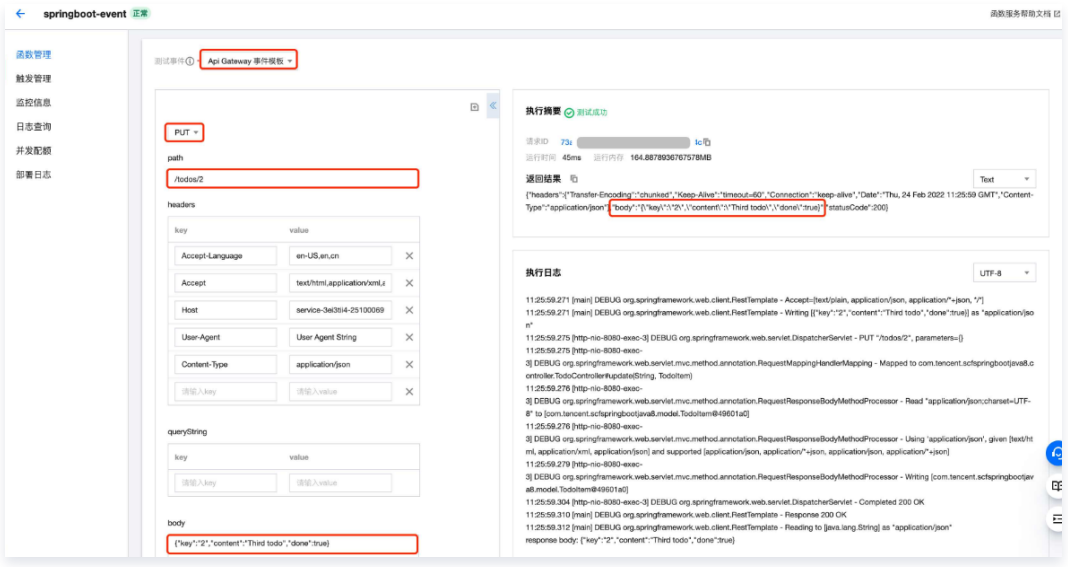
- 增加待办事项：
  请求方式选择 POST, path 填写 `/todos`，headers 填写 `Content-Type: application/json`，body 填写 `{"key":"3","content":"Third todo","done": false}`，单击**测试**增加一个待办事项。如下图所示：



- **删除待办事项：**
  请求方式选择 DELETE，以删除 key 为 2 的待办事项为例，path 填写 `/todos/2`，单击测试。如下图所示：



- **修改待办事项：**
  请求方式选择 PUT，以将 key 为 3 的待办事项由未完成改为完成为例，path 填写 `/todos/2`，body 填写 `{"key":"2","content":"Third todo","done": true}`，单击测试。如下图所示：



代码示例

在 创建函数 步骤中，您也可以根据业务需求修改函数模板。在**模板选择**页面，单击模板卡片右上角的**查看详情**，在展开的页面中单击**单击下载模板函数**即可获取模板函数源码。
可参考以下流程操作：

1. 增加一个 `ScfHandler` 类， `ScfHandler` 类主要用于接收事件触发，并转发消息给 Spring 服务，函数接收到 Spring 服务的返回后再把结果返回给调用方。 `ScfHandler` 类增加后项目结构如下：

```
.
├── src
│   └── main
│       ├── java
│       │   └── com.tencent.scfspringbootjava8
│       │       ├── controller
│       │       ├── model
│       │       └── repository
│       │           ├── ScfHandler.java
│       │           └── ScfSpringbootJava8Application.java
│       └── resources
```

2. 编译 JAR 包
   下载代码之后，在根目录下运行编译命令：

```
gradle build
```

   编译完成后可在 `build/libs` 目录下获取到打包完成的 jar 包，选择后缀为 `-all` 的 jar 包。
3. 将编译生成的 jar 包部署到云函数。部署函数步骤如下：
   3.1 登录 Serverless 控制台 。
   3.2 在**函数服务页**中单击**新建**。
   3.3 在**新建**页中，选择**从头开始**。参考以下内容进行配置：
   - **函数类型**：事件函数
   - **运行环境**：Java8
   - **提交方法**：本地上传 zip 包
   - **执行方法**：com.tencent.scfspringbootjava8.ScfHandler:: mainHandler
   - **函数代码**：单击上传选择打包好的 zip 文件

3.4 其他保持默认配置，单击**完成**即可完成函数创建。如下图所示：

Serverless Cloud Framework 部署静态网站

最近更新时间：2024-03-29 10:21:21

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，方便的进行多人协作开发。
- **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 Vue + Express + PostgreSQL 全栈网站

最近更新时间：2024-12-25 18:32:21

操作场景

该模板可以快速部署一个基于 Vue + Express + PostgreSQL 的全栈 Serverless 应用。主要包含以下组件：

- Serverless RESTful API：通过**云函数**和 **API 网关**构建的 Express 框架实现 RESTful API。
- Serverless 静态网站：前端通过托管 Vue.js 静态页面到 **COS 对象存储**中。
- PostgreSQL Serverless：通过创建 **PostgreSQL DB** 为全栈网站提供数据库服务。
- VPC：通过创建 **VPC** 和 **子网**，提供 SCF 云函数和数据库的网络打通和使用。

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**）
- 账号已经配置 **QcloudPostgreSQLFullAccess** 策略，配置方法请参见 账号和权限配置 。

操作步骤

安装

通过 npm 全局安装 Serverless Cloud Framework，详情请参见 安装 Serverless Cloud Framework 。

配置

1. 新建一个本地文件夹，使用 `scf init` 命令，下载相关 template。

```
scf init fullstack
```

2. 在项目根目录下新建 .env 文件，并在其中配置对应的腾讯云 SecretId、SecretKey、地域和可用区信息。

```
# .env
TENCENT_SECRET_ID=xxx  // 您账号的 SecretId
TENCENT_SECRET_KEY=xxx // 您账号的 SecretKey

# 地域可用区配置
REGION=ap-guangzhou //资源部署区，该项目中指云函数与静态页面部署区
ZONE=ap-guangzhou-2 //资源部署可用区 ，该项目中指 DB 部署所在的可用区
```

> **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，请保证您的账号已经授权了 AdministratorAccess 权限。您可以在 API 密钥管理 中获取 SecretId 和 SecretKey。
> - ZONE 目前只支持 ap-beijing-3 、ap-guangzhou-2、ap-shanghai-2。

3. 在 fullstack 目录下执行以下命令，安装所需依赖：

```
npm run bootstrap
```

部署

1. 执行 `scf deploy --all` 命令进行部署。返回信息如下所示：

```
$ scf deploy --all

serverless-cloud-framework

serverlessVpc:
```

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

fullstackDB:
  region:         ap-guangzhou
  zone:           ap-guangzhou-2
  vpcConfig:
    subnetId: subnet-100000
    vpcId:    vpc-1000000
  dBInstanceName: fullstackDB
  dBInstanceId:   postgres-100000
  private:
    connectionString: postgresql://tencentdb_100000xxxxxxxxxxxxx@172.16.250.15:5432/tencentdb_1000000
    host:             172.16.250.15
    port:             5432
    user:             tencentdb_100000
    password:         xxxxxxxx
    dbname:           tencentdb_100000

fullstack-api:
  region: ap-guangzhou
  apigw:
    serviceId:   service-100000
    subDomain:   service-100000-123456789.gz.apigw.tencentcs.com
    environment: release
    url:         https://service-100000-123456789.gz.apigw.tencentcs.com/release/
  scf:
    functionName: fullstack-api
    runtime:      Nodejs10.15
    namespace:    default

fullstack-frontend:
  website: https://fullstack-serverless-db-123456789.cos-website.ap-guangzhou.myqcloud.com

50s › tencent-fullstack › Success
```

部署成功后，您可以使用浏览器访问项目产生的 website 链接，即可看到生成的网站。

> **说明：**
> 本项目云函数因 VPC，导致无法直接访问外网，如需访问外网请参见 云函数网络配置 。

2. 执行 `scf remove --all` 命令，可移除项目。返回信息如下所示：

> **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 --all

serverless-cloud-framework

38s › tencent-fullstack › Success
```

部署流式转码应用

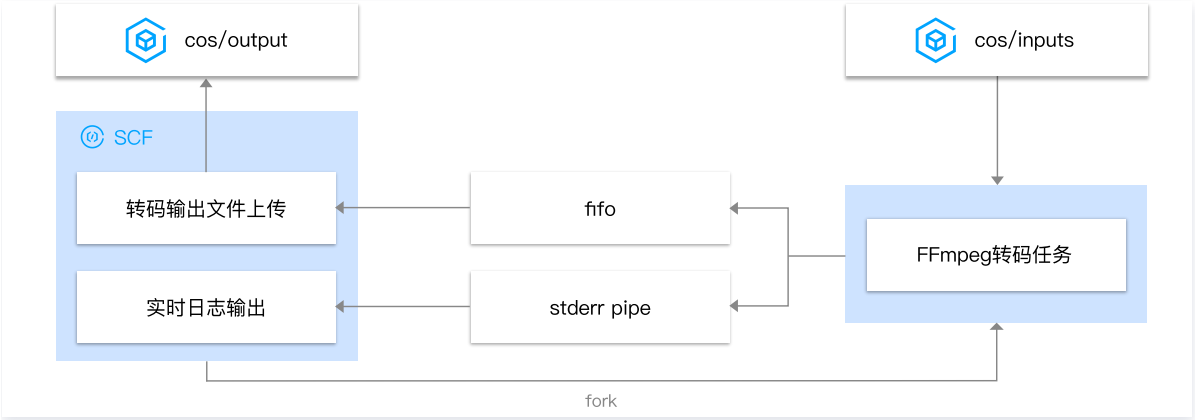
最近更新时间：2025-02-26 16:10:43

应用简介

通过结合 COS、云函数、CLS 和 FFmpeg，您可以快速构建高可用、并行处理、实时日志、高度自定义的视频转码服务。

架构原理

通过云函数创建 FFmpeg 任务进程，云函数进程与 FFmpeg 任务进程通过 pipe 和 FIFO 的方式进行数据传输。云函数进程中的两个任务线程分别接收 FFmpeg 任务进程向函数进程输出的 FFmpeg 日志流与转码后的文件流。实时日志线程将日志流输出，上传任务负责缓存文件流并上传至用户定义的输出 COS。



应用优势

- **流式转码**
  采用流式拉取源视频文件，流式上传转码文件的工作方式，突破了本地存储的限制，且不需要额外部署 CFS 等产品。
- **实时日志**
  视频转码过程中，可通过 CLS 日志实时查看转码进度。同时支持输出 FFmpeg 应用的完整日志。
- **长时运行**
  利用云函数的 长时运行机制，支持 12h-24h 的运行时长，可覆盖大文件耗时较长的转码场景。
- **自定义参数**
  支持用户自定义配置 FFmpeg 命令参数。

应用资源

转码应用部署后，将为您创建以下资源：

- **云函数**：流式读取 COS 文件，使用 FFmpeg 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。
- **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，详情参考 CLS 计费规则 。

注意事项

- 转码应用需要依赖云函数长时运行能力，详情请参考 异步执行 。
- 转码输出桶与函数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。
- 需要为转码应用的云函数创建运行角色，并授权cos读写权限。详细配置参考 运行角色 。
- FFmpeg 不同转码场景下指令配置参数不同，因此需要您具有一定的 FFmpeg 使用经验。本文中仅提供几个样例作为参考，更多 FFmpeg 指令参考 FFmpeg 官网 。

前提条件

1. 安装 Serverless Cloud Framework 。
2. 配置部署账号权限。详情见 账号和权限配置 。
3. 配置 运行角色 权限。

操作步骤

1. 下载转码应用

```
scf init transcode-app
```

进入项目目录 `transcode-app`，将看到目录结构如下：

```
transcode-app
|- .env  #环境配置
|- serverless.yml # 应用配置
|- log/ #log 日志配置
|   └── serverless.yml
└──transcode/  #转码函数配置
       |- src/
       |    |- ffmpeg   #转码 FFmpeg 工具
       |    └── index.py
       └── serverless.yml
```

- `log/serverless.yml` 定义一个 CLS 日志集和主题，用于转码过程输出的日志保存，目前采用腾讯云 CLS 日志存储。每个转码应用将会根据配置的 CLS 日志集和主题去创建相关资源，CLS 的使用会产生计费，具体参考 CLS 计费规则 。
- `transcode/serverless.yml` 定义函数的基础配置及转码参数配置。
- `transcode/src/index.py` 转码功能实现。
- `transcode/src/ffmpeg` 转码工具 FFmpeg。

2. 配置环境变量和应用参数

- 应用参数，文件 `transcode-app/serverless.yml`：

```
#应用信息
app: transcodeApp # 您需要配置成您的应用名称
stage: dev # 环境名称，默认为dev
```

- 环境变量，文件 `transcode-app/.env`：

```
REGION=ap-shanghai  # 应用创建所在区
TENCENT_SECRET_ID=xxxxxxxxxxxx # 您的腾讯云 secretId
TENCENT_SECRET_KEY=xxxxxxxxxxxx # 您的腾讯云 secretKey
```

> **说明：**
> - 您可以登录腾讯云控制台，可以在 API 密钥管理 中获取 SecretId 和 SecretKey。
> - 如果您的账号为主账号，或者子账号具有扫码权限，也可以不配置 SecretId 与 SecretKey，直接扫码部署应用。更多详情参考 账号和权限配置 。

3. 配置转码需要的参数信息

- CLS 日志定义，文件 `transcode-app/log/serverless.yml`：

```
#组件信息全量配置参考 https://github.com/serverless-components/tencent-cls/blob/master/docs/configure.md
component: cls # 引用 component 的名称
name: cls-video # 创建的实例名称，请修改成您的实例名称

# 组件参数
inputs:
  name: cls-log # 您需要配置一个 name，作为您的 CLS 日志集名称
  topic: video-log # 您需要配置一个 topic，作为您的 CLS 日志主题名称
  region: ${env:REGION} # 区域，统一在环境变量中定义
  period: 7 # 日志保存时间，单位天
```

- 云函数及转码配置，文件 `transcode-app/transcode/serverless.yml`：

```
#组件信息 全量配置参考https://github.com/serverless-components/tencent-scf/blob/master/docs/configure.md
component: scf # 引用 component 的名称
name: transcode-video # 创建的实例名称，请修改成您的实例名称

# 组件参数
inputs:
  name: transcode-video-${app}-${stage}
  src: ./src
  handler: index.main_handler
  role: transcodeRole # 函数运行角色，已授予 COS 对应桶全读写权限
  runtime: Python3.6
  memorySize: 3072 # 内存大小，单位 MB
  timeout: 43200 # 函数执行超时时间，单位秒，即本 demo 目前最大支持 12h 运行时长
  region: ${env:REGION} # 函数区域，统一在环境变量中定义
  asyncRunEnable: true # 开启长时运行
  cls: # 函数日志
    logsetId: ${output:${stage}:${app}:cls-video.logsetId} # CLS 日志集，cls-video 为 CLS 组件的实例名称
    topicId: ${output:${stage}:${app}:cls-video.topicId} # CLS 日志主题
  environment:
    variables: # 转码参数
      REGION: ${env:REGION} # 输出桶区域
      DST_BUCKET: test-123456789 # 输出桶名称
      DST_PATH: video/outputs/ # 输出桶路径
      DST_FORMAT: avi # 转码生成格式
      FFMPEG_CMD: ffmpeg -i {input} -y -f {dst_format} {output} # 转码基础命令，您可自定义配置，但必须包含
ffmpeg 配置参数和格式化部分，否则会造成转码任务失败。
      FFMPEG_DEBUG: 1 # 是否输出 ffmpeg 日志，0 为不输出，1 为输出
      TZ: Asia/Shanghai # CLS 日志输出时间的时区
  events:
    - cos: # COS 触发器
        parameters:
          bucket: test-123456789.cos.ap-shanghai.myqcloud.com # 输入文件桶
          filter:
            prefix: video/inputs/ # 桶内路径
          events: 'cos:ObjectCreated:*' # 触发事件
        enable: true
```

说明：

- 输出桶与函数建议配置在同一区域，跨区域配置应用稳定性及效率都会降低，并且会产生跨区流量费用。
- 内存大小上限为3072MB，运行时长上限为43200s。如需调整，请通过 在线咨询 申请配额调整。
- 转码应用必须开启函数长时运行 asyncRunEnable: true。
- 运行角色请根据 运行角色 创建并授权。
- 示例配置的 FFmpeg 指令仅适用于 AVI 转码场景，详细介绍参考 FFmpeg 指令 。

4. 部署项目

在 `transcode-app` 项目目录下，执行 `scf deploy` 部署项目。

```
cd transcode-app && scf deploy
```

5. 上传视频文件

上传视频文件到已经配置好的cos桶指定路径，则会自动转码。本示例中是 COS 桶 `test-123456789.cos.ap-shanghai.myqcloud.com` 下的 `/video/inputs/` 。
转码成功后，文件将保存在您配置的输出桶路径中。本示例中是 COS 桶 `test-123456789.cos.ap-shanghai.myqcloud.com` 下的 `/video/outputs/` 。

6. 重新部署

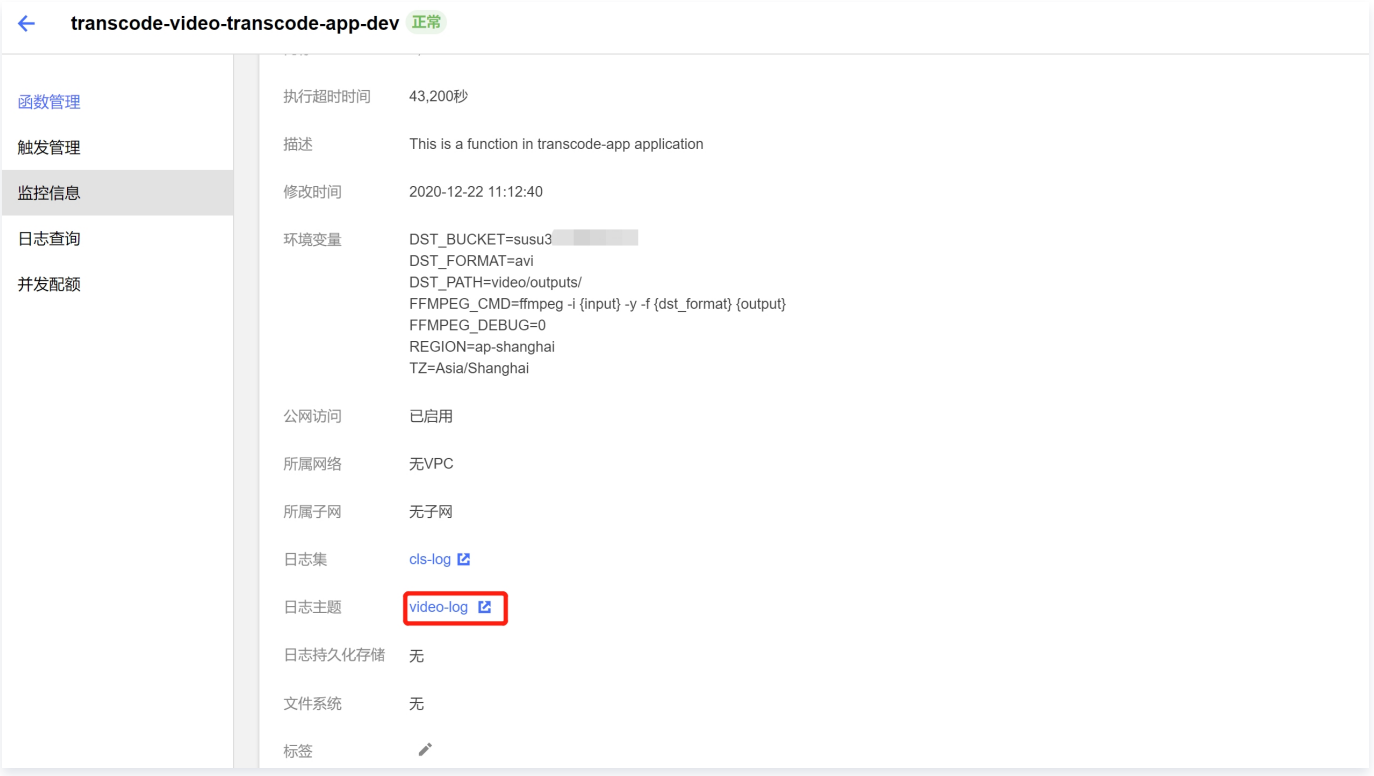
如果需要调整转码配置，修改文件 `transcode/serverless.yml` 后，重新部署云函数即可：

```
cd transcode && scf deploy
```

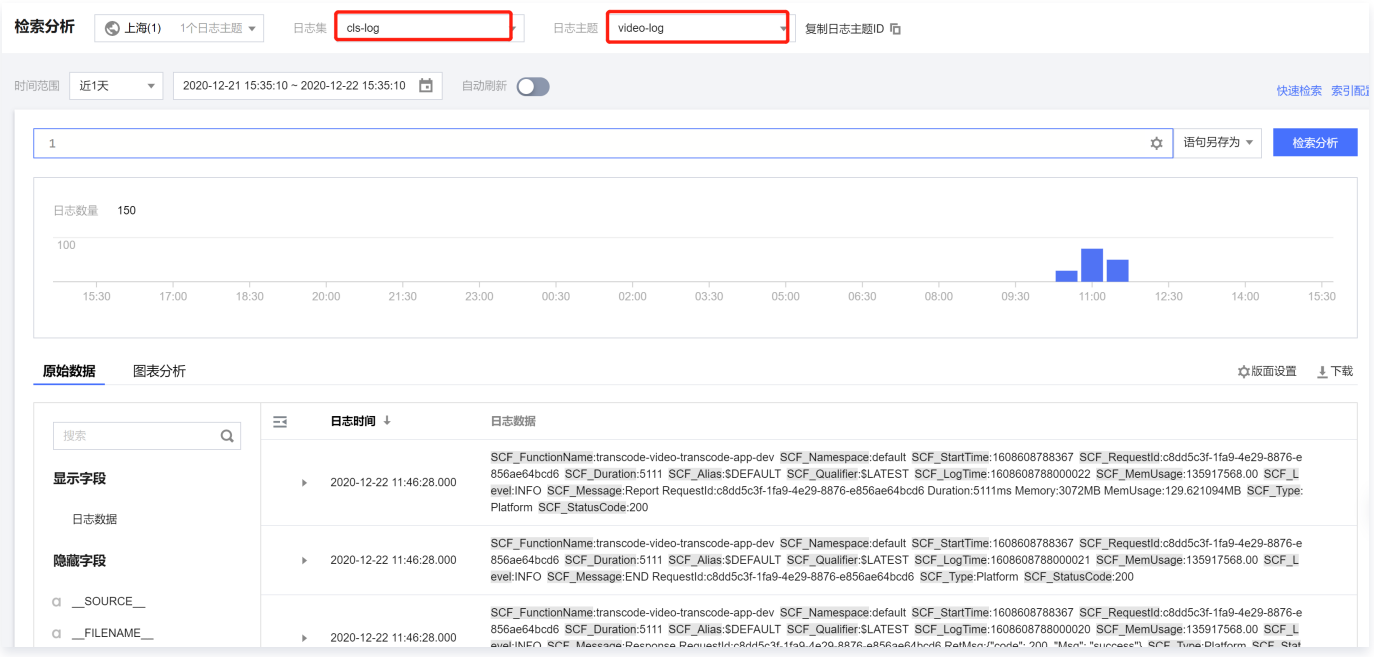
监控与日志

批量文件上传到 COS 会并行触发转码执行。

1. 登录 Serverless 控制台 的**函数服务**页面中，单击函数名进入函数管理页面。
2. 单击**日志查询**，即可查看日志监控。
3. 单击**函数管理** > **函数配置**，单击日志主题的链接，跳转至日志服务控制台。



4. 在日志服务控制台的**检索分析**页面中，选择日志集合日志主题，即可查看日志检索分析 。



FFmpeg 工具

FFmpeg 指令

yml 文件 `transcode-app/transcode/serverless.yml` 中 `DST_FORMAT` 与 `FFMPEG_CMD` 指定了转码应用的转码指令，您可根据应用场景自定义配置。
例如，转码 MP4 格式视频，可以将 FFMPEG_CMD 配置如下：

```
DST_FORMAT: mp4
FFMPEG_CMD: ffmpeg -i {input} -vcodec copy -y -f {dst_format} -movflags frag_keyframe+empty_moov {output}
```

说明：
- FFMPEG_CMD 必须包含 FFmpeg 配置参数和格式化部分，否则会造成转码任务失败。
- FFmpeg 不同转码场景下指令配置参数不同，因此需要您具有一定的 FFmpeg 使用经验。以上提供的指令仅是针对这几个应用场景的指令。更多 FFmpeg 指令参考 FFmpeg 官网 。

自定义 FFmpeg

转码应用场景中提供了默认的 FFmpeg 工具，如果您想自定义 FFmpeg，执行以下操作：
1. 将样例中的 FFmpeg 替换成您自定义的 FFmpeg。
2. 在 `transcode-app/transcode` 目录下再次执行 `scf deploy` 部署更新。

```
cd transcode && scf deploy
```

说明：
自行编译的 FFmpeg 环境与云函数运行环境如果不同，可能会导致 FFmpeg 权限问题。我们提供了云函数执行环境的官方镜像，请使用 官方镜像环境 编译您的 FFmpeg。

运行角色

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，因此需要给函数配置一个授权 COS 全读写的运行角色。更多参考 函数运行角色 。
1. 登录 访问管理控制台 ，单击**新建角色**。
2. 角色载体选择**腾讯云产品服务**。
3. 在“输入角色载体信息”步骤中勾选**云函数（scf）**，并单击**下一步**。
4. 在“配置角色策略”步骤中，您可以直接选择 `QcloudCOSFullAccess` 对象存储（COS）全读写访问权限，如果需要更细粒度的权限配置，请根据实际情况配置选择。单击**下一步**。
5. 在“配置角色标签”步骤中，设置不同维度的标签，使用如职位、部门、籍贯等，标签对用户进行分类管理。
6. 在“审阅”步骤中，输入角色名称，完成创建角色及授权。该角色将作为函数的运行角色，配置在文件 `transcode-app/transcode/serverless.yml` 中。

说明：
由于运行角色密钥有效期为8~12小时，因此函数配置的超时时间超过8小时，临时密钥即有过期风险，最长不能大于12小时。如果您需要更长的函数执行时长，可以通过改造 `transcode-app/transcode/src/index.py` 中的访问 COS 方式，配置永久密钥去读写访问 COS。但这样会暴露您的密钥在代码中，请谨慎使用。

API 网关 APIGW
SCF + API 网关提供 API 服务
示例说明

最近更新时间：2024-11-15 17:24:13

本教程假设以下情况：

- 您希望使用云函数来实现 Web 后端服务，例如提供博客内的文章查询和文章内容。
- 您希望使用 API 来对外提供服务供网页和 App 使用。

实现概要

下面是该服务的实现流程：

- 创建函数，在 API 网关中配置 API 规则并且后端服务指向函数。
- 用户请求 API 时带有文章编号。
- 云函数根据请求参数，查询编号对应内容，并使用 json 格式响应请求。
- 用户可获取到 json 格式响应后进行后续处理。

请注意，完成本教程后，您的账户中将具有以下资源：

- 一个由 API 网关触发的 SCF 云函数。
- 一个 API 网关中的 API 服务及下属的 API 规则。

本教程分为了三个主要部分：

- 完成函数代码编写、函数创建和测试。
- 完成 API 服务和 API 规则的设计，创建及配置。
- 通过浏览器或 HTTP 请求工具测试验证 API 接口工作的正确性。

API 设计

现代应用的 API 的设计通常遵循 RESTful 规范。因此，在本示例中，我们可以设计获取博客文章的 API 如下：

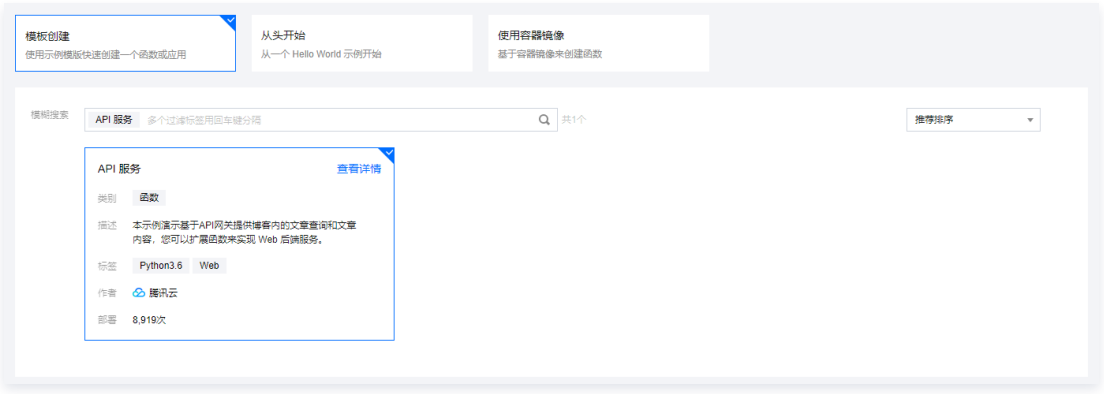
- `/article GET`：返回文章列表。
- `/article/{articleId} GET`：根据文章 ID，返回文章内容。

步骤 1. 创建 blogArticle 函数

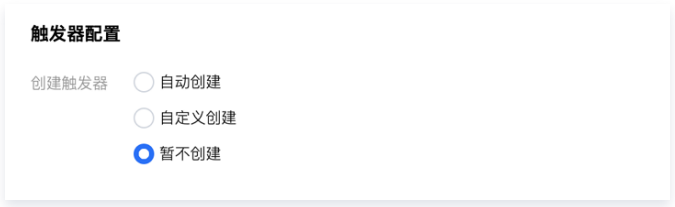
最近更新时间：2023-09-18 11:12:01

在此部分中，将创建一个云函数并通过控制台的 API 测试正确性，来实现博客文章的 API 响应。

1. 登录 Serverless 控制台，单击左侧导航栏的函数服务。
2. 在**函数服务**页面上方选择**北京**地域，并单击**新建**进入新建函数页面。
   设置以下参数信息，并单击**下一步**。如下图所示：
   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入“API 服务”，并进行搜索。
     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。



3. 函数名称默认填充，可根据需要自行修改。在**触发器配置**中，选择**暂不创建**。如下图所示：



4. 单击**完成**，即可完成函数创建。

> **注意**
> 保存文章的数据结构采用 testArticleInfo 变量进行的保存和模拟，此处在实际应用中通常从数据库中或者文件中读取。

步骤 2. 创建并测试 API 服务

最近更新时间：2025-06-13 15:09:02

操作场景

在此部分中，将创建一个 API 网关中的服务和相关的 API 规则，对接在 步骤1 中已创建的 SCF 云函数，并通过控制台的 API 测试，来测试 API 的正确性。

> **注意**
> API 服务和函数必须位于同一个地域下。在本教程中，将使用北京区域来创建 API 服务。

创建 API 服务及规则

1. 登录 API 网关控制台，选择左侧导航栏中的**服务**。
2. 在“服务”页面上方选择北京地域，并单击**新建**进入新建 API 服务页面。
3. 在弹出的“新建服务”窗口中，设置以下参数信息，并单击**提交**创建服务。
   - **服务名**：`blogAPI`。
   - **访问方式**：选择“公网”。
4. 在服务列表中选择已创建的 `blogAPI` 服务，进入“管理 API”页。
5. 单击**新建**进入“新建 API” 页面，在“前端配置”步骤中，参考以下主要参数信息进行创建 API：
   - **API名称**：自定义 API 名称。
   - **路径为** `/article`。
   - **请求方法**：GET。
   - **鉴权类型**：选择“免认证”。
     其余配置请保持默认设置，单击**下一步**。
6. 在“后端配置”步骤中，参考以下主要参数信息进行创建 API：
   - **后端类型**：选择“云函数SCF”。
   - **云函数**：选择函数为 步骤1 中已创建的 `blogArticle` 函数。
     其余设置请保持默认值，单击**下一步**。
7. 在“响应结果”中单击**完成**即可完成 API 的创建。在弹窗中选择“发布环境”为“测试”并选择**发布服务**。
8. 再次在“管理API”页签中单击**新建**创建 API。
   - **路径为**：`/article/{articleId}`。
   - **请求方法**：GET。
   - **鉴权类型**：选择“免认证”。
   - **参数配置**：选择“新增参数配置”，并参考以下参数进行配置：
     - **参数名**：articleId
     - **参数位置**：Path
     - **类型**：int
9. 在“后端配置”步骤中，参考以下主要参数信息进行创建 API：
   - **后端类型**：选择“云函数 SCF”。
   - **云函数**：选择函数为 步骤1 中已创建的 `blogArticle` 函数。
     其余设置请保持默认值，单击**下一步**。
10. 在“响应结果”中单击**完成**即可完成 API 的创建。在弹窗中选择“发布环境”为“测试”并选择**发布服务**。

调试 API 规则

1. 针对创建 API 服务及规则 步骤5 创建的 `/article` API，单击**调试**，在调试页面发送请求，确保返回结果内的响应 Body，为如下内容：

```
[{"id": 1, "category": "blog", "title": "hello world", "time": "2017-12-05 13:45"}, {"id": 2,
"category": "blog", "title": "record info", "time": "2017-12-06 08:22"}, {"id": 3, "category":
"python", "title": "python study", "time": "2017-12-06 18:32"}]
```

2. 针对创建 API 服务及规则 步骤8 创建的 `/article/{articleId}` API，单击**API 调试**，在调试页面将请求参数修改为1后发送请求，确保返回结果内的响应 Body，为如下内容：

```
{"id": 1, "category": "blog", "title": "hello world", "content": "first blog! hello world!", "time":
"2017-12-05 13:45"}
```

> **说明**
> 您可以修改请求参数 `articleId` 的值为其他数字，并查看响应内容。

步骤 3. 发布 API 服务并在线验证

最近更新时间：2022-12-14 17:48:02

如果您已完成了 步骤2：创建并测试 API 服务，且测试结果符合预期，则接下来可以对外发布此服务并从浏览器发起请求来验证接口的工作情况。

API 服务发布

1. 登录 API网关控制台，选择左侧导航栏中的**服务**。
2. 在“服务”页中，选择 步骤2 中创建的 blogAPI 服务名称右侧的**发布**。
3. 在弹出的“发布服务”中，发布环境选择“发布”，备注中填写“发布API”，单击**提交**。

API 在线验证

通过发布动作，完成了 API 服务的发布，使得 API 可以被外部所访问到，接下来通过浏览器发起请求来查看 API 是否能正确响应。

1. 在 blogAPI 服务中，单击 API 名称，进入 API 的**基本信息页**，复制“发布”环境的默认访问地址。例如 `service-kzeed206-1251762227.ap-guangzhou.apigateway.myqcloud.com/release` 。

> **注意**
> 这里由于每个服务的域名均不相同，您的服务所分配到的域名将与本文中的服务域名有差别，请勿直接拷贝本文中的地址访问。

2. 在此路径后增加创建的 API 规则的路径，形成如下路径。

```
service-kzeed206-1251762227.ap-guangzhou.apigateway.myqcloud.com/release/article
service-kzeed206-1251762227.ap-guangzhou.apigateway.myqcloud.com/release/article/1
service-kzeed206-1251762227.ap-guangzhou.apigateway.myqcloud.com/release/article/2
```

3. 将第2步中的路径复制到浏览器中访问，确定输出内容与测试 API 时的输出相同。
4. 可进一步修改请求中的文章编号并查看输出，查看代码是否能正确处理错误的文章编号。

至此完成了通过 SCF 云函数实现服务，通过 API 对外提供服务。后续可以通过继续修改代码，增加功能并增加 API 规则，使其完善成为一个更丰富的应用模块。

SCF + API 网关快速构建文字识别小工具示例说明

最近更新时间：2020-11-12 11:34:43

实现概要

本示例主要演示：

- 在 SCF 控制台上快速创建 API 网关触发器，由云函数实现 Web 后台。
- 在云函数中，通过调用 AI 的接口，实现图像转文字处理。

实现步骤

1. 环境准备：
   - 在 文字识别控制台 开通通用印刷体识别功能。
   - 创建 COS Bucket。
2. 在云函数控制台中，部署云函数，实现 AI 接口的调用，并配置 API 网关触发器。
3. 在本地创建 HTML 静态页面，实现本地图片的上传和处理结果的展示。

演示流程

1. 在本地 Web 页面上传带有文字的图片，将图片做 base64 编码，再通过 API 接口 post 到云函数。
2. 云函数收到请求后，自动运行，在 event 参数中获取文件编码并做 base64 解码，还原图片。
3. （可选）将还原后的图片上传到 COS bucket 中，留存该图片。
4. 通过 AI 的接口对图片进行识别，将识别后的结果返回到前端 Web 页面。

函数部署

最近更新时间：2022-12-22 12:17:57

操作步骤

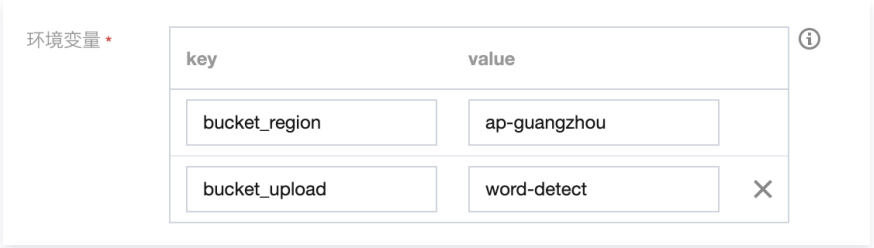
环境准备

1. 前往 文字识别控制台，开通相应服务。操作详情见 文字识别操作指引。
2. 登录 对象存储控制台。创建一个 Bucket，命名为 word-detect，并选择广州地域，权限选择 “私有读写”。

创建云函数及 API 网关触发器

1. 登录 Serverless 控制台，进入**函数服务**页面。
2. 选择**广州**地域，单击**新建**，进入新建函数页面。
3. 填写以下参数信息，并单击**下一步**。如下图所示：
   - 创建方式：选择 “模板创建”。
   - 模糊搜索：输入 “图片转文字” 并进行搜索，本文以运行环境 Python 2.7 为例。
     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. 在**基础配置**中，默认生成**函数名称**，可根据使用需求自行修改。按照引导配置环境变量和运行角色：
   - 环境变量：环境变量填写可参考下表



key	value
bucket_region	存储桶所在地域。以 ap- 开头，加上地域拼音。
bucket_upload	已创建的存储桶名称。

   - 运行角色：勾选 “启用”，选择 “配置并使用SCF模板运行角色”，将会自动创建并选择关联了 COS 全读写权限的 SCF 模板运行角色，或选择 “使用已有角色”，在下拉列表中选择包含上述权限的已有角色。本文以 “配置并使用SCF模板运行角色” 为例。如下图所示：

> **说明**
> 示例需要授权 SCF 操作 COS 的权限，已默认勾选“运行角色”并自动完成函数运行角色的创建和所需 COS 操作权限策略 QcloudCOSFullAccess 的关联，如需调整，请选择“使用已有策略”或取消“运行角色”勾选。

5. **在触发器配置中，选择“自动创建”，如下图所示：**
   如需使用已有 API 服务创建 API 网关触发器或修改触发器配置，请选择“自定义创建”。

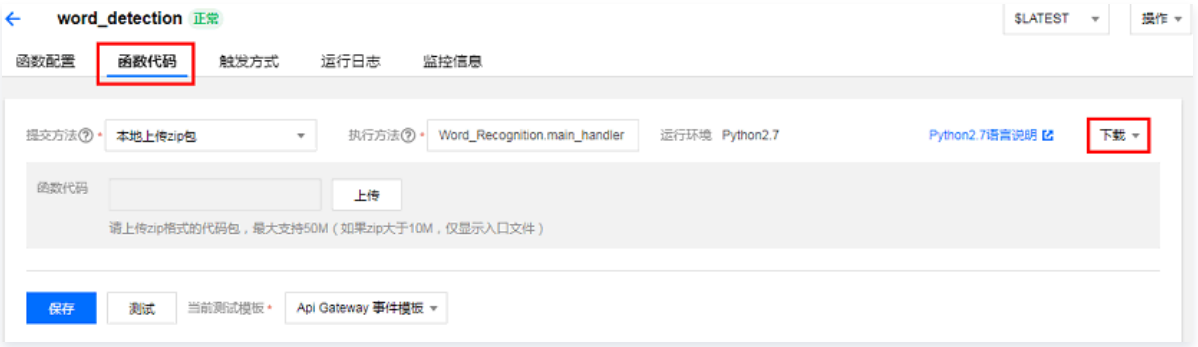
6. **单击完成，即可完成函数和触发器创建并获得该函数的 HTTP 触发域名。**

相关操作

获取函数代码包

1. **当您已成功创建函数后，您可选择函数代码进入函数代码详情页。**

2. 可单击右上角的**下载**，下载代码包或 YAML 文件。如下图所示：



前后台对接

最近更新时间：2022-04-18 10:46:09

操作场景

本文档指导您修改文字识别小工具的 HTML 文件，体验文字识别小工具前后台对接的效果。

前提条件

- 已 下载文字识别小工具的 HTML 文件 。
- 已在 函数部署 中生成的域名。
- 已准备一张带有文字的图片。

操作步骤

修改 URL 地址

1. 使用编辑器打开已下载的 HTML 文件。

> **说明**
> 如果没有专业的编辑器，可以通过文本方式打开。

2. 将下图中的 url 参数修改为 函数部署 中生成的域名。

```
/**
 * 在这里写上传后的处理逻辑
 * */
$("#file").change((e) => {
    const file = e.target.files[0]
    const form = new FormData(file)
    const type = file.type
    if (!(type == 'image/png' || type == 'image/jpeg')) {
        alert('请上传png或jpg')
        return
    }
    const sliceLength = type === 'image/png' ? 22 : 23
    getBase64(e.target.files[0]).then(res => {
        $('.loadingTips').fadeIn()
        $("#filePreviewer").html(`<img width="100%" src=${res} alt=""/>`)

        /**
         * 请把该访问地址替换为您在"函数部署"里生成的域名
         * */
        $.ajax({
            url: "http://service.ap-guangzhou.apigateway.myqcloud.com/release/Word_Recognition", //只用替换这里
            method: 'POST',
            data: res.slice(sliceLength),
            dataType: 'json',
```

体验效果

1. 在浏览器中打开该 HTML 文件，单击**选择文件**，上传图片。如下图所示：

2. 登录 Serverless 控制台，查看函数的运行日志。
3. 切换至 对象存储 COS 的控制台，查看该文件是否已成功上传到 bucket。

SCF + API 网关实现 Web 静态页面托管

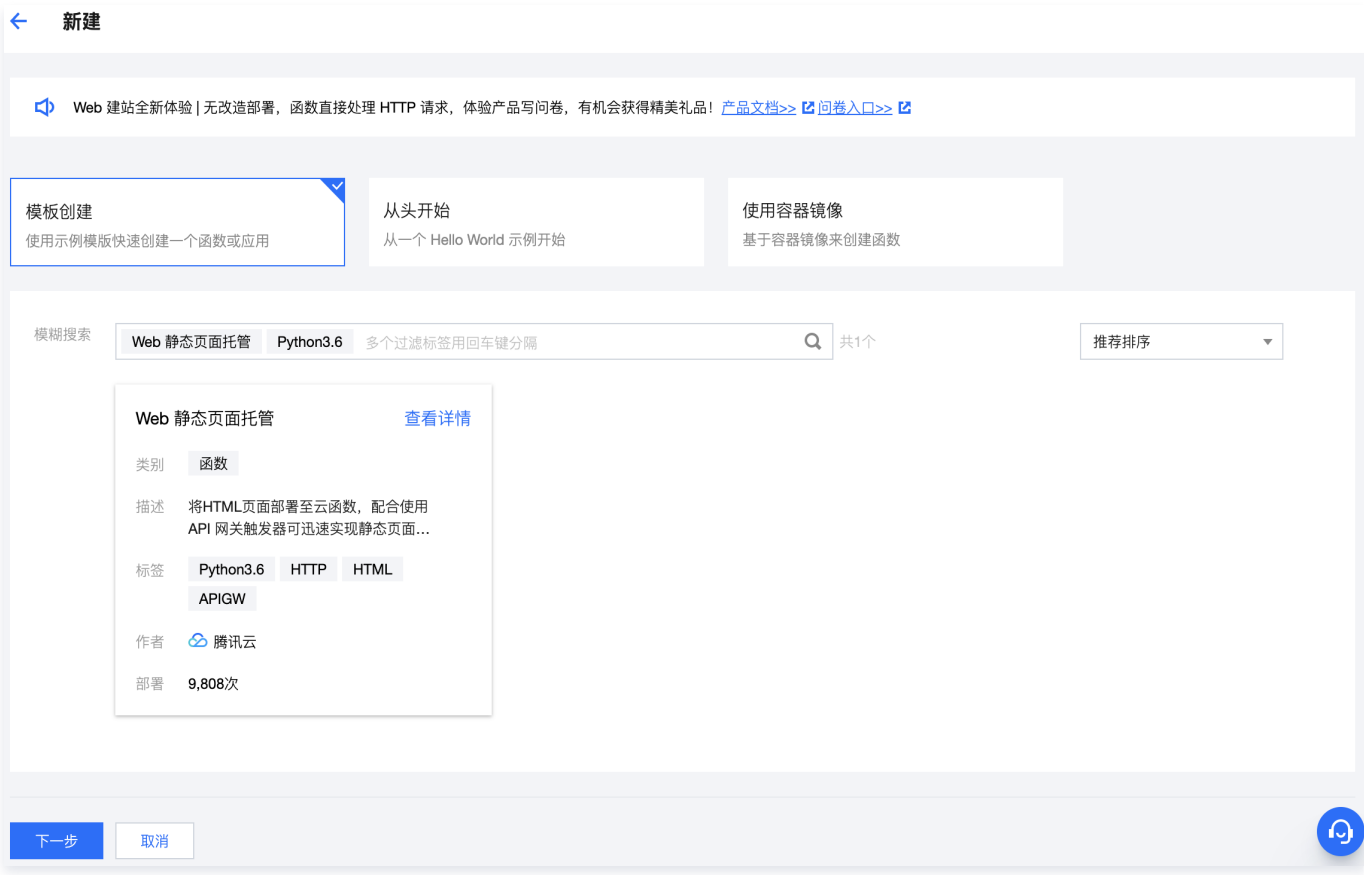
最近更新时间：2022-12-22 11:18:34

操作场景

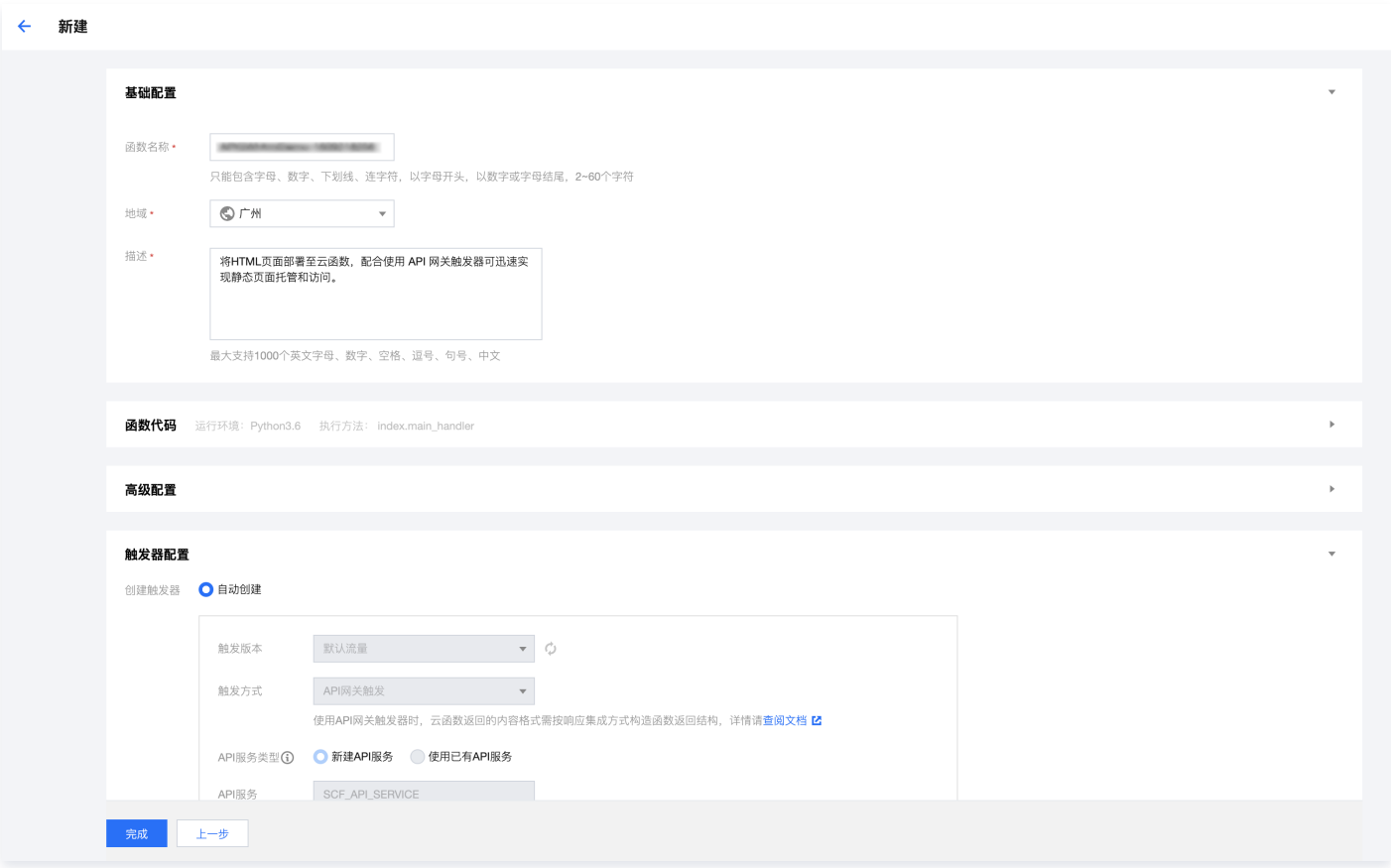
本示例主要为您介绍如何通过云函数 SCF 结合 API 网关，快速实现一个对公网服务的 Web 页面。

操作步骤

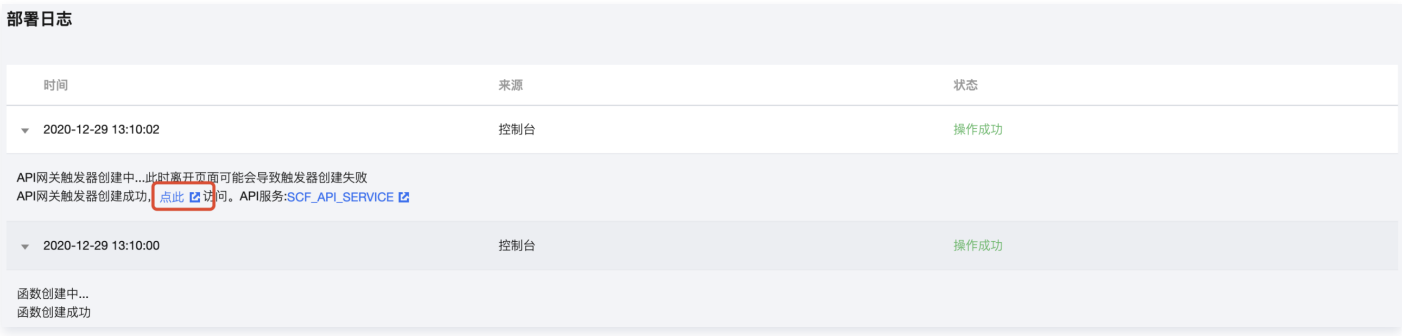
1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”页面上方选择**广州**地域，并单击**新建**进入新建函数页面。
   设置以下参数信息，并单击**下一步**。如下图所示：
   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入 “Web 静态页面托管” “Python3.6”，并进行搜索。
     单击模板中的**查看详情**，即可在弹出的“模板详情”窗口中查看相关信息，支持下载操作。



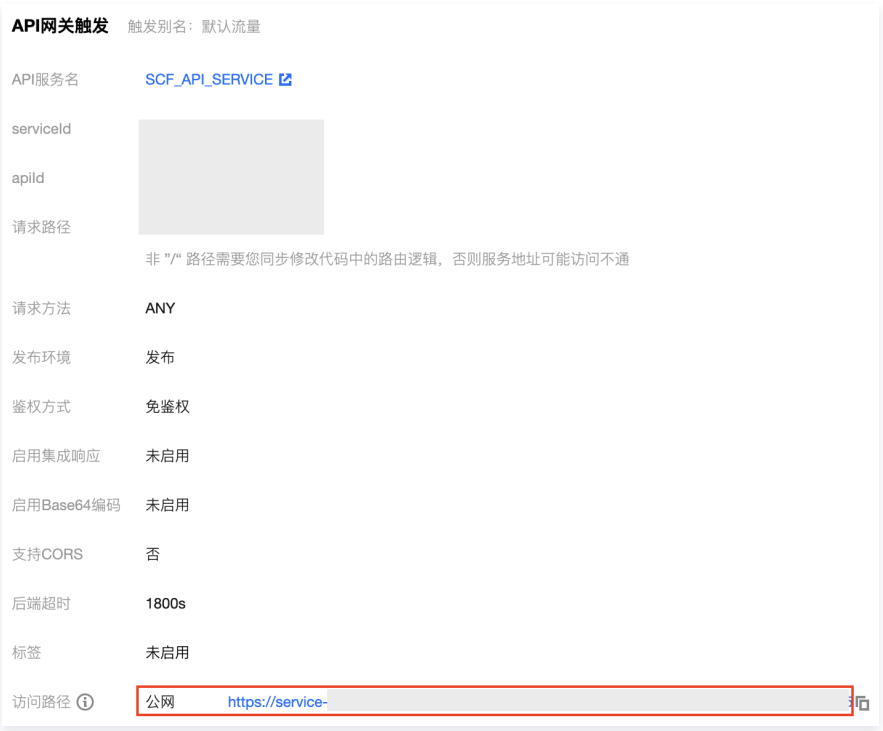
3. 在**基础配置**中，默认生成**函数名称**，可根据使用需求自行修改。保持默认配置，单击**完成**，跳转到**部署日志**中查看函数和触发器创建进度。如下图所示：



4. 函数及触发器创建完成后，可在“部署日志”中点击快速访问链接，如下图所示：



或在“触发管理”页中获取 API 网关触发器访问路径，如下图所示：



查看 Web 页面。如下图所示：

说明

您可在**触发管理**中查看和管理触发器。如果您需要自定义页面展示内容，可以在**函数代码**页签中进行配置。

SCF + API 网关基于 Websocket 搭建匿名聊天室示例说明

最近更新时间：2022-05-16 17:57:05

本示例主要演示：
使用 API 网关和 SCF 搭建 Websocket 后台，实现匿名聊天室。具体的实现原理可参考 Websocket 原理介绍 。

实现概要

实现步骤

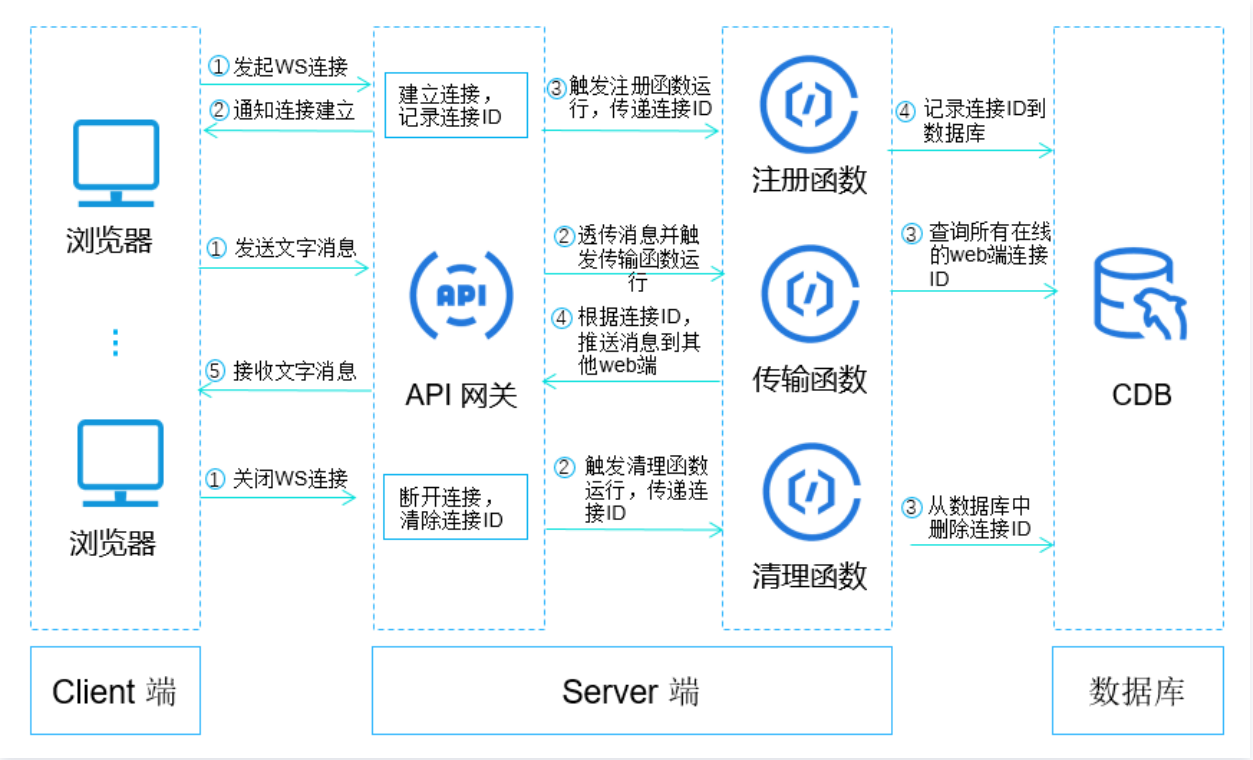
1. 环境准备：腾讯云 MySQL 数据库。
2. 在云函数控制台中，部署云函数。
3. 在 API 网关控制台创建 API 服务。
4. 在本地创建 HTML 静态页面，实现 Websocket 连接及聊天功能。

演示流程

1. 打开本地 HTML 页面，并发送文字内容。
2. 云函数收到文字内容后，自动运行，把文字内容转发给其他 Websocket 网页端，实现多人的匿名聊天功能。

交互流程

具体的交互流程如下图所示：



分别包含如下内容：

- Websocket 连接建立
- 消息发送
- Websocket 连接断开

系统部署

最近更新时间：2025-06-13 15:56:52

环境准备

在开始部署云函数前，您需要完成以下操作：

1. 登录 云数据库 TencentDB 控制台 。
2. 在广州地域下，新建并购买一个数据库。
   - 新购买的数据库默认在 VPC 网络中（即只有内网地址），本文使用外网地址。如下图所示：

   - 如果您未创建数据库实例，建议购买“按量计费”的最小实例，用于所有终端广播消息时使用，存储 Websocket 的连接 ID 及相关信息。
   - 如果您已购买创建数据库实例，可以直接新建数据库或表，供云函数使用。

> **说明**
> 如果您担心数据库的安全问题，需要在 VPC 内网中操作数据库。您可以在部署云函数时，修改函数的配置，将云函数部署在 VPC 网络环境中，详情请参见 函数部署到 VPC 网络 。

创建云函数

1. 登录 Serverless 控制台，进入**函数服务**页面。
2. 选择**广州**地域，default 命名空间。

创建注册函数

1. 单击**新建**，进入新建函数页面。

2. 设置以下参数信息，如下图所示：

- **创建方式：**选择**模板创建**。
- **模糊搜索：**输入 “Websocket注册函数”，并进行搜索，本文以运行环境 Python 2.7 为例。
  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。

3. 单击**下一步**并保持默认配置，单击**完成**即可创建注册函数。

创建传输函数

1. 单击**新建**，进入新建函数页面。
2. 设置以下参数信息，如下图所示：

- **创建方式：**选择**模板创建**。
- **模糊搜索：**输入 “Websocket传输函数”，并进行搜索，本文以运行环境 Python 2.7 为例。
  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。

3. 单击**下一步**并保持默认配置，单击**完成**即可创建传输函数。

创建清理函数

1. 单击**新建**，进入新建函数页面。
2. 设置以下参数信息，如下图所示：

   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入 “Websocket清理函数”，并进行搜索，本文以运行环境 Python 2.7 为例。
     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。
3. 单击**下一步**并保持默认配置，单击**完成**即可创建清理函数。

配置 API 网关

创建 API

1. 登录 API 网关控制台，单击左侧导航栏**服务**。
2. 在服务页面，选择页面上方的**广州**地域，并单击**新建**。

3. 在弹出的“新建服务”页面填写如下参数信息，单击**下一步**，按照提示完成相应配置，如下图所示：

主要参数信息说明如下：

- **服务名**：最长50个字符，支持 `a - z，A - Z，0 - 9，_`，此处以 Websocket 为例。
- **实例**：选择**共享型**。
- **备注**：此处以 websocket 为例。

4. 单击**下一步**，填写如下参数信息。如下图所示：

主要参数信息说明如下：

- **前端类型**：选择HTTP。

- **访问方式**：选择公网。

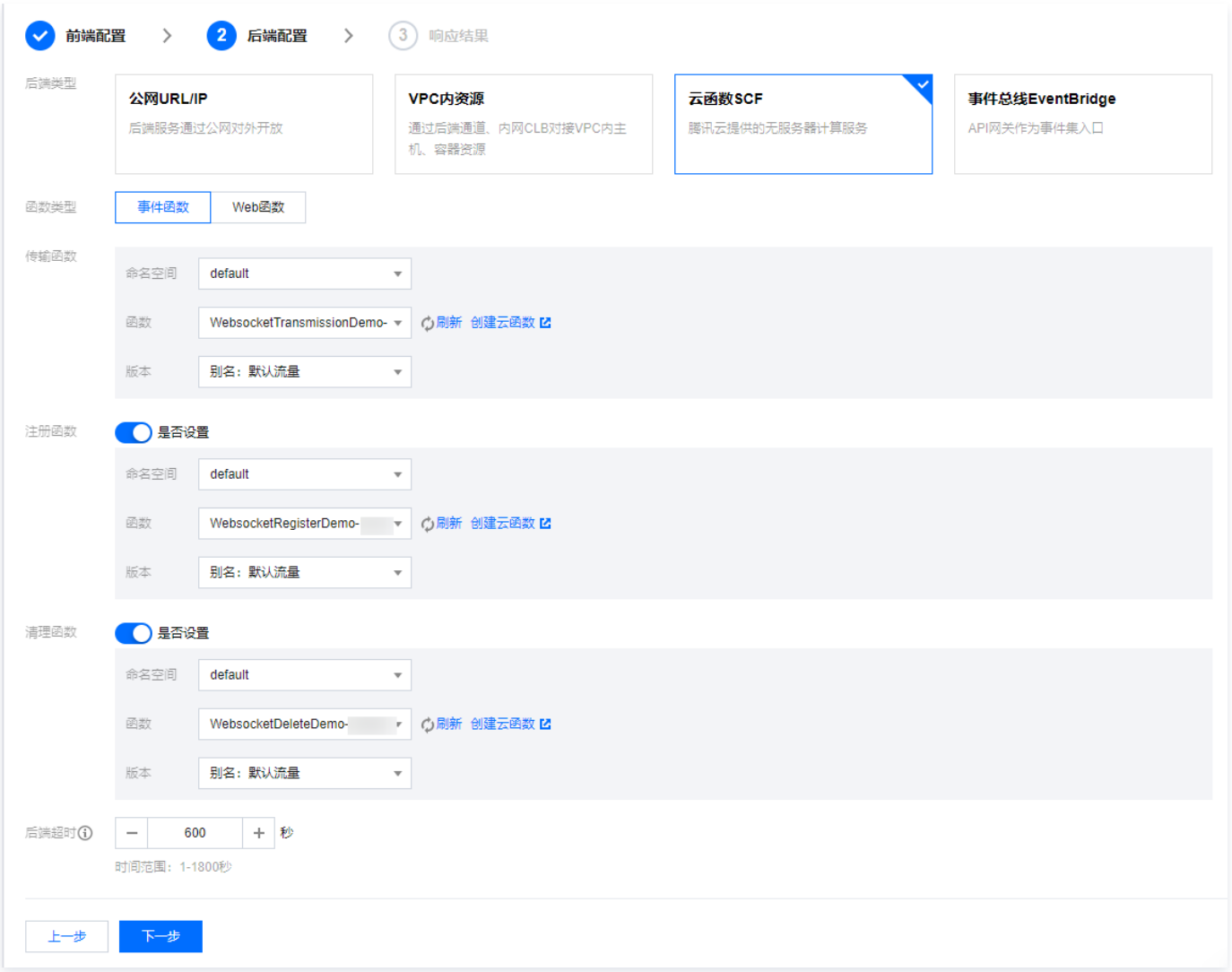
5. 单击**提交**完成创建。
6. 单击已创建的服务 ID，进入服务详情页。选择**管理API > 新建**。如下图所示：

7. 在“前端配置”页面填写以下参数信息，其余设置保持默认，填写完成后单击**下一步**，按照提示完成相应配置，如下图所示：

主要参数信息说明如下：

- **API名称**：命名为 “websocket”。
- **前端类型**：选择**WS**。
- **路径**：填写 `/websocket`。
- **鉴权类型**：选择**免认证**。

8. 在“后端配置”页面填写以下参数信息，单击**下一步**，按照提示完成相应配置，如下图所示：



主要参数信息说明如下：

- **后端类型**：选择**云函数SCF**。
- **传输函数**：选择新创建的传输函数，即 `WebsocketTransmissionDemo-xxx` 。
- **注册函数**：勾选，并选择新创建的注册函数，即 `WebsocketRegisterDemo-xxx` 。
- **清理函数**：勾选，并选择新创建的清理函数，即 `WebsocketDeleteDemo-xxx` 。
- **后端超时**：设置为600秒。Client 端在建立 Websocket 连接后，如果一直没有消息发送，将会在超时时间到达后，由 API 网关断开连接。

9. 在“响应结果”页面填写以下参数信息，如下图所示：

- **返回类型**：选择**HTML**。
- **其余保持默认设置**。

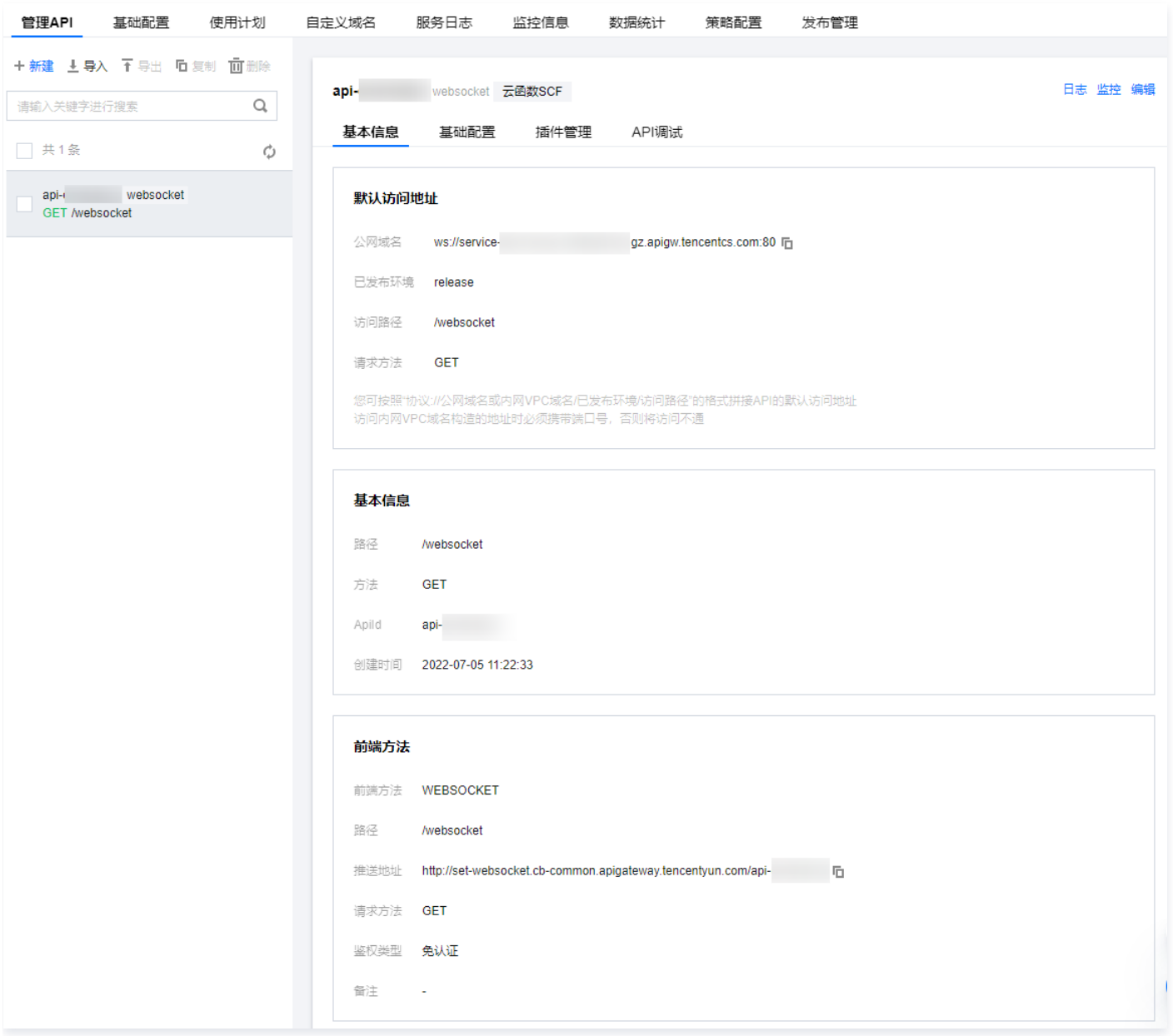
10. 单击**完成**即可创建 API 网关。

发布服务

在弹窗中单击**发布服务**即可将服务发布到正式环境中。

获取 Websocket 反向推送地址

选择**管理API**页签，单击 API 的 ID/名称，**查看 Websocket 的反向推送地址**，及其它配置信息。如下图所示：



注意

Websocket 的反向推送地址会在云函数主动向 Client 端发送消息、或者主动断开与 Client 端的连接时使用。

获取 Websocket 连接地址

选择**发布管理**页签，查看 API 服务地址。如下图所示：

注意

- Websocket 的 API 地址需要带上 API 的路径 `/websocket` 。
- 根据服务地址，可知 Websocket 连接地址为：`ws://service-phe5h6qu-xxxxxxxxx.gz.apigw.tencentcs.com/release/websocket` 。

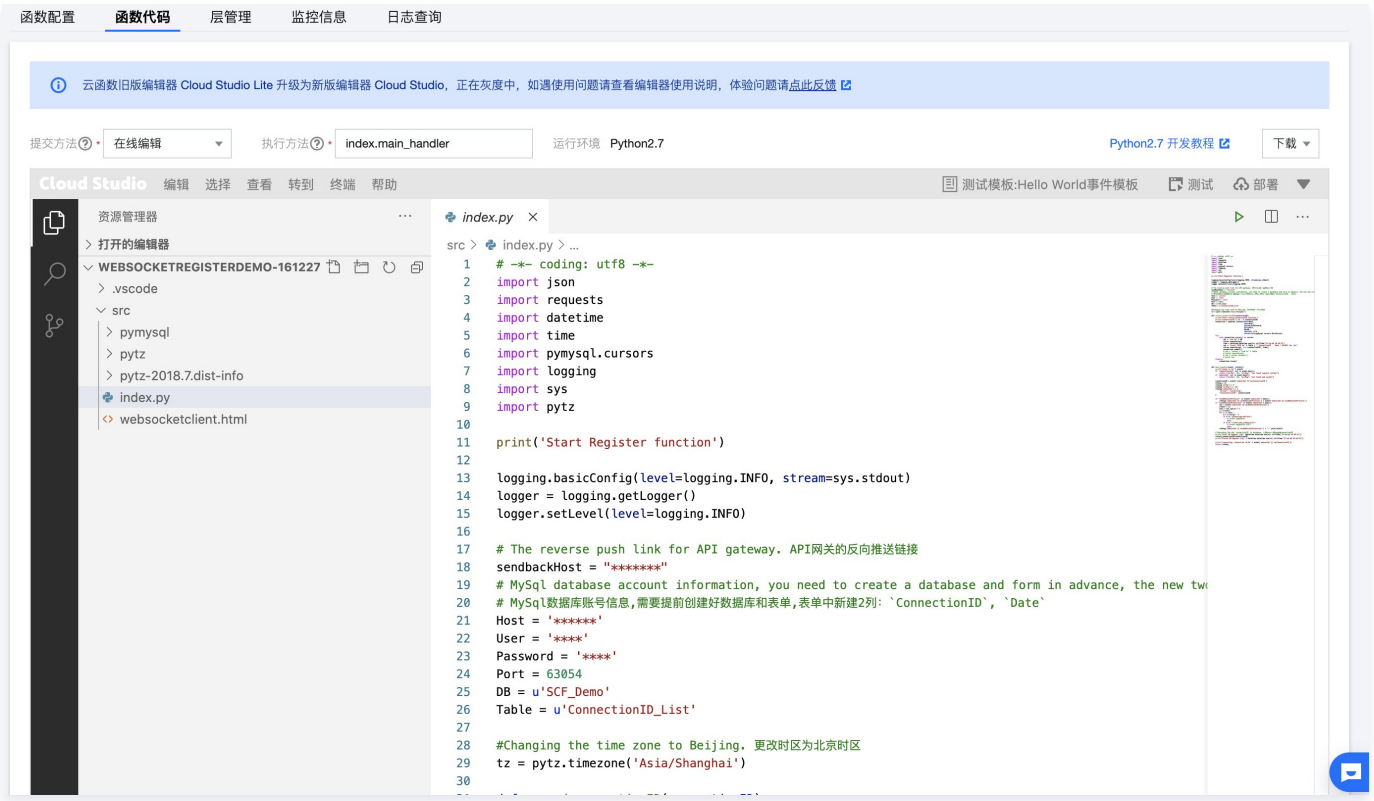
修改函数初始化信息

1. 如若未创建数据库，可参考 建立数据库和表，创建以下数据库、数据表和列：
   - **数据库**：命名为 “SCF_Demo”。
   - **数据表**：命名为 “ConnectionID_List”。
   - **列**：命名为 “ConnectionID、Date”。
2. 打开数据管理，确认已在数据库实例中创建数据库。如下图所示：

修改注册函数

1. 登录 Serverless 控制台。

2. 进入 `WebsocketRegisterDemo-xxx` 的**函数代码**页签，按照以下提示修改信息，并单击**保存**。如下图所示：



- **sendbackHost**：API 网关反向推送链接，请参见步骤 获取 Websocket 反向推送地址 。
- **Host**：数据库外网地址。
- **User**：数据库账号。
- **Password**：数据库账号密码。
- **Port**：数据库外网端口。
- **DB 及 Table**：保持默认。

修改传输函数

请参见 修改注册函数，修改传输函数。

修改清理函数

请参见 修改注册函数，修改清理函数。

系统测试

最近更新时间：2025-06-24 17:24:11

操作场景

本文档指导您修改聊天室的 HTML 文件，体验聊天室的前后台对接效果。

前提条件

- 已 下载聊天室 HTML 文件 。
- 已保存**系统部署** > **配置 API 网关** > 获取 Websocket 连接地址 中的连接地址。

操作步骤

修改 URL 地址

1. 使用编辑器打开已下载的 websocketclient.html 文件。

> ⚠ **注意：**
> 如果您没有专业的编辑器，可以通过文本方式打开。

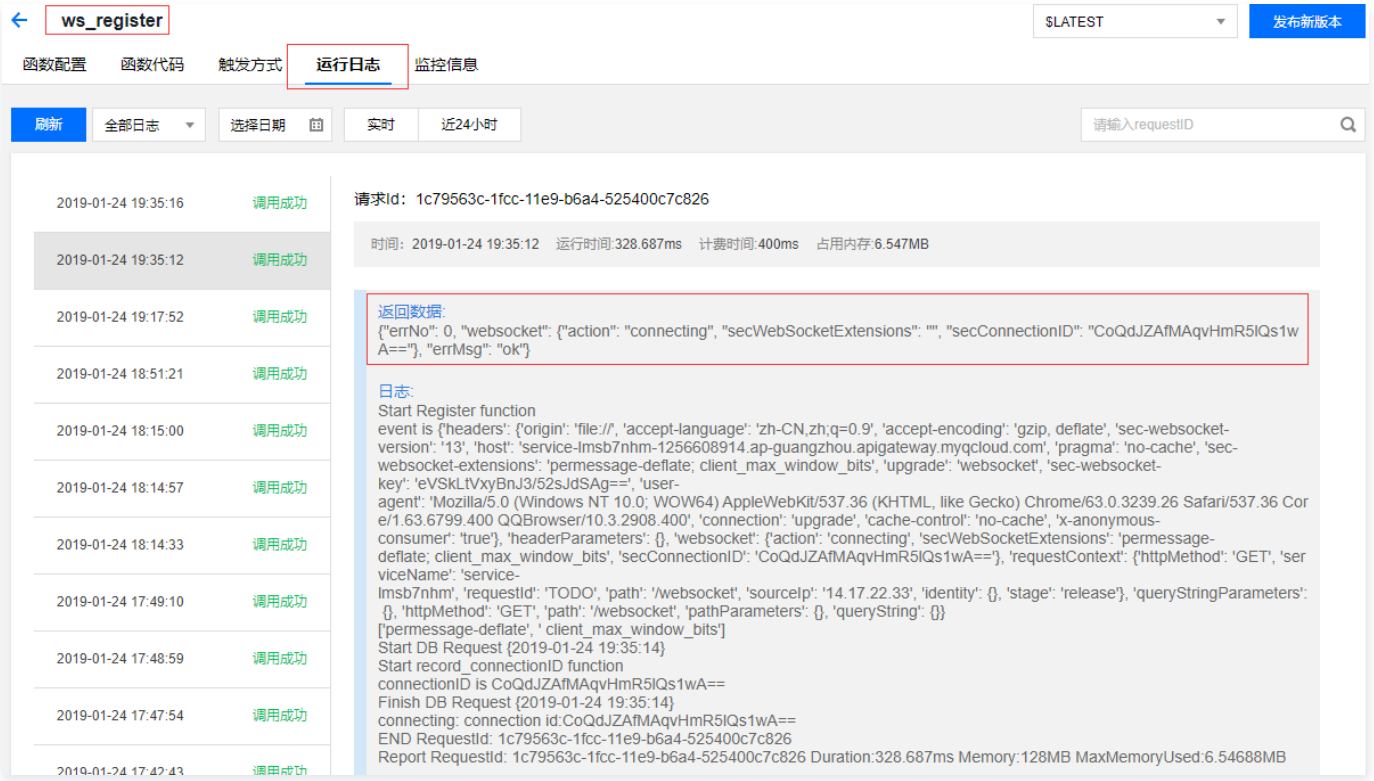
2. 将下图中的 URL 参数修改为 Websocket 连接地址，并保存。

```
if (window["WebSocket"]) {
    //替换为websocket连接地址
    conn = new WebSocket("ws://service-lmsb7nhm-xxxx.ap-guangzhou.apigateway.myqcloud.com/release/websocket");
    conn.onclose = function (evt) {
        var item = document.createElement("div");
        item.innerHTML = "<b>Connection closed.</b>";
        appendLog(item);
    };
    conn.onmessage = function (evt) {
        var item = document.createElement("div");
        item.innerText = "接收↓:";
        appendLog(item);

        var messages = evt.data.split('\n');
        for (var i = 0; i < messages.length; i++) {
            var item = document.createElement("div");
            item.innerText = messages[i];
            appendLog(item);
        }
    };
```

体验效果

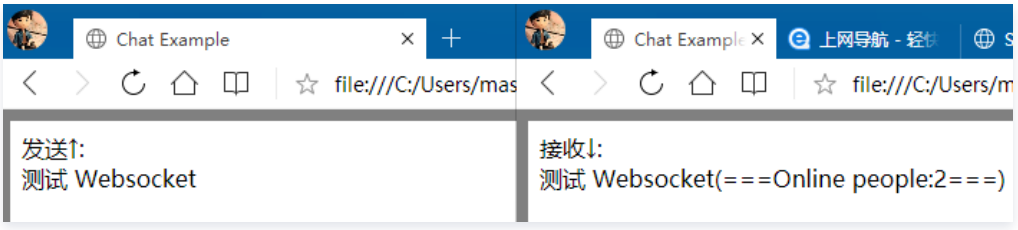
1. 在浏览器中打开 websocketclient.html 文件，并保持为2个及以上窗口。
2. 登录 Serverless 控制台 ，进入**函数服务**页面。

3. 选择 `WebsocketRegisterDemo-xxx` 函数，并切换至**运行日志**页签，查看云函数的执行结果。如下图所示：

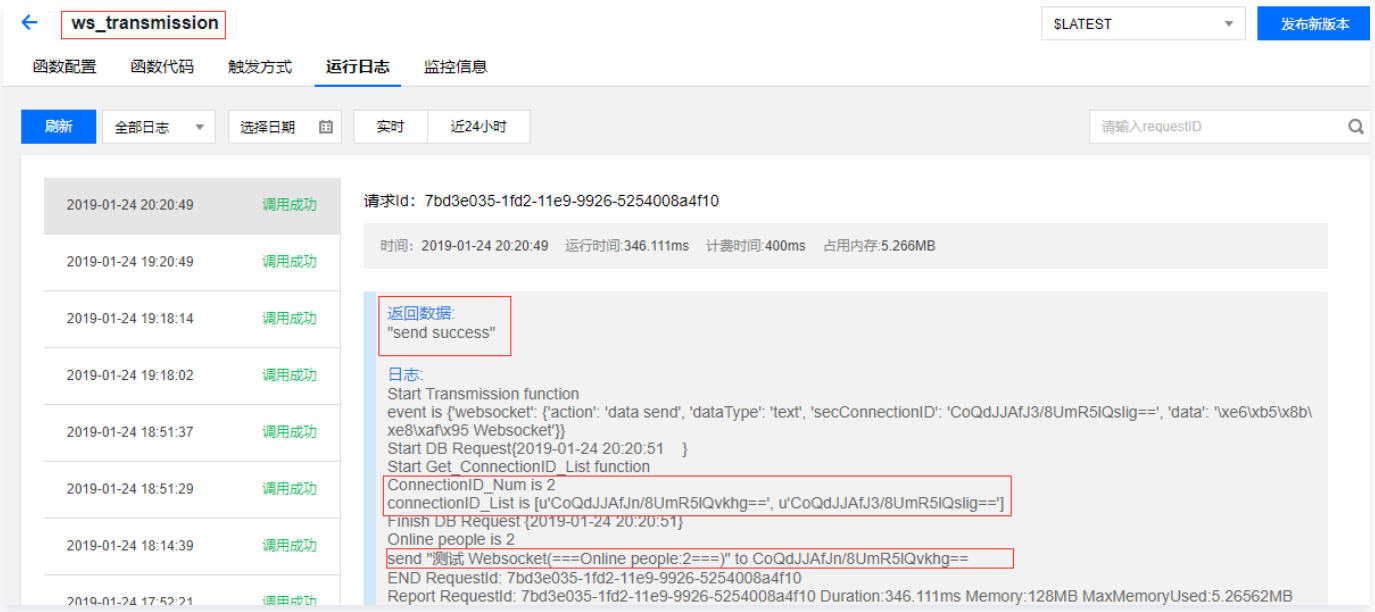


4. 打开数据库管理，查看数据库中记录的信息。如下图所示，数据库管理已记录2个 ConnectionID。

5. 切换至浏览器的 websocketclient 页面，在底部编辑消息，并单击**Send**。如下图所示：
例如，输入 “测试 Websocket”，单击**Send**，在另外一个窗口中，即可查看接收到的消息。



6. 切换至 Serverless 控制台，在**函数服务**页面，选择 `WebsocketTransmissionDemo-xxx` 函数，并切换至**运行日志**页签，查看云函数的执行结果。如下图所示：



7. 切换至 websocketclient 页面，关闭其中一个页面。
8. 切换至 Serverless 控制台，在**函数服务**页面，选择 `WebsocketDeleteDemo-xxx` 函数，并切换至**运行日志**页签，查看云函数的执行结果。如下图所示：

9. 切换至数据库管理，刷新数据库数据，查看数据库中记录的信息。如下所示，即可发现数据库管理已经删除一个 ConnectionID。

SCF + API 网关处理多文件上传

最近更新时间：2024-12-06 09:23:11

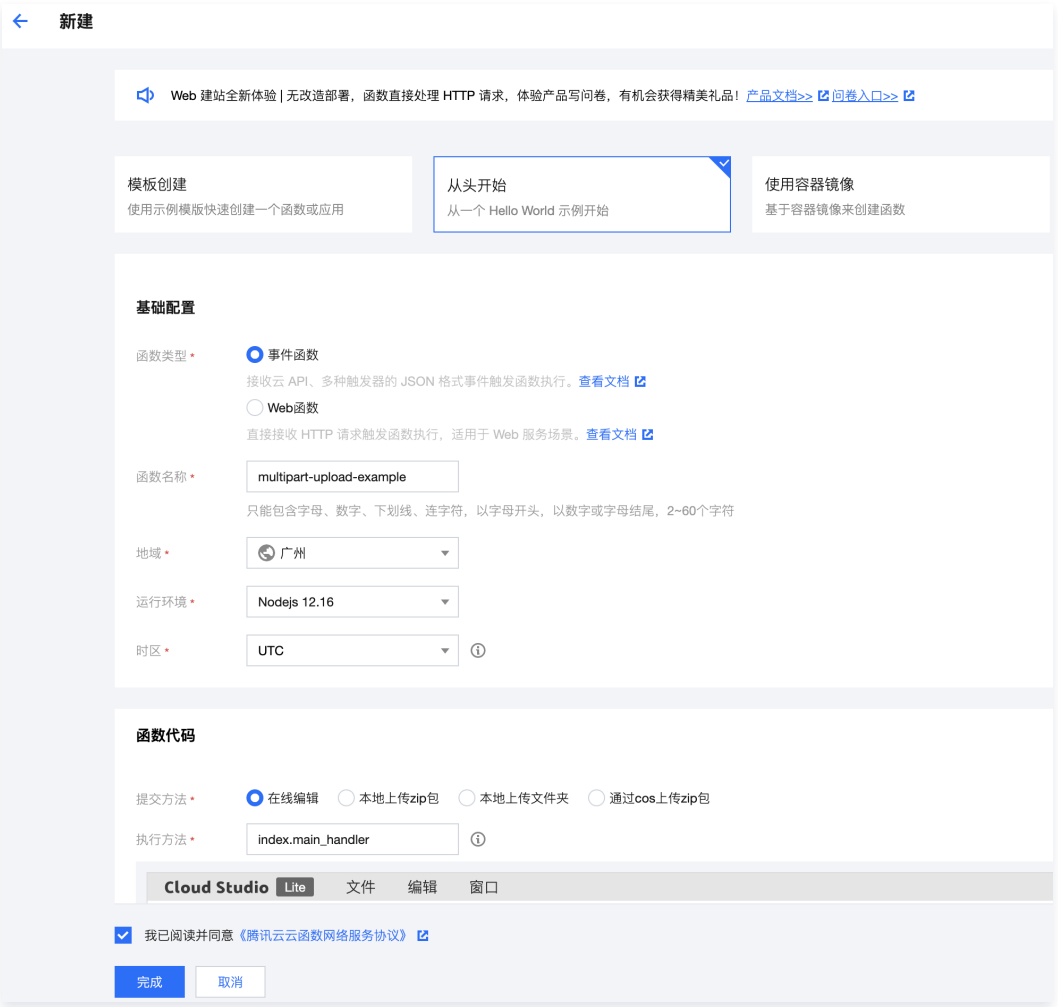
操作场景

通过腾讯云 Serverless 处理 multipart/form-data 多文件上传的 HTTP 请求，原理上需要利用 API 网关的 Base64 编码能力，将原始 HTTP 请求中的 multipart 字节流编码为字符串，以便将 HTTP Event 序列化，传入云函数 SCF 进行处理。
云函数将 API 网关传来 event 中的 body 获取并解码 Base64 后，生成的字节流则与普通 HTTP 请求中的无异，正常处理即可。在 Node.JS 中，我们可以利用 `busboy` 等库进行处理。

操作步骤

步骤1：创建云函数

1. 登录 Serverless 控制台 。
2. 在**函数服务**页面，单击**新建**，创建一个 `Node.js` 云函数。
3. 创建时，具体参数如下：



4. 单击**完成**，完成云函数的创建。

步骤2：编写代码并部署

1. 云函数创建完成后，可以参考以下示例代码编写处理 multipart/form-data 的具体逻辑。

```
// handler.js
"use strict";
const stream = require("stream");
```

```
const Busboy = require("busboy");

/** 处理用户上传 （POST） */
const handlePost = (event) => {
  return new Promise((resolve, reject) => {
    const busboy = new Busboy({ headers: event.headers });
    let html = "";
    /** 接收到文件 */
    busboy.on("file", (fieldname, file, filename, encoding, mimetype) => {
      let buf = Buffer.alloc(0);
      console.log({ fieldname });
      /** 接收到文件的数据块，拼接出完整的 buffer */
      file.on("data", function (data) {
        buf = Buffer.concat([buf, data]);
      });
      /** 文件的数据块接收完毕，生成 DOM 字符串 */
      file.on("end", function () {
        const imgBase64 = buf.toString("base64");
        html += `<img src="data:${mimetype};base64, ${imgBase64}" />`;
      });
    });
    /** multipart/form-data 接收完毕，构造并返回生成的 html */
    busboy.on("finish", function () {
      console.log({ msg: "Parse form complete!", html });
      resolve({
        statusCode: 200,
        headers: {
          "content-type": "text/html",
        },
        body: html,
      });
    });

    /**
     * busboy 需要 stream pipe 的方式来进行处理，
     * 我们将 body 解码为 buffer后，
     * 转换为 stream，最终 pipe 给 busbody
     */
    const bodyBuf = Buffer.from(event.body, "base64");
    var bufferStream = new stream.PassThrough();
    bufferStream.end(bodyBuf);
    bufferStream.pipe(busboy);
  });
};

/** 返回静态文件 */
const handleGet = (event) => {
  const html = `<html><head></head><body>
    <form method="POST" enctype="multipart/form-data">
    <input type="file" name="image-1" accept="image/*"><br />
     <input type="file" name="image-2" accept="image/*"><br />
     <input type="submit">
    </form>
    </body></html>`;
  console.log({ msg: "Get form complete!", html });
  return {
    statusCode: 200,
    headers: {
      "content-type": "text/html",
    },
    body: html,
```

```
  };
};

/** 云函数入口函数 */
exports.main_handler = async (event, context) => {
  const method = event.httpMethod;
  /** 当请求为 POST 请求时，我们处理用户的 multipart/form-data，并生成展示上传结果的页面 */
  if (method === "POST") {
    return handlePost(event);
  }
  /** 当请求为 GET 请求时，我们返回上传文件的页面 */
  if (method === "GET") {
    return handleGet(event);
  }
};
```

2. 编写代码后，您也可以为云函数安装运行时需要的依赖。例如，利用 busboy 进行 multipart/form-data 数据的解码。

> **注意：**
> 依赖要安装在 src 文件夹下。

```
[root@ws-lgivut-0 src]# npm i busboy
npm WARN saveError ENOENT: no such file or directory, open '/usr/local/var/
npm notice created a lockfile as package-lock.json. You should commit this
npm WARN enoent ENOENT: no such file or directory, open '/usr/local/var/fur
npm WARN src No description
npm WARN src No repository field.
npm WARN src No README data
npm WARN src No license field.

+ busboy@0.3.1
added 3 packages from 1 contributor and audited 3 packages in 0.386s
found 0 vulnerabilities
```

3. 单击**部署**，完成云函数的部署。

步骤3：绑定 API 网关触发器

在云函数的触发管理中，我们需要为云函数绑定 API 网关触发器，才能够处理用户具体的 HTTP 请求，具体的绑定方式和配置如下图：

这个时候，如果访问 API 网关绑定的链接，会发现虽然静态页面能够工作，但是上传图片后，页面没有展示正确的结果。这是因为默认情况下，API 网关没有开启 base64 编码功能，multipart formdata 被错误编码为字符串传入 handler 函数，busboy 自然无法进行解码。
因此，我们需要进入 API 网关，找到绑定的 API 服务，在其中的基础配置中打开 base64 编码。

打开并发布服务后，我们的服务就可以正常工作了。

SCF + API 网关实现自定义邀请函

最近更新时间：2022-12-22 10:24:31

操作场景

本文档介绍如何通过云函数 SCF 结合 API 网关实现自定义邀请函，输入嘉宾名字即可生成邀请函。

操作步骤

创建存储桶

存储桶用于存储自定义生成的邀请函。具体步骤如下：

1. 登录 对象存储控制台，选择左侧导航栏中的**存储桶列表**。
2. 在“存储桶列表”页面中，单击**创建存储桶**。
3. 在弹出的“创建存储桶”窗口中，参考以下信息进行创建。如下图所示：

主要参数信息如下，其余参数请保持默认设置：

- **所属地域**：选择“广州”。
- **名称**：自定义名称，本文以 test 为例。
- **访问权限**：公有读私有写。

4. 单击**创建**。
5. 选择存储桶左侧的**安全管理 > 跨域访问CORS设置**，单击**添加规则**。

6. 在“添加跨域访问CORS规则”中，参考以下信息添加规则。

主要参数信息如下，其余参数请保持默认设置：

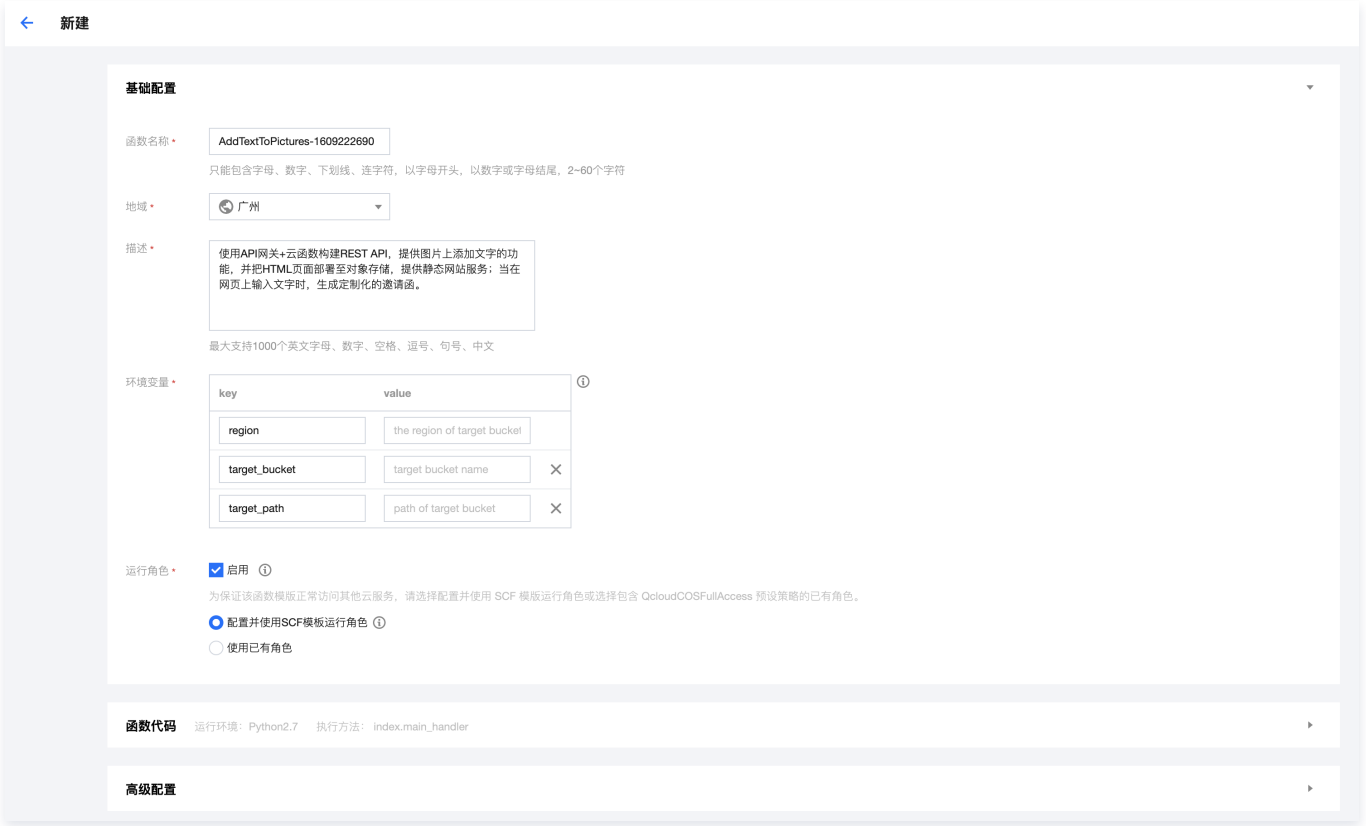
- **来源 Origin**：输入 * 。
- **操作 Methods**：勾选 “GET” 。
- **Expose-Headers**：输入 “-” 。

7. 单击**保存**。

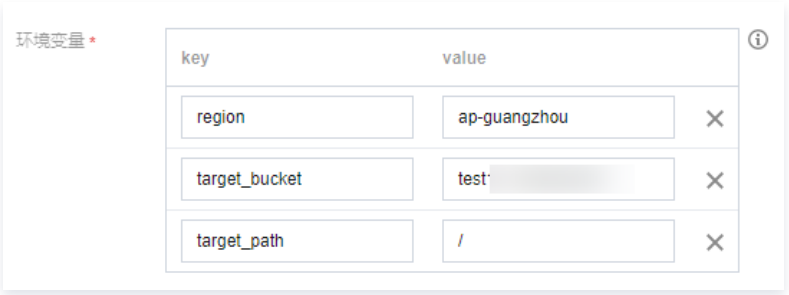
创建函数及 API 网关触发器

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”页面上方选择**广州**地域，单击**新建**进入新建函数页面。
3. 按照如下流程搜索自定义邀请函模板，如下图所示，本文以运行环境 Python2.7为例：
   - **创建方式**：选择**模板函数**。
   - **模糊搜索**：输入“自定义邀请函”，并进行搜索。
     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，函数名称默认填充，可根据需要自行修改。按照引导在“基础配置”中填入该模板需要的环境变量对应的值，其他保持默认配置。如下图所示：**



环境变量填写可参考下表：



key	value
region	存储桶所在地域。以 ap- 开头，加上地域拼音。本文示例为 `ap-guangzhou`。
target_bucket	已创建的存储桶名称。
target_path	目标存储桶路径。用于存放邀请函的存储桶目录路径，请前往 存储桶列表，进入“对应的存储桶”，查看对应的目录路径。例如目录为 example，则此处的目标路径即为 /example。若在存储桶在未创建目录，那么此处输入 “/” 即可。

说明

本示例需要授权 SCF 操作 COS 的权限，已默认勾选“运行角色”并自动完成函数运行角色的创建和所需 COS 操作权限策略 QcloudCOSFullAccess 的关联，如需调整，请选择“使用已有策略”或取消“运行角色”勾选。

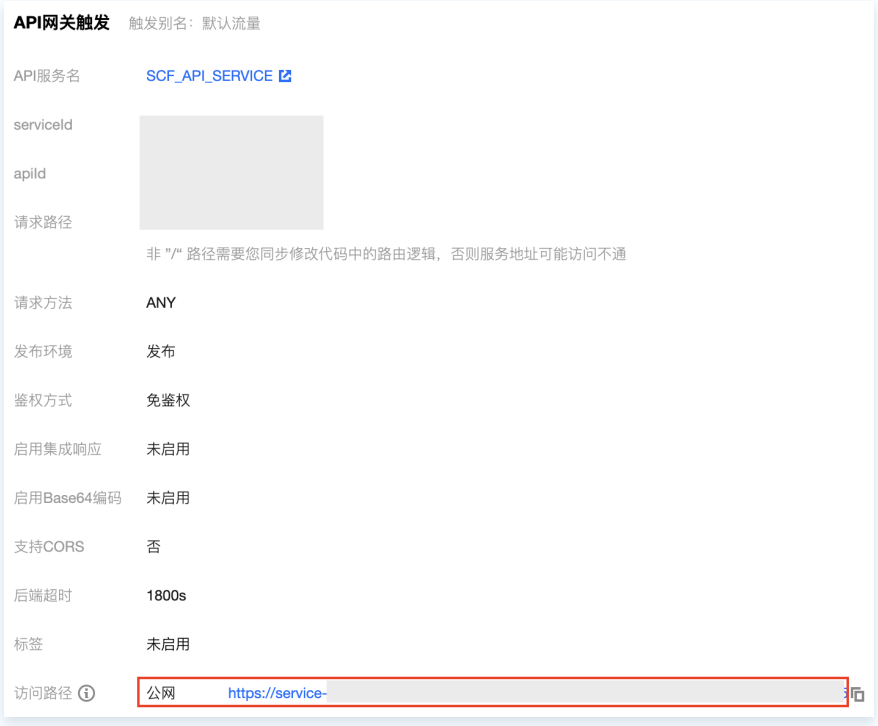
5. 单击**完成**，完成函数和 API 网关触发器创建。

生成邀请函

您可通过以下两种方式，进行邀请函生成：

说明

您可在**函数详情 > 触发管理**中获取 API 网关触发器访问路径，如下图所示：



方式1

在命令行中，执行以下命令。

```
curl API网关触发器地址  -d '邀请嘉宾名字'
```

可在终端获取邀请函的下载地址。例如：

```
curl https://service-xxxx.gz.apigw.tencentcs.com/release/test-123456 -d 'name'
https://testxxxx.com//邀请函-yun-ServerlessDays.jpg%
```

方式2

1. 下载 HTML 页面，并将链接修改为 API 网关触发器的链接。如下图所示：

```
21          // 绑定点击事件
22          document.querySelector('#btnAjax').onclick = function () {
23              // 更改后端服务地址
24              var url="https://service-[redacted].gz.apigw.tencentcs.com/release/test";
25              // 异步对象
26              var ajaxObj = new XMLHttpRequest();
```

2. 打开 HTML 页面，输入邀请嘉宾的名字，则可以生成海报，访问链接可直接下载海报。如下图所示：

小程序云开发 TCB
在小程序云开发中实现函数互调及邮件发送
示例说明

最近更新时间：2024-11-07 10:23:52

本示例主要演示：

- 在小程序云开发中，使用微信提供的 SDK “wx-server-sdk”，实现云函数之间的互相调用。
- 在小程序云开发的云函数中，实现邮件发送功能。
- 针对自身的实际业务进行功能扩充。

实现概要

实现步骤

1. 环境准备：
   - 获取小程序 AppID。
   - 下载安装微信开发者 IDE。
   - 运行环境 Node8.9 或以上。
2. 在微信 IDE中，本地创建两个云函数，分别命名为 test 和 sendemail。
3. 将新建的两个云函数上传部署到云端。
4. 使用 test 云函数发起对 sendemail 的调用，实现邮件发送。

演示流程

1. 在 CloudBase 云开发控制台，选择**云函数**，单击 “test” 函数，进入 “test” 函数信息页面。
2. 在 “test” 函数信息页中，单击**测试**，即可根据配置的邮箱地址，收到邮件。
3. 在日志中查看 test 函数和 sendemail 函数被执行的结果。

详细操作步骤请参见 函数部署 。

函数部署

最近更新时间：2023-09-19 10:22:02

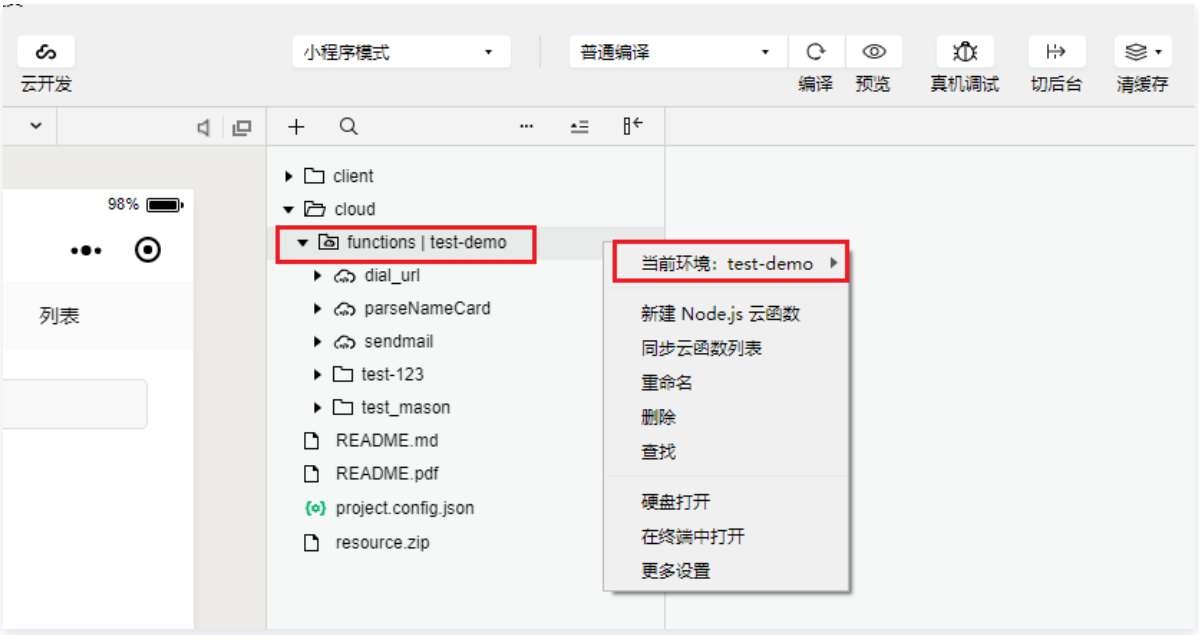
环境准备

在开始部署云函数前，您需要：

1. 申请获取小程序 AppID 。
2. 下载并安装 微信开发者 IDE 。
3. 在您的电脑上，安装 Node8.9 或以上运行环境。
4. 在您的电脑上，安装 npm 工具，关于 npm 详细可参考 npm 官网 。
5. 阅读并了解小程序云开发使用的基础信息，详情可参考 微信官网教程 。

创建小程序项目并部署云函数

1. 在微信开发者 IDE中，新建一个项目，并填写项目信息。
2. 在 IDE 的工具栏中，单击**云开发**，申请开通云端环境（即自动生成数据库实例、对象存储桶、云函数运行环境）。
3. 在 IDE 的代码栏中，右键单击 cloud 目录下的 functions，将 “当前环境” 设置为指定本地函数目录的云端环境。如下图所示：



4. 右键单击 functions，选择**新建 Nodejs 云函数**，新建 Nodejs 云函数。例如，新建一个名称为 test 的 Nodejs 云函数。
5. 在新建的 Nodejs 云函数的 index.js 中，编写主函数。例如，将以下代码作为主函数复制至 index.js 中。

```
const cloud = require('wx-server-sdk')
cloud.init({
  env: 'test-demo-id'   //环境初始化
})
// 云函数入口函数
exports.main = async (event, context) => {
  console.log("Start to test")
  return await cloud.callFunction({
    name:'sendmail',
  })
}
```

6. 判断是否需要安装第三方库。
   - 是，使用 npm 工具安装或者在 package.json 中声明。 package.json 声明如下所示：

```
{
 "name": "test",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
 "test": "echo \"Error: no test specified\" && exit 1"
  },
 "author": "",
 "license": "ISC",
 "dependencies": {
 "wx-server-sdk": "latest"
  }
}
```

- 否，请执行下一步。

7. 右键单击新建的 Nodejs 云函数（即 test 函数），选择 “上传并部署（云端安装依赖）”，将修改后的代码同步到云端。
8. 重复执行 步骤4，新建一个 Nodejs 云函数，并将该云函数命名为 sendemail。
9. 在 sendemail 云函数的 index.js 中，编写主函数。例如，将以下代码作为主函数复制至 sendemail 云函数的 index.js 中。

```
// 云函数入口文件
const nodemailer = require("nodemailer");
var transporter = nodemailer.createTransport({
  service: 'qq',
  port: 465,                // SMTP 端口
  secure: true,             // 使用 SSL
  auth: {
    user: 'xxxx@qq.com',    // 发送邮件的邮箱
    pass: '*******'         // 邮箱密码
  }
});
var mailOptions = {
  from: 'xxxxxxx@qq.com',    // 发件地址
  to: 'xxxxxxxx@qq.com',     // 收件列表
  subject: '测试云函数',      // 标题
  text: '测试云函数'
};
// 云函数入口函数
exports.main = async (event, context) => {
  console.log("Start to sendemail")
  //开始发送邮件
  const info = await transporter.sendMail(mailOptions);
  console.log('Message sent: ' + info.response);
  return info
}
```

> **注意**
> 在该段代码中，需要填写您实际的邮箱地址、密码、想要发送的邮箱地址等信息。

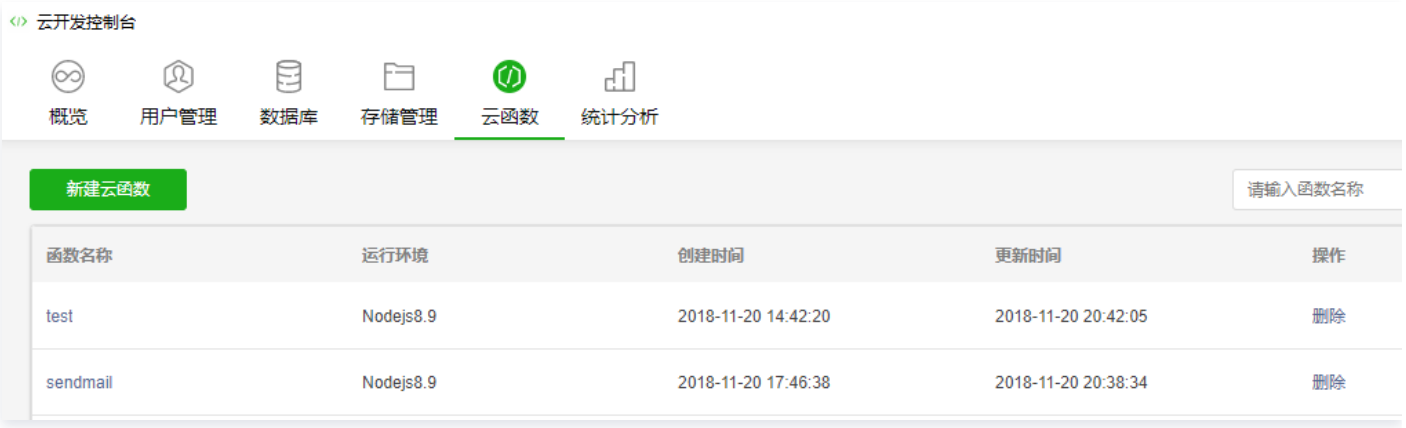
10. 使用 npm 工具安装或者在 package.json 中声明第三方 “nodemailer” 库。 package.json 声明如下所示：

```
{
 "name": "sendmail",
 "version": "1.0.0",
 "description": "",
```

```
  "main": "index.js",
  "scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
   },
  "author": "",
  "license": "ISC",
  "dependencies": {
  "wx-server-sdk": "latest",
  "nodemailer":"^4.7.0"
   }
}
```

11. 右键单击 sendemail 函数，选择 “上传并部署（云端安装依赖）”，将修改后的代码同步到云端。

12.切换至**云开发控制台**，选择 “云函数”，查看新创建的函数是否显示在列表中。如下图所示，即表示 test 函数和 sendemail 函数已经在云端部署成功。



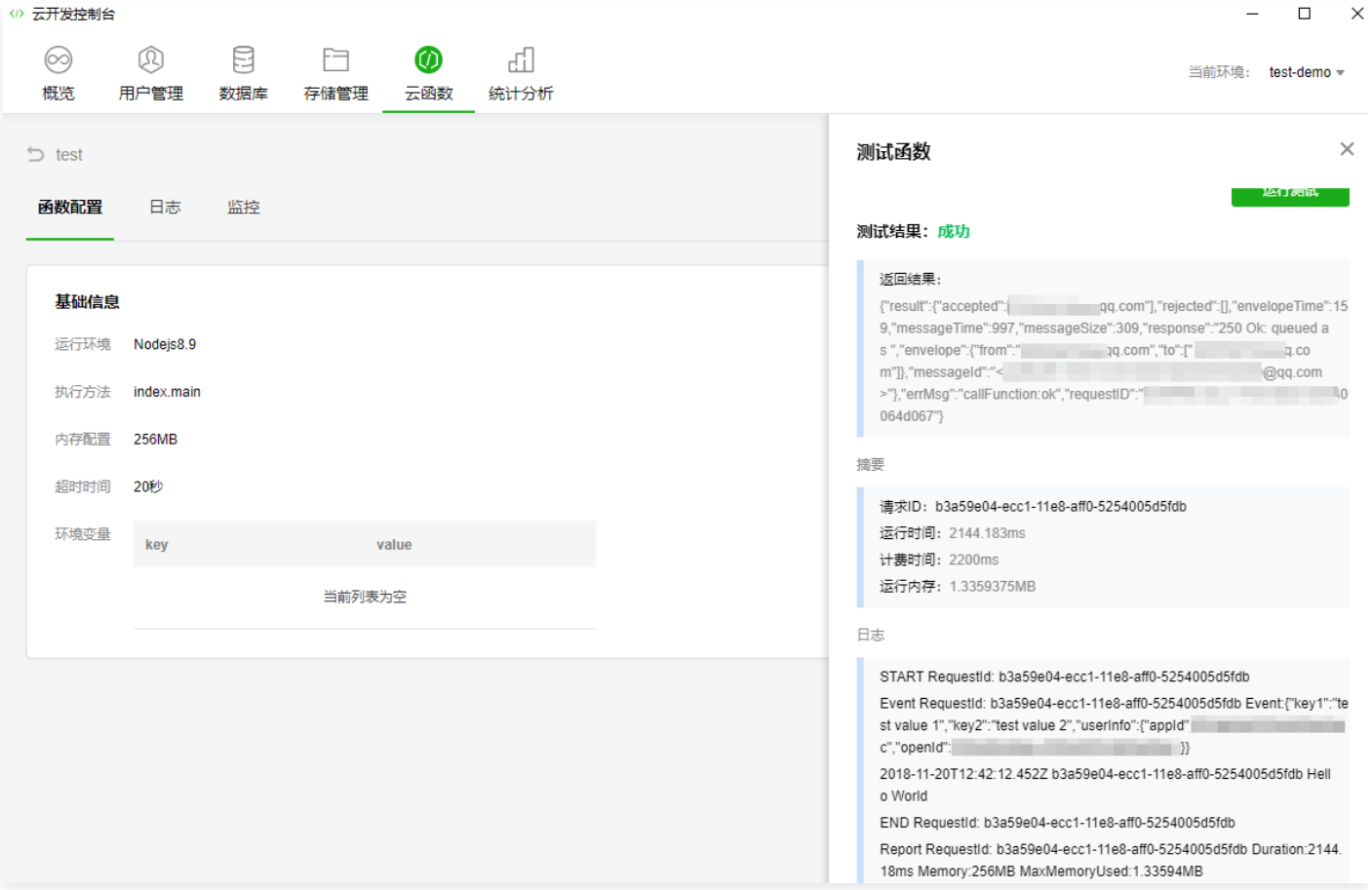
函数测试

最近更新时间：2022-04-18 10:47:15

本示例主要演示了函数之间的互相调用，以及通过函数发送邮件。您可以根据自身业务的需要，自行调整代码逻辑。

运行 test 函数

1. 在 “云开发控制台” 中，选择**云函数**，单击 “test” 函数，进入 “test” 函数信息页面。
2. 在 “test” 函数信息页中，单击**测试**，查看效果。如下图所示：



查看 sendemail 的运行日志

1. 在云函数信息页中，选择 “sendemail” 函数，进入 “sendemail” 函数信息页面。
2. 选择 “日志” 页签，查看该函数的执行日志。
   同时，您还可以前往您配置的邮箱，查看收到的邮件。

实时音视频 TRTC
SCF + TRTC 输入在线媒体流

最近更新时间：2024-09-09 14:21:14

使用场景

案例

AI 互动课堂

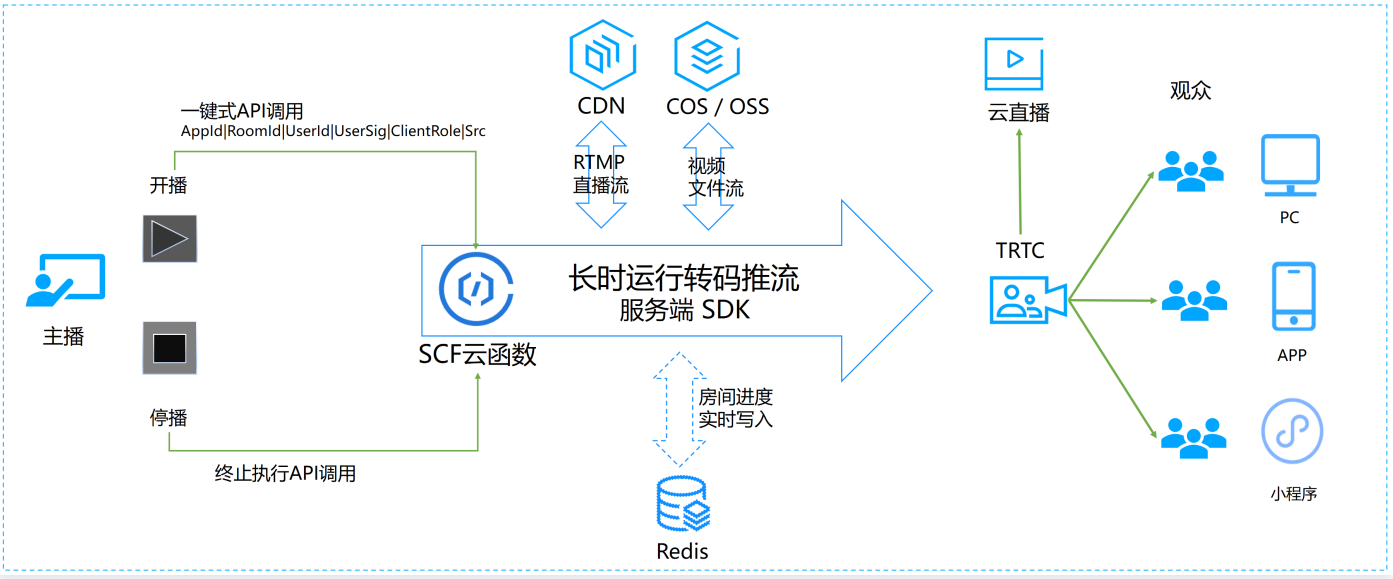
通过录播真人教学视频结合 AI 技术进行线上直播互动教学。
上课前，根据教师的课程设置，将知识点讲解、互动提问、问题反馈和解答等信息录制成视频片段，上传到视频库。课堂中，通过云函数将已有的录播视频推送到 TRTC 房间进行直播。学生通过语音、触屏实现互动式学习。服务端通过 AI 技术，智能识别学生的实时语音和作答，并根据学生的表现，无缝切换教学片段，实时给予不同的反馈，从而提供个性化的教学体验。

“一起看”房间服务

游戏直播、秀场、体育赛事等直播类内容，可以通过云函数将 RTMP 直播流推送到 TRTC 房间，实时交流，带动热点。电影、音乐等点播类节目，可以通过云函数将媒体文件转换为在线媒体流输入至TRTC房间，增值服务，打造社区圈层。云函数一键触发、免运维、弹性伸缩等特性可以快速支撑实时互动娱乐社交应用的构建。云函数的可编程性，可以快速整合其他云服务及三方服务，扩展业务边界，高效创新玩法。

业务流程

本文为您介绍如何 使用 API 网关集成云函数，将已有的录播视频或者 RTMP 直播流推送到实时音视频 TRTC 房间进行直播，提供开箱即用、灵活便捷、可编程的在线媒体流输入能力。如您需开启推流直播的实时记录，可以选择使用 Redis，API 网关会将进度实时写入 Redis。工作流程如下图所示：



API 网关调用涉及的参数如下：

参数名称	类型	必选	描述
VideoSrc	String	是	被推流的视频源，例如 `https://test-123456789.cos.ap-shanghai.myqcloud.com/video/1.mp4` 。
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，则使用 RoomId。
Mode	String	否	vod：点播模式，即推流为某个录制好的文件，默认模式。 live：直播模式，即推流为 rtmp 直播源。

UserId	String	是	推流用户 ID，用于在一个 TRTC 应用中唯一标识一个用户。
UserSig	String	是	推流用户签名，用于对一个用户进行登录鉴权认证。
Redis	Boolean	否	是否使用 Redis，默认为 false。
RedisHost	String	否	Redis 为 true 时，redis 的 host 地址。
RedisPort	Integer	否	Redis 为 true 时，redis 的访问端口号。
RedisPassword	String	否	Redis 为 true 时，redis 的访问密码。

说明：

- 如果 Redis 值为 false，从 VideoSrc 视频源拉流进行直播推流，直播流将从最新开始。
- 如果 Redis 值为 true，对于同一个 VideoSrc 视频源，API 网关将先在 Redis 中查询是否有上一次直播流推流记录：
- 若存在记录，则恢复上一次推流。
- 若无记录，则重新开始推流。直播推流进度通过回调实时写入 Redis。

操作步骤

创建云函数

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在函数服务页面上方选择**北京**地域，并单击**新建**进入新建函数页面，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**模板创建**。
- **模糊搜索**：输入“TRTC直播推流”，并进行搜索。
  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。

3. 单击**下一步**，根据页面相关信息提示进行配置。如下图所示：

- **函数名称**：默认填充。
- **异步执行**：勾选以开启。开启后，函数将以异步执行模式响应事件，事件调用无需阻塞等待处理结果，事件将在被调用后进入异步执行状态。
- **状态追踪**：勾选以开启。开启后，针对异步执行的事件，将开始记录响应事件的实时状态，并提供事件的统计、查询及终止服务，产生的事件状态数据将为您保留3天。
- **执行超时时间**：可根据需要自行修改。

4. 配置 API 网关触发器，默认新建 API 服务，不开启集成响应。您也可以选择自定义创建，自定义创建时确保集成响应关闭。如下图所示：

5. 单击**完成**即可完成函数创建和 API 网关触发器创建。

6. 如需使用 Redis 实时记录推流进度，由于 Redis 只能私有网络访问，因此必须将云函数的 VPC 配置在与 Redis 在同一个私有网络下。如下图所示：

创建 TRTC 应用

1. 登录实时音视频控制台，选择左侧导航栏中的**开发辅助** > 快速跑通 Demo 。
2. 填写 Demo 名称，单击**创建**完成应用创建。您可以根据自己的客户端选择模板试运行。

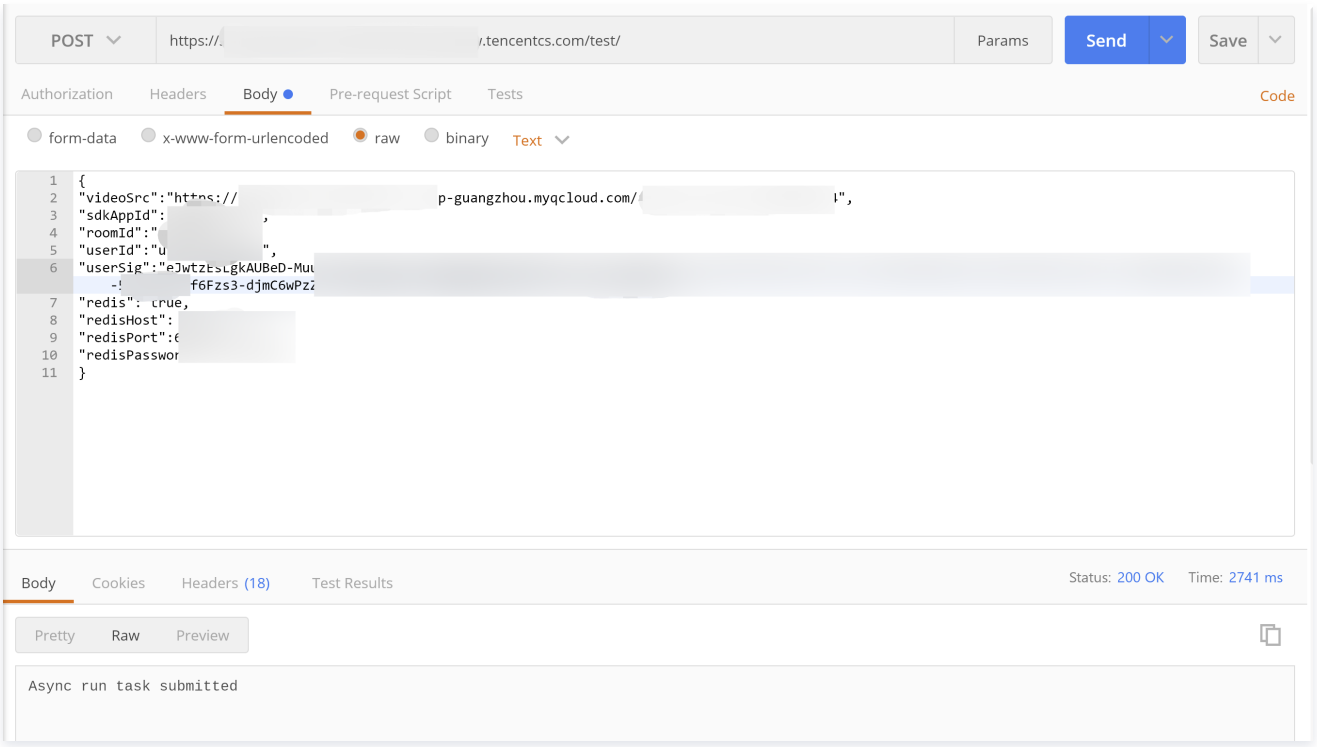
测试函数功能

1. 使用 Postman 构造 HTTP 请求。示例如下：

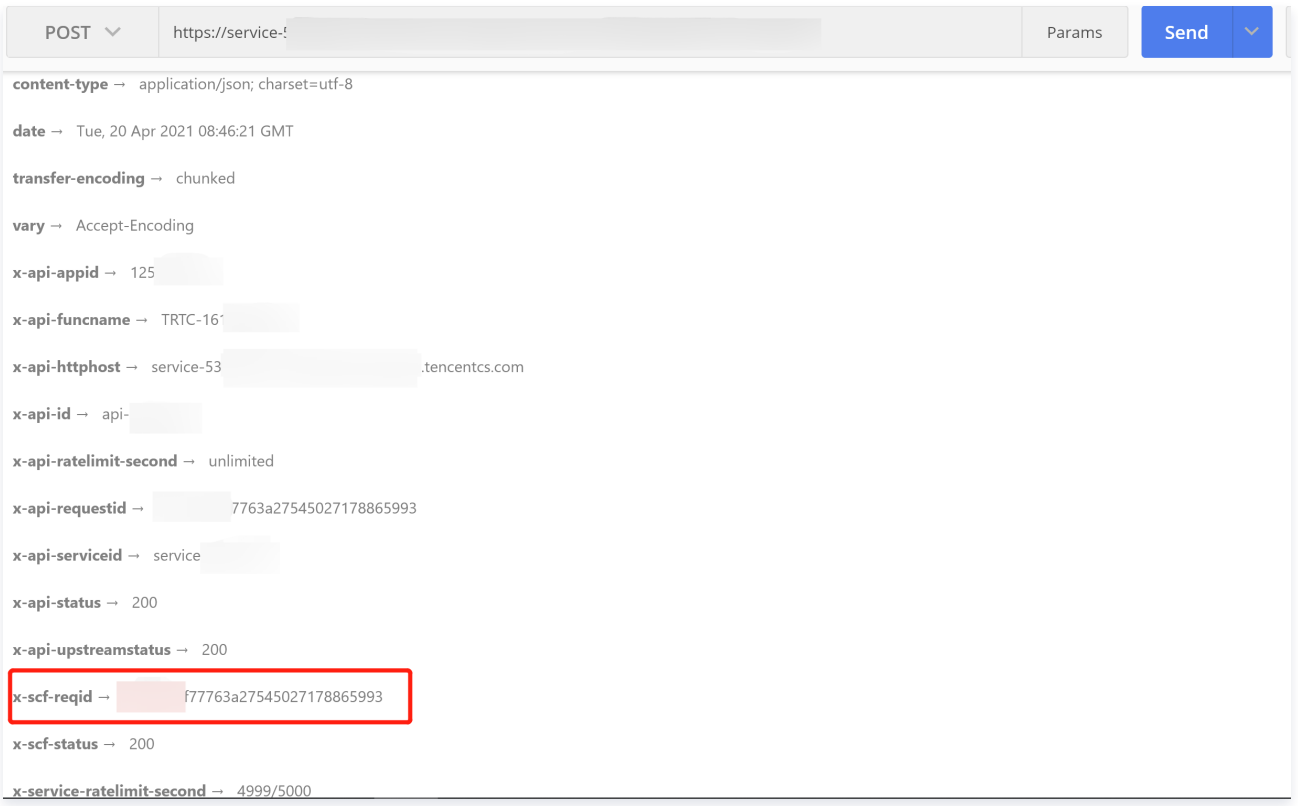
```
{
    "SdkAppId": 1400000000,
    "StrRoomId": "98915abc",
    "UserId": "user_55952144",
    "Mode": "vod",
    "UserSig": "eJwtzN0KgkAQBeB32dtCZqcd-
6CbMMRYIrGiu6jcZAhL1NogevdMvxxxxxxxxxxxxxxxxxAIdENM*c27uLV*552dj6iNRQCiVGgdNfjtVFecilApABSg9OTTmXXFtOic
iBIBBWy7xxxxxxxxRHbbjo-kFrN0UvOfuRH1tMyiDeLGcPu7ocxxxxxxxxxxx5uL7A0DEMb8_",
    "Redis": false,
```

```
    "VideoSrc": "https://test-123456789.cos.ap-shanghai.myqcloud.com/video/1.mp4"
}
```

如下图所示：

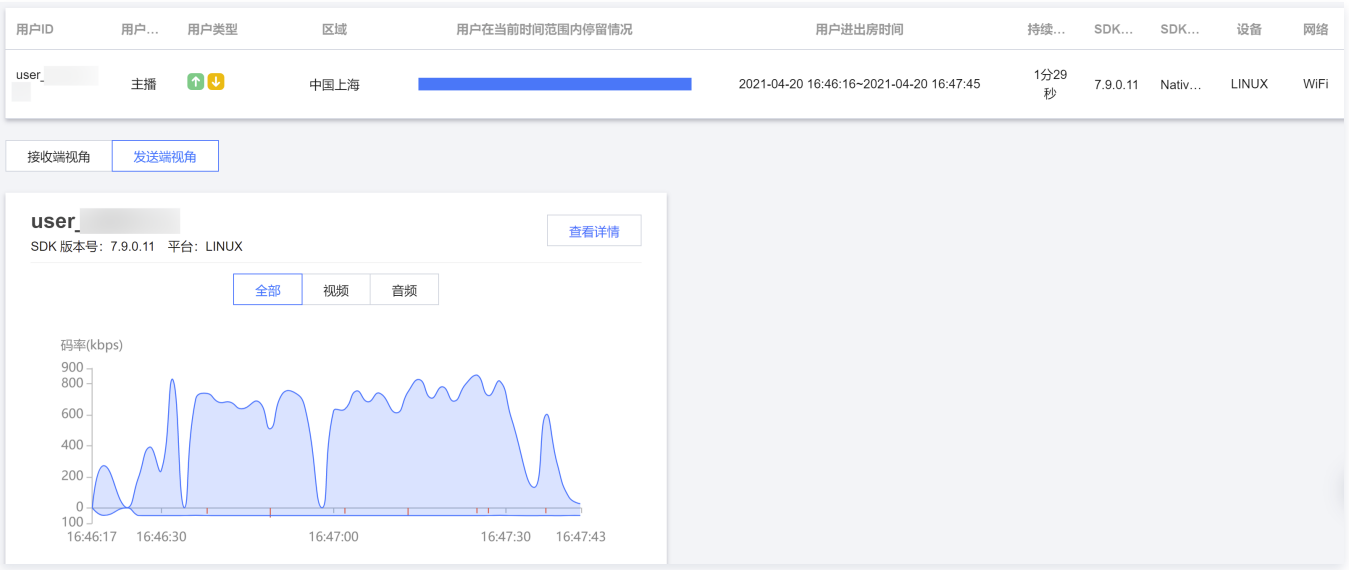


2. 请求发送后会收到异步函数响应 “Async run task submitted”，此次函数的 RequestId 会通过 HTTP 头部信息中的 x-scf-reqid 返回。如下图所示：



3. 在云函数控制台 函数服务 页面中，单击上述 创建云函数 步骤中创建的云函数名称，进入“函数详情”页面。
4. 在“函数详情”页面中选择**日志查询**页签，可以查看到打印出的推流日志信息。

5. 切换至 实时音视频控制台，在“监控仪表盘”页面单击房间 ID，查看推流监控详情信息。如下图所示：



6. 如需在推流过程中停止推流，可以调用 终止异步函数接口 InvokeRequestId 参数停止推流（**必须开启状态追踪**）。
其中 InvokeRequestId 可从上述步骤2的响应头部信息 x-scf-reqid 中获取。

SCF + TRTC 实现单流录制

最近更新时间：2022-12-21 15:55:28

使用场景

案例

在线教育

在一对一或一对多的小班课中，可以针对不同学生多维度进行录制：

- 对于单一学生，可以录制学生的单独数据流合成相关数据，实现记录每个学生的精彩瞬间并推送给家长。
- 对房间内数据进行定向录制，并生成回放，学生可以观看回放重复进行学习。
- 为了方便用户反复观看视频、重复学习，录制的过程可以去除冗余数据。

客服中心

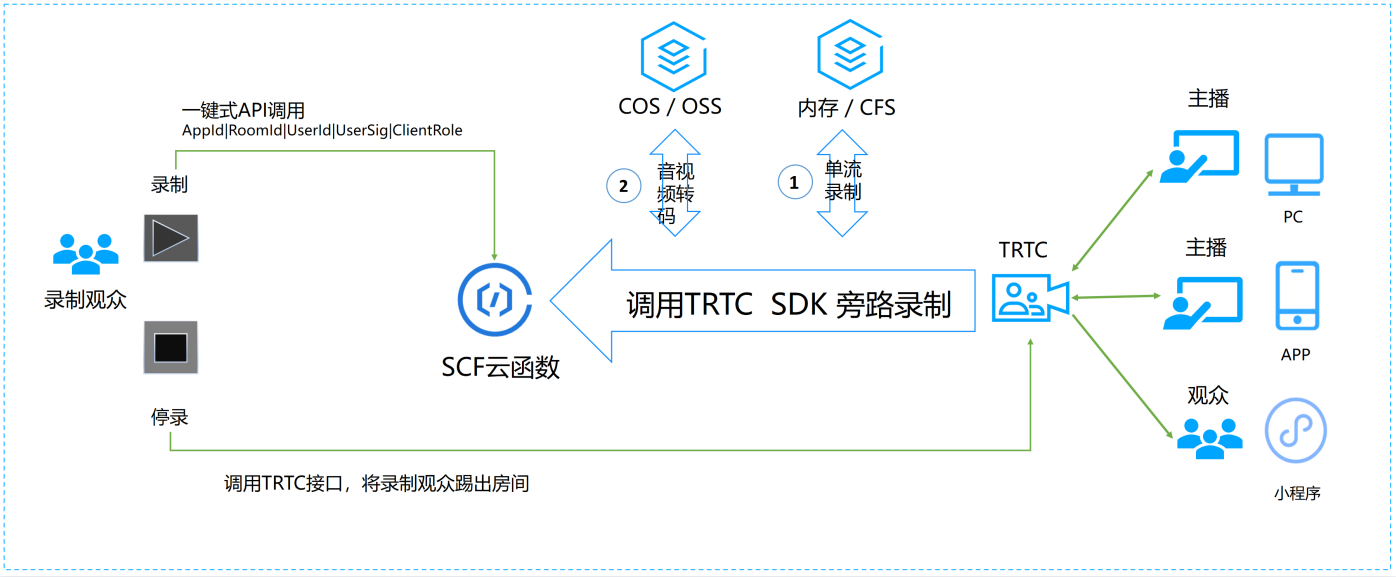
- 智能客服场景支持单独录制用户的数据流，和识别相关接口进行合成后，可实现办卡及智能开户过程的信息核实。
- 支持单独录制客服和用户的声音，自动识别关键字评估服务质量，对智能客服进行迭代训练。
- 可以将服务过程的数据进行保存与归档。

社交

- 仅录制房间需要录制的视频流，用做数据保存，以节省存储和混流的成本。
- 录制房间指定的数据，调用审核接口进行审核，可为不同的用户设定不同的审核标准。
- 可以将直播片段定向生成片段文件。

业务流程

本文为您介绍如何 使用 API 网关集成云函数，将实时音视频 TRTC 房间的主播音视频进行单流录制，录制完毕后上传到 COS 存储，提供开箱即用、灵活便捷、可编程的直播录制能力。云函数默认提供512MB内存来存储录制文件，如果您需要更大的存储空间，可以选择使用 CFS 挂载能力 。工作流程如下图所示：



API 网关调用涉及的参数如下：

参数名称	类型	必选	描述
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，则使用 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。
CosConfig	cosConfig	是	COS 存储配置。用于存储录制文件。
Callback	String	否	录制结束后的回调地址，并使用 POST 方式进行回调。
Mode	String	否	00：单流音频，输出 MP3 格式。默认模式。 01：单流视频，输出 MP4 格式。 02：单流音视频，输出 MP4 格式。

CosConfig 涉及的参数如下：

参数名称	类型	必选	描述
SecretId	String	否	腾讯云账号的 SecretId。详情请参见 访问管理 。
SecretKey	String	否	腾讯云账号的 SecretKey。详情请参见 访问管理 。
Region	String	是	COS 所在区。例如 `ap-guangzhou` 。
Bucket	String	是	桶名称。例如 `susu-123456789` 。
Path	String	是	桶内路径。例如 `/test` ，根目录为 `/` 。

说明
- UserId 为指定用户 ID， 多次请求 API 网关不保证幂等。
- CosConfig 中如果不配置 SecretId 与 SecretKey，函数访问 COS 时将使用运行角色 SCF_ExecuteRole 权限去执行。

停止录制的触发条件：
- TRTC 房间被销毁。当 TRTC 房间超过300s没有主播，房间会自动销毁。
- 主动调用移除用户接口，将录制观众移出房间。
- 使用 RoomId 的用户停止录制，需要调用 移除用户 接口。
- 使用 StrRoomId 的用户停止录制时，需要调用 移除用户（字符串房间号）接口。

停止录制后，函数返回数据格式如下：

参数名称	类型	必选	描述
SdkAppId	String	是	应用 ID。
RoomId	String	是	整型房间 ID。
UserId	String	是	录制用户 ID。
StrRoomId	String	是	字符串房间 ID。
Files	Array	是	[{},{},{},{}]

说明
如果配置了 Callback，停止结束后，云函数将以 POST 方式将返回数据传递给回调地址。

Files 数组中每一项为 JSON Object，如下：

参数名称	类型	必选	描述
UserId	String	是	被录制的用户 ID。
RecordFile	String	是	录制文件最后上传到 COS 的 URL。

Status	Int	是	0：失败。 1：成功。
Message	String	是	录制任务的执行结果。例如，录制失败、转码失败、写入 COS 失败等。

操作步骤

创建云函数

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”页面上方选择**广州**地域，并单击**新建**进入新建函数页面，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**模板创建**。
- **模糊搜索**：输入“TRTC”进行搜索，选择**单流音视频录制**。
  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。

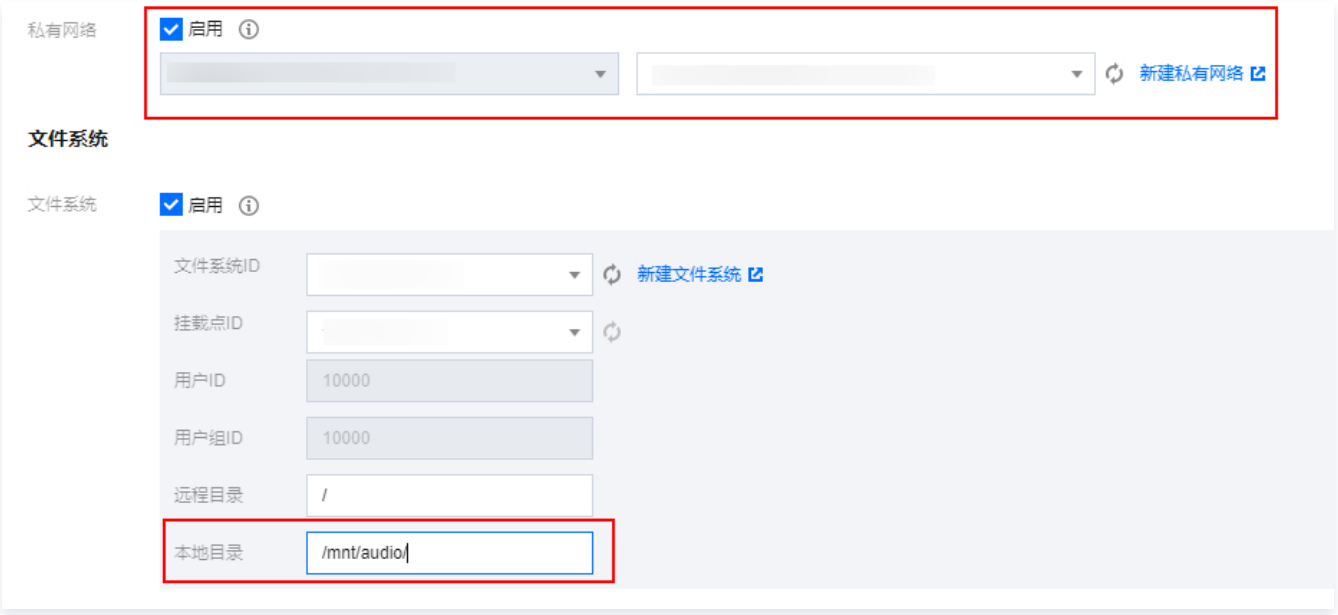
3. 单击**下一步**，根据页面相关信息提示进行配置。如下图所示：

- **函数名称**：默认填充。
- **异步执行**：勾选以开启。开启后，函数将以异步执行模式响应事件，事件调用无需阻塞等待处理结果，事件将在被调用后进入异步执行状态。
- **状态追踪**：勾选以开启。开启后，针对异步执行的事件，将开始记录响应事件的实时状态，并提供事件的统计、查询及终止服务，产生的事件状态数据将为您保留3天。
- **执行超时时间**：可根据需要自行修改。
- **运行角色**：默认使用 SCF_ExecuteRole 作为运行角色，并授予 QcloudCOSFullAccess、QcloudCFSFullAccess 访问权限。

4. 配置 API 网关触发器，默认新建 API 服务，不开启集成响应。您也可以选择自定义创建，自定义创建时确保集成响应关闭。如下图所示：

5. 单击**完成**即可完成函数创建和 API 网关触发器创建。

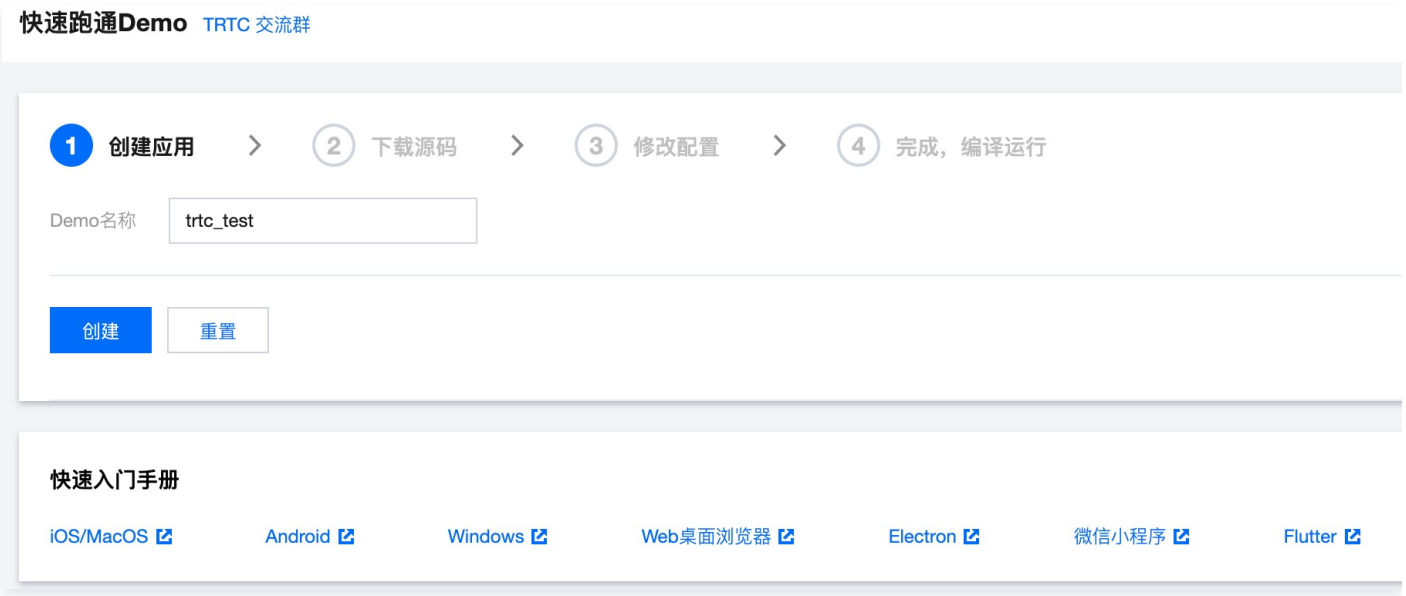
6. 如需使用 CFS 挂载能力，由于 CFS 只能私有网络访问，因此必须将云函数的 VPC 配置在与 CFS 在同一个私有网络下。如下图所示：



> **说明**
> 启用 CFS，需要将环境变量 CFS_PATH 设置为本地目录，例如 `/mnt/audio/` 。

创建 TRTC 应用

1. 登录实时音视频控制台，选择左侧导航栏中的**开发辅助** > 快速跑通 Demo 。
2. 填写 Demo 名称，单击**创建**完成应用创建。您可以根据自己的客户端选择模板试运行，例如 跑通Demo(桌面浏览器) 。



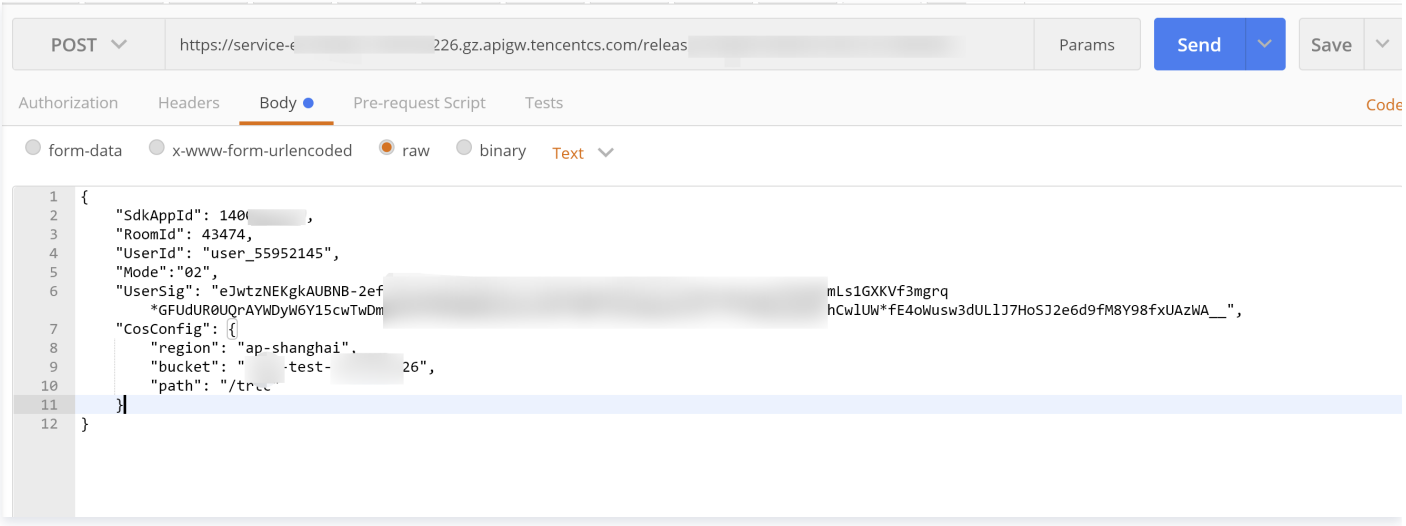
测试函数功能

1. 创建 TRTC 应用 并进入应用。
2. 使用 Postman 构造 HTTP 请求。其中 roomId 为已创建 TRTC 应用的房间号，userId 为随机另一个用户 ID（必须唯一）。示例如下：

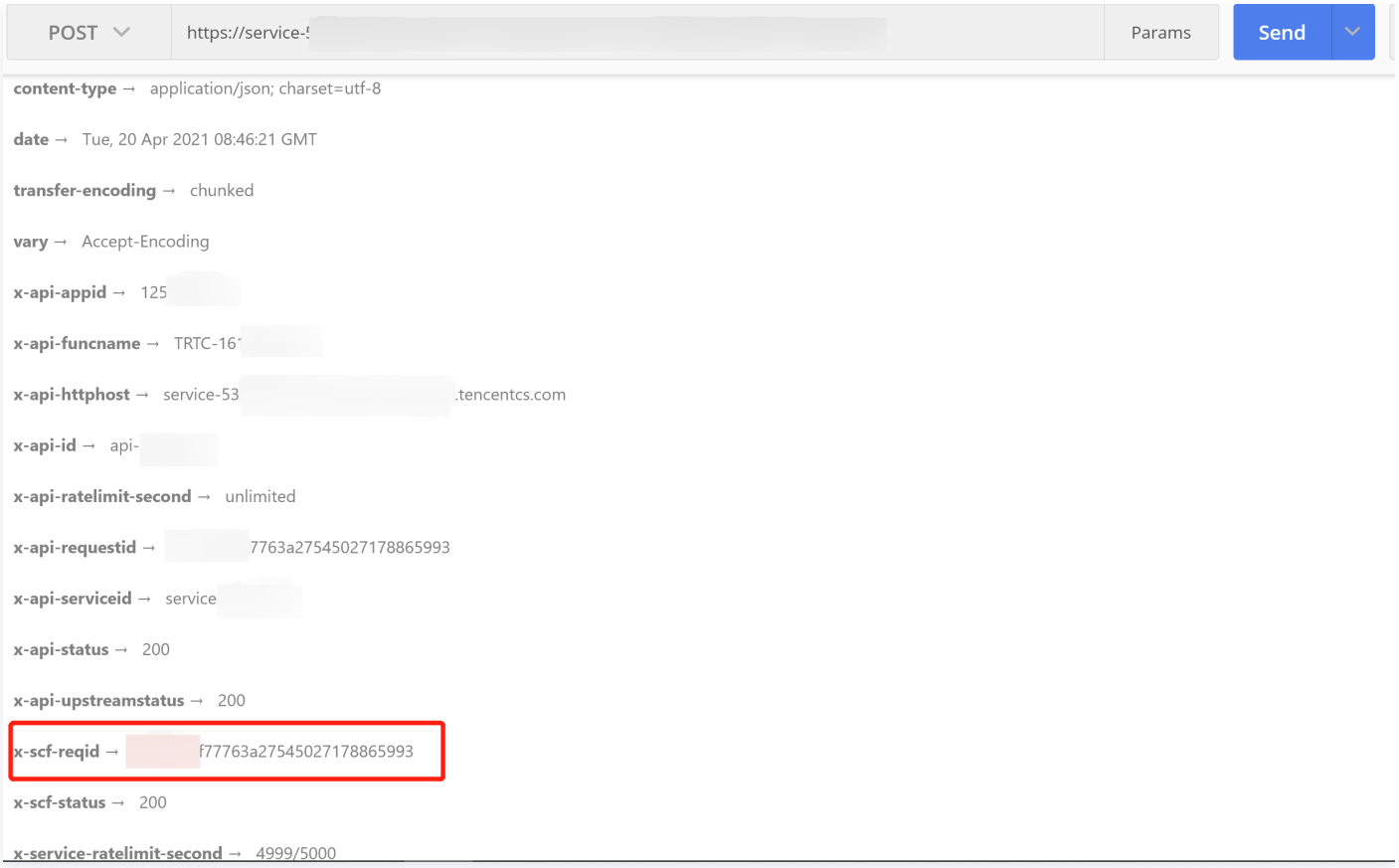
```
{
    "SdkAppId": 1400000000,
    "RoomId": 43474,
    "UserId": "user_55952145",
```

```
    "Mode": "02",
    "UserSig": "eJwtzNEKgkAUBNB-
2efQ3e3eUqG3tMCKJJEIIxxxxxxxxxxxxxxxhvmweLWzGlUxj0mLs1GXKVf3mgrq*GFUdUR0UQrAYWDyW6Y15cwTwDm4UkxF36iXpkq
1joiSc9xxxxxxxxxxxx-S*CZeOk9sHfnEhCwlUW*fE4oWusw3dULlJ7HoSJ2e6d9fM8Y98fxUAzWA__",
    "CosConfig": {
        "Region": "ap-shanghai",
        "Bucket": "test-123456789",
        "Path": "/trtc"
        }
}
```

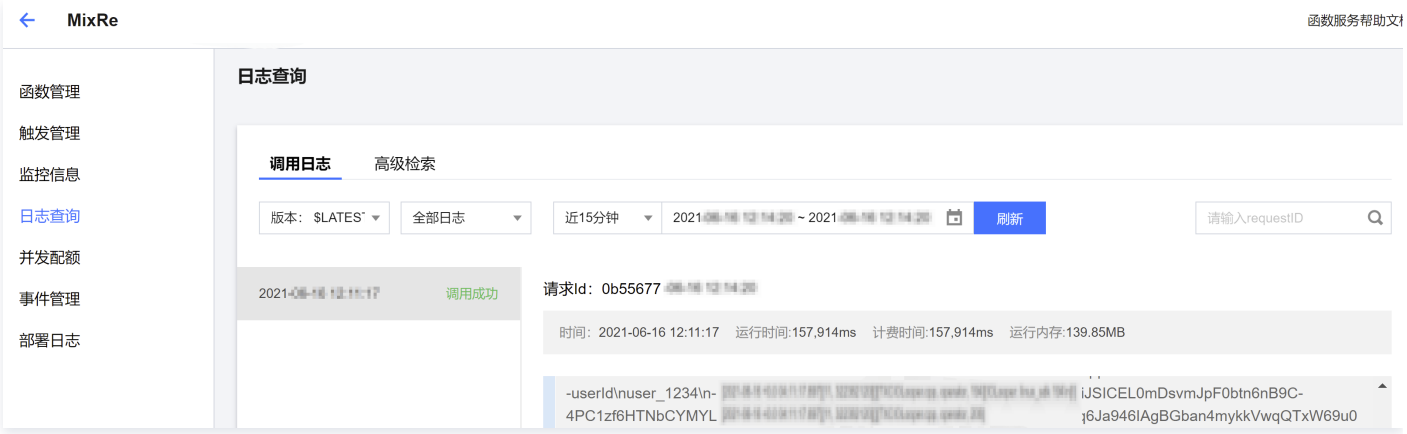
如下图所示：



3. 请求发送后会收到异步函数响应 “Async run task submitted”，此函数的 RequestId 会通过 HTTP 头部信息中的 x-scf-reqid 返回。如下图所示：



4. 在云函数控制台 函数服务 页面中，单击上述 创建云函数 步骤中创建的云函数名称，进入“函数详情”页面。
5. 在“函数详情”页面中选择**日志查询**页签，可以查看到打印出的录制日志信息。如下图所示：



6. 切换至 实时音视频控制台，在“监控仪表盘”页面单击房间 ID，查看所有在房间中的用户，其中一个观众就是我们的录制观众。如下图所示：

7. 如需在录制过程中停止录制，可以调用 移除用户接口 或者 移除用户（字符串房间号）接口 将用户移出房间。

SCF + TRTC 实现混流录制

最近更新时间：2023-08-21 09:41:22

使用场景

案例

在线教育

在线教育的场景中，可以实现在上课的过程中将老师的音视频和学生的音视频进行合成录制，并且加入上课过程中的其他素材与人工智能分析，在真实还原上课场景的同时增加一些业务功能，增加回放视频观看效果。

社交直播

在直播过程，可以采取混流录制的方法将多流合成、旁路审核，从而降低审核的成本，并且根据国家的要求将混流之后录制文件存储，应对监管要求。

金融监管

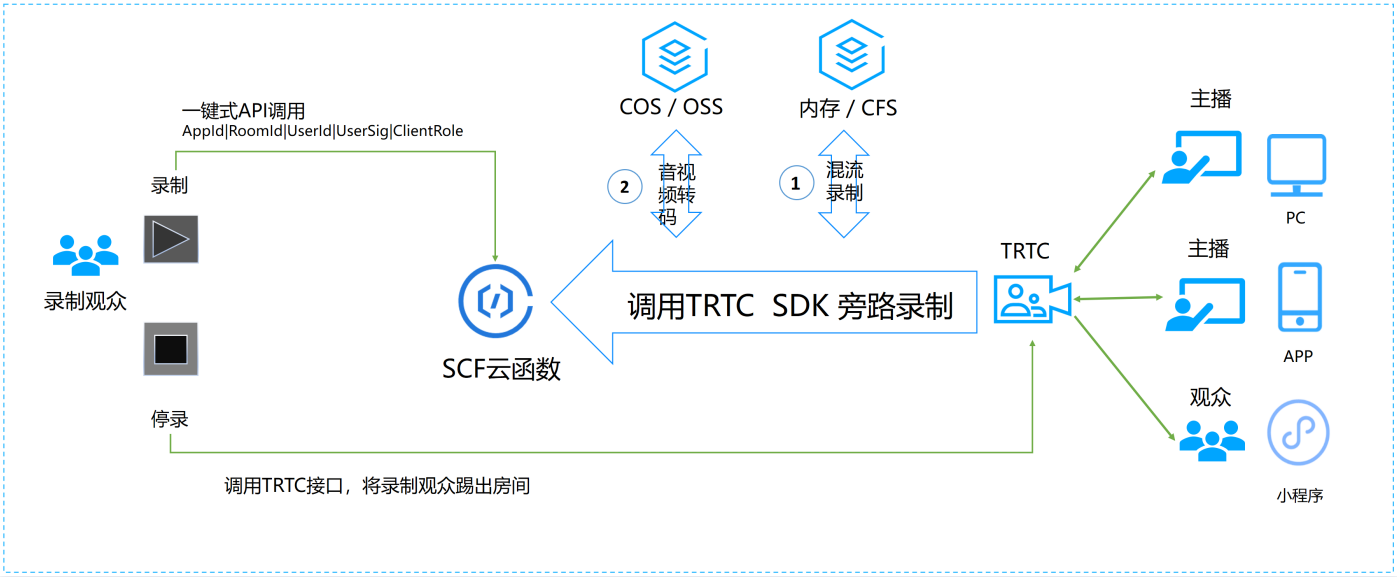
在线开户，金融双录等场景中，以全局视角录制全部交易过程，并进行存档保留，满足后续服务或者监管要求。

其他

客户服务、投诉等场景，可以实时启动录制，录制服务过程与投诉内容，便于优化服务质量，排查投诉合理性，提升投诉处理效率。

业务流程

本文为您介绍如何 使用 API 网关集成云函数，将实时音视频 TRTC 房间的主播音视频进行混流录制，录制完毕后上传到COS存储，提供开箱即用、灵活便捷、可编程的直播录制能力。云函数默认提供 512MB 内存来存储录制文件，如果您需要更大的存储空间，可以选择使用 CFS 挂载能力 。工作流程如下图所示：



录制规则如下：

- 最多只会录制6路音视频流。
- 当房间低于6个人时，会自动混流后续加入的用户。当房间超过6个人时，只会录制最先进房的6个人。
- IsReserve 为 true 时，在用户退出房间后，该用户流仍然在混流任务中，混流任务中将以最后一帧画面和静音补全流。IsReserve 为 false 时，在用户退出房间后，该用户流将从混流任务中删除，如果后续有新用户进入，则以新用户的流加入混流中。

API 网关调用涉及的参数如下：

参数名称	类型	必选	描述
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。
CosConfig	cosConfig	是	COS 存储配置。用于存储录制文件。
Callback	String	否	录制结束后的回调地址，并使用 POST 方式进行回调。
Mode	String	否	10：混流音频，默认模式。输出 MP3 格式。 11：混流视频。输出 MP4 格式。 12：混流音视频，输出 MP4 格式。
IsReserve	Boolean	否	false，主播退出房间，自动删除混流中的流。 true，主播退出房间，混流任务中将以最后一帧画面和静音补全流。默认值为 true。

CosConfig 涉及的参数如下：

参数名称	类型	必选	描述
SecretId	String	否	腾讯云账号的 SecretId。详情请参见 访问管理 。
SecretKey	String	否	腾讯云账号的 SecretKey。详情请参见 访问管理 。
Region	String	是	COS 所在区。例如 `ap-guangzhou` 。
Bucket	String	是	桶名称。例如 `susu-123456789` 。
Path	String	是	桶内路径。例如 `/test` ，根目录为 `/` 。

说明
- UserId 为指定用户 ID， 多次请求 API 网关不保证幂等。
- CosConfig 中如果不配置 SecretId 与 SecretKey，函数访问 COS 时将使用运行角色 SCF_ExecuteRole 权限去执行。

停止录制的触发条件：
- TRTC 房间被销毁。当 TRTC 房间超过300s没有主播，房间会自动销毁。
- 主动调用移除用户接口，将录制观众移出房间。
- 使用 RoomId 的用户停止录制，需要调用 移除用户 接口。
- 使用 StrRoomId 的用户停止录制时，需要调用 移除用户（字符串房间号）接口。

停止录制后，函数返回数据格式如下：

参数名称	类型	必选	描述
SdkAppId	String	是	应用 ID。
RoomId	String	是	整型房间 ID。
UserId	String	是	录制用户 ID。
StrRoomId	String	是	字符串房间 ID。
Files	Array	是	[{},{},{},{}]

说明
如果配置了 Callback，停止结束后，云函数将以 POST 方式将返回数据传递给回调地址。

Files 数组中每一项为 JSON Object，如下：

参数名称	类型	必选	描述
UserId	String	是	被录制的用户 ID。
RecordFile	String	是	录制文件最后上传到 COS 的 URL。
Status	Int	是	0：失败。 1：成功。
Message	String	是	录制任务的执行结果。例如，录制失败、转码失败、写入 COS 失败等。

操作步骤

创建云函数

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”页面上方选择**广州**地域，并单击**新建**进入新建函数页面，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**模板创建**。
- **模糊搜索**：输入“TRTC”进行搜索，选择**混流音视频录制**。
  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。

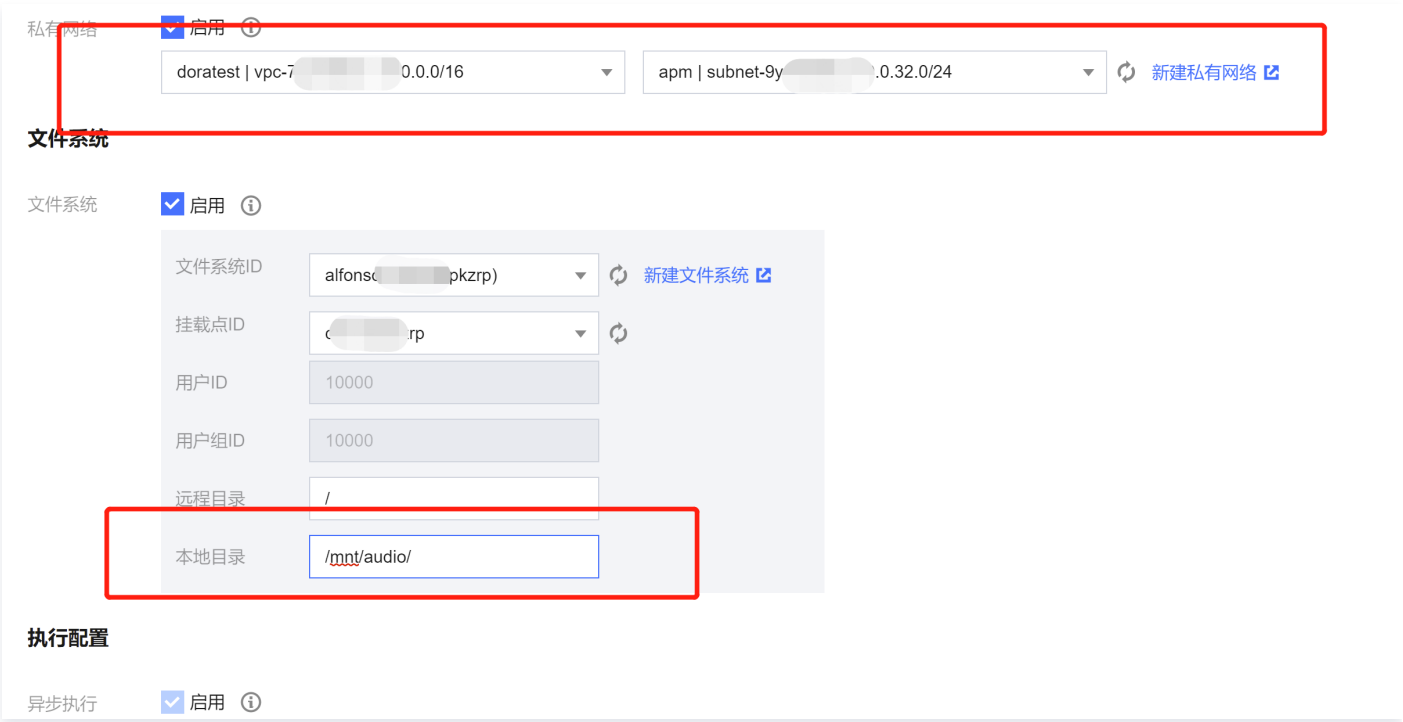
3. 单击**下一步**，根据页面相关信息提示进行配置。如下图所示：

- **函数名称**：默认填充。
- **异步执行**：勾选以开启。开启后，函数将以异步执行模式响应事件，事件调用无需阻塞等待处理结果，事件将在被调用后进入异步执行状态。
- **状态追踪**：勾选以开启。开启后，针对异步执行的事件，将开始记录响应事件的实时状态，并提供事件的统计、查询及终止服务，产生的事件状态数据将为您保留3天。
- **执行超时时间**：可根据需要自行修改。
- **运行角色**：默认使用 **SCF_ExecuteRole** 作为运行角色，并授予 **QcloudCOSFullAccess**、**QcloudCFSFullAccess** 访问权限。

4. 配置 API 网关触发器，默认新建 API 服务，不开启集成响应。您也可以选择自定义创建，自定义创建时确保集成响应关闭。如下图所示：

5. 单击**完成**即可完成函数创建和 API 网关触发器创建。

6. 如需使用 CFS 挂载能力，由于 CFS 只能私有网络访问，因此必须将云函数的 VPC 配置在与 CFS 在同一个私有网络下。如下图所示：

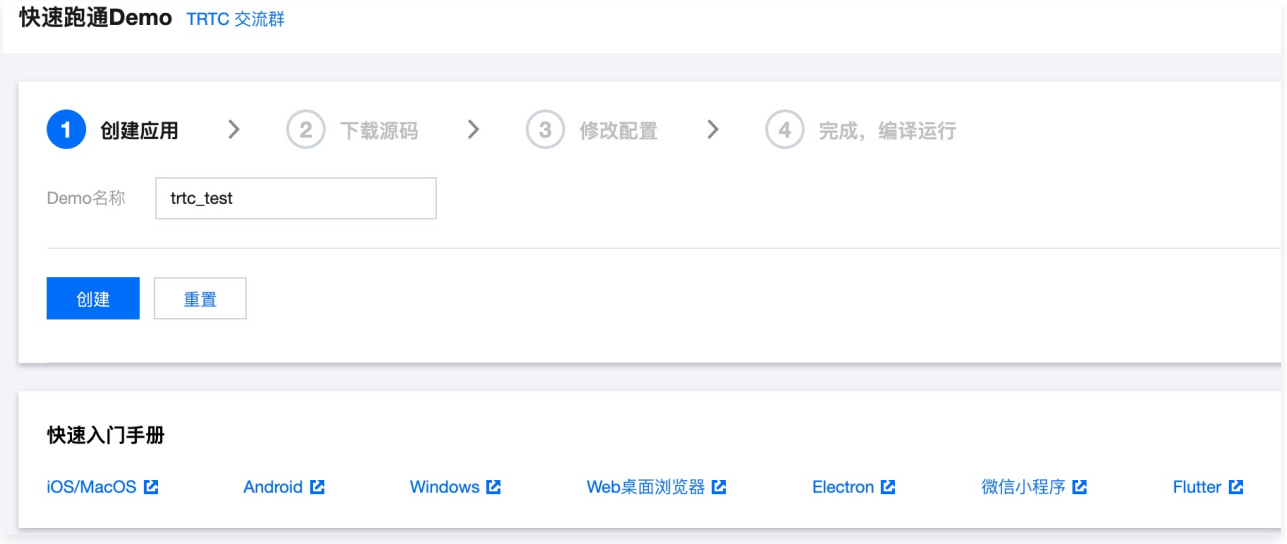


说明

启用 CFS，需要将环境变量 CFS_PATH 设置为本地目录，例如 `/mnt/audio/` 。

创建 TRTC 应用

1. 登录实时音视频控制台，选择左侧导航栏中的**开发辅助** > 快速跑通 Demo 。
2. 填写 Demo 名称，单击**创建**完成应用创建。您可以根据自己的客户端选择模板试运行，例如 跑通Demo(桌面浏览器) 。



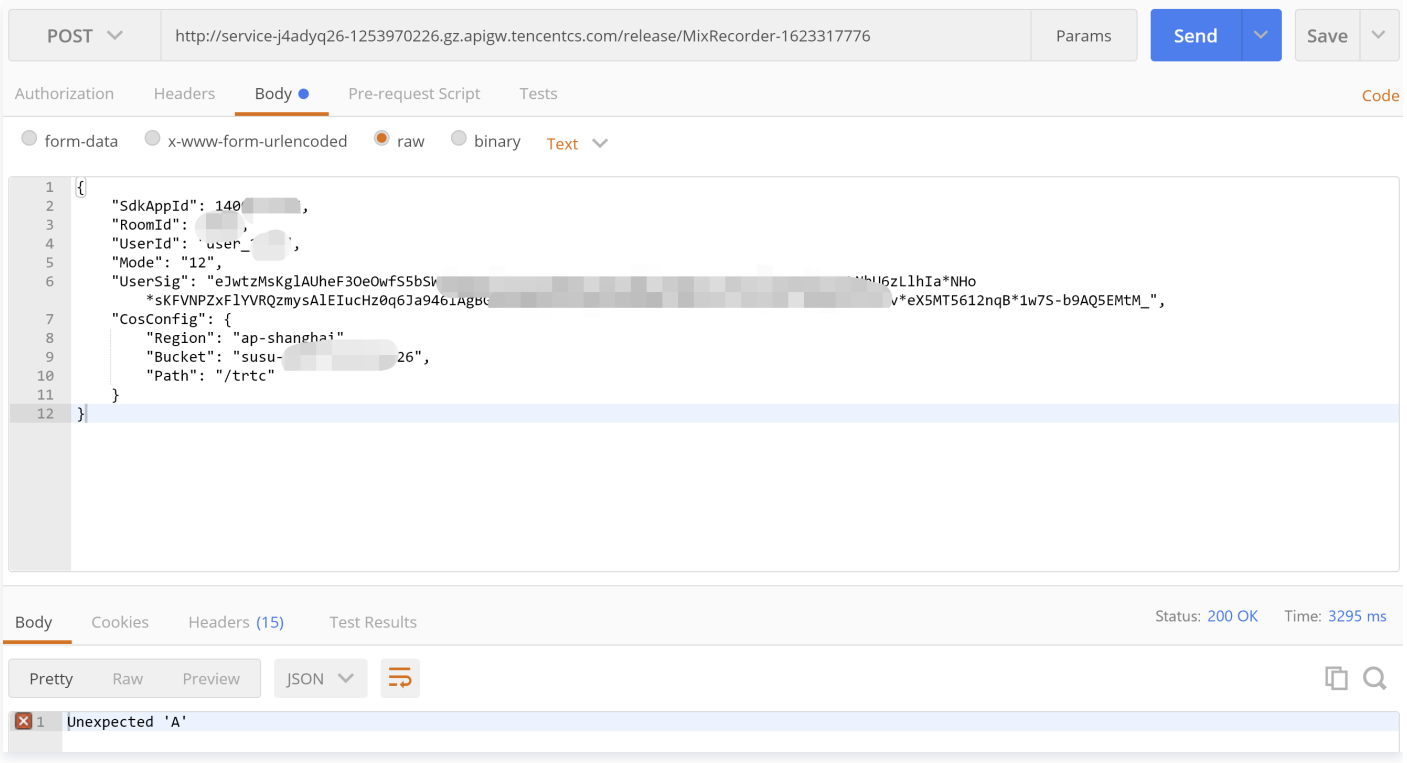
测试函数功能

1. 参考 跑通Demo(桌面浏览器)，用户user_00001与user_00002进入一个 TRTC 房间。
2. 使用 Postman 构造 HTTP 请求。其中 roomId 为已创建 TRTC 应用的房间号，userId 为随机另一个用户 ID（必须唯一）。如下图所示：

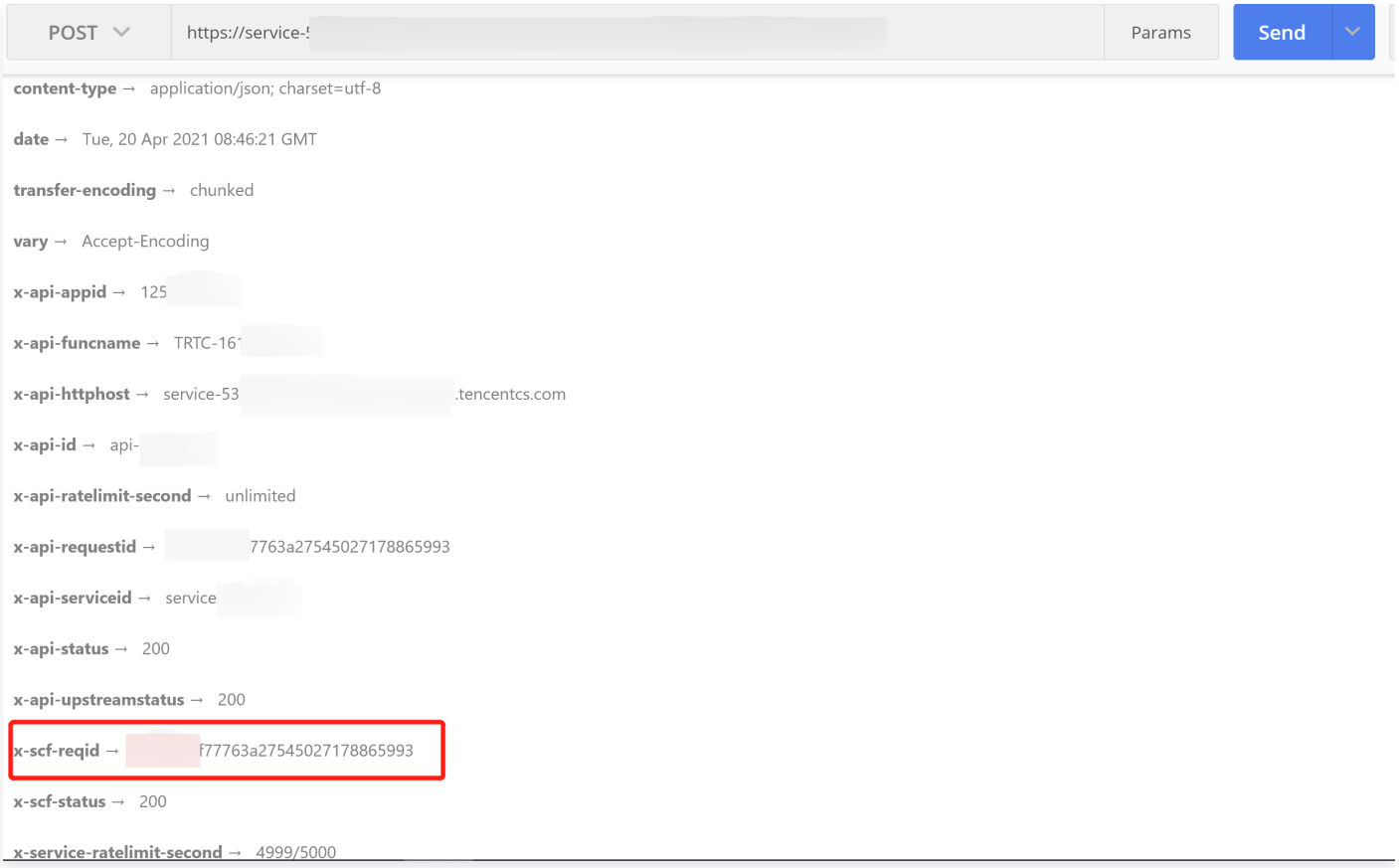
```
{
    "SdkAppId": 1400000000,
    "RoomId": 43474,
```

```
    "UserId": "user_55952145",
    "Mode":"12",
    "UserSig": "eJwtzNEKgkAUBNB-
2efQ3e3eUqG3tMCKJJEIIxxxxxxxxxxxxxxxhvmweLWzGlUxj0mLs1GXKVf3mgrq*GFUdUR0UQrAYWDyW6Y15cwTwDm4UkxF36iXpkq
1joiSc9xxxxxxxxxxxxx-S*CZeOk9sHfnEhCwlUW*fE4oWusw3dULlJ7HoSJ2e6d9fM8Y98fxUAzWA__",
    "CosConfig": {
      "Region": "ap-shanghai",
      "Bucket": "test-123456789",
      "Path": "/trtc"
      },
    "IsReserve":false,
    "Callback":"https:xxxxxxxx.com/post/xxx"
}
```

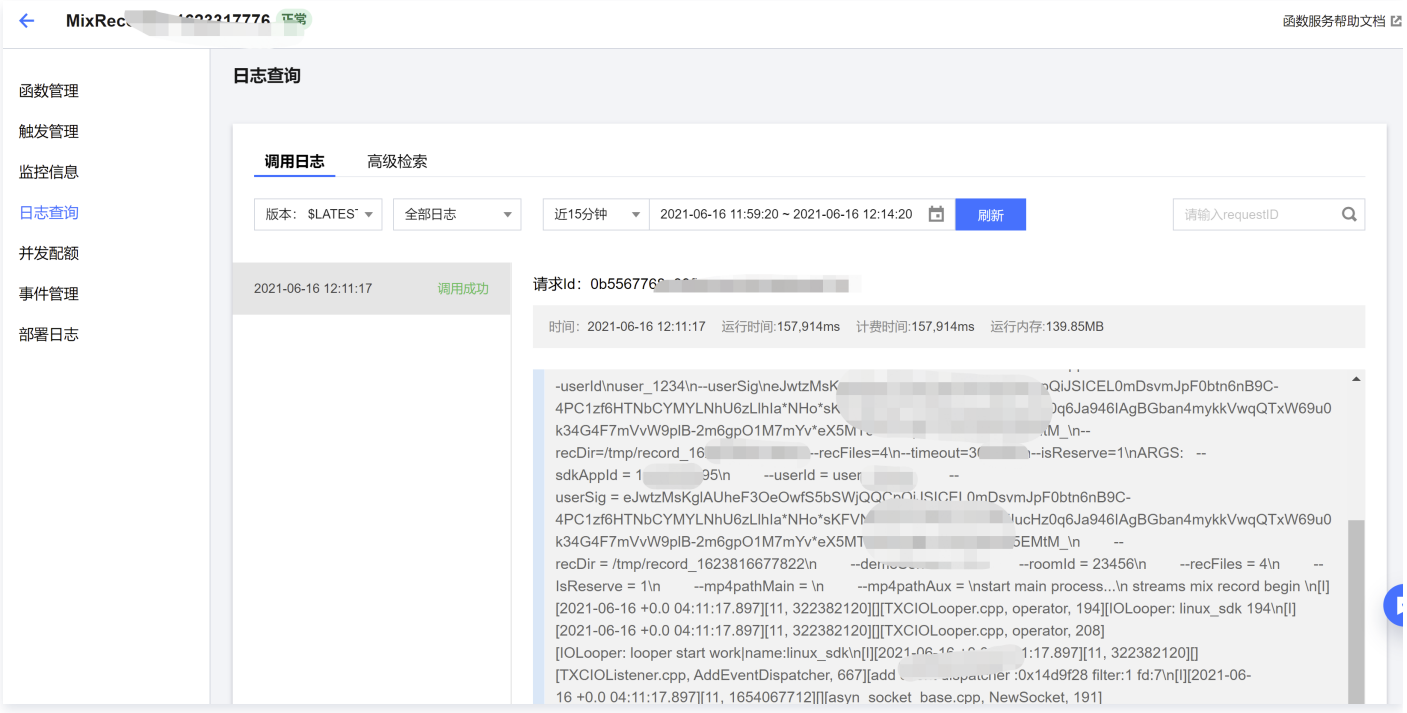
如下图所示：



3. **请求发送后会收到异步函数响应 “Async run task submitted”，此函数的 RequestId 会通过 HTTP 头部信息中的 x-scf-reqid 返回。如下图所示：**



4. 在云函数控制台 函数服务 页面中，单击上述 创建云函数 步骤中创建的云函数名称，进入“函数详情”页面。
5. 在“函数详情”页面中选择**日志查询**页签，可以查看到打印出的录制日志信息。如下图所示：



6. 切换至 实时音视频控制台，在“监控仪表盘”页面单击房间 ID，查看所有在房间中的用户，其中一个观众即为录制观众。如下图所示：

7. 如需在录制过程中停止录制，可以调用 移除用户 或者 移除用户（字符串房间号） 将用户提出房间。

对象存储 COS
SCF + COS 实现实时音视频转码

最近更新时间：2022-04-18 10:47:54

操作场景

在视频、社交应用等场景下，用户上传的图片、音视频的总量大、频率高，对处理系统的实时性和并发能力都有较高的要求。而上传的视频短片，可对应不同的清晰度使用多个云函数对其分别处理，以满足不同场景下用户的需求，同时适应移动网络带宽较小且不稳定的特性。

运行原理

使用云函数、ffmpeg 及对象存储 COS 联动实现音视频转码的运行原理图如下：

在云函数中，可基于不同的编程语言（Python、Node、PHP、JAVA 及 GO）撰写自定义业务逻辑。以转码为例，操作步骤如下：

1. 创建函数，并在其中部署 ffmpeg 资源包及转码逻辑。
2. 配置 COS Bucket 触发器，对源视频实时处理加工。旁路生成日志和监控、支持告警。
3. 对转码后的视频回传 COS，并触发自动预热。

与容器服务的对比

与容器服务对比，使用云函数及 ffmpeg 实现音视频转码服务的优势和不足如下表：

对比项	基于容器的实现	基于云函数的实现	分析
实现方法	在 docker 容器中运行 ffmpeg。 ffmpeg 进行了自定义处理，不同版本具备不同的 binary 依赖，均打包为镜像。 自主控制 docker 镜像版本，不同服务加载不同的 docker 镜像，其中包含不同的 ffmpeg。	不同的 ffmpeg 编译为不同的可执行文件，并部署到云函数“层”，和业务逻辑解耦。 编写不同的函数，分别和“层”中不同的 ffmpeg 绑定，提供不同的转码功能。 编写调度函数，分片视频和调度不同的转码服务。	实现方案差异不大。
开发/测试/部署体验	本地开发测试后，触发 CI/CD 流程，烧制镜像，完成部署。 需要维护灰度发布系统，容器服务集群每次更新速度较慢。	本地开发测试后，触发 CI/CD 流程，完成部署。 云函数提供灰度/发布/回滚功能，部署流程可以直接集成 API 接口实现全自动化流程。 个人子账号可以通过 CLI 工具，在测试环境实时开发及调试，效率更高。	在开发流程方面，云函数更加简单高效。
日志/监控/告警	需要在容器集群里启动 Agent，并对接日志平台、监控中心及告警平台。	自带日志/监控/告警能力，同时开放 API 接口，可以对接第三方日志/监控平台。支持运行时启动 Agent 进程，同步上报数据。	云函数自带能力较完善，但如需对接自建平台，启动 Agent 相比容器较复杂。
运维	需自主维护容器集群，弹性伸缩效率较低。	无需维护，云函数保障集群可用性、负载均衡及弹性伸缩。	云函数更加易用。
费用	需根据峰值预留容器资源，存在资源浪费可能性。	根据实际流量弹性伸缩、计费，费用节省30%以上。	云函数具有明显优势。

通过以上与容器的多维度对比，可得出以下结论：

优势：

- 云函数提供标准运行环境，并且保障资源的高可用和弹性伸缩，无需专人维护。
- 云函数基于实际业务消耗收费，不存在资源浪费。

- 云函数的开发调试流程效率会更加高效，依赖和业务解耦，可以分别单独更新，支持实时热更新。
- 运行环境隔离，单次请求失败不影响其他请求的正常执行。

不足：

- 云函数的引入，需要对接现有 CI/CD 流程，开发方式上有一定的转变。
- 现有业务代码需进行一定程度的改造，主要集中在 ffmpeg 编包上。云函数可以提供编包工具及相关研发协助改造。

前提条件

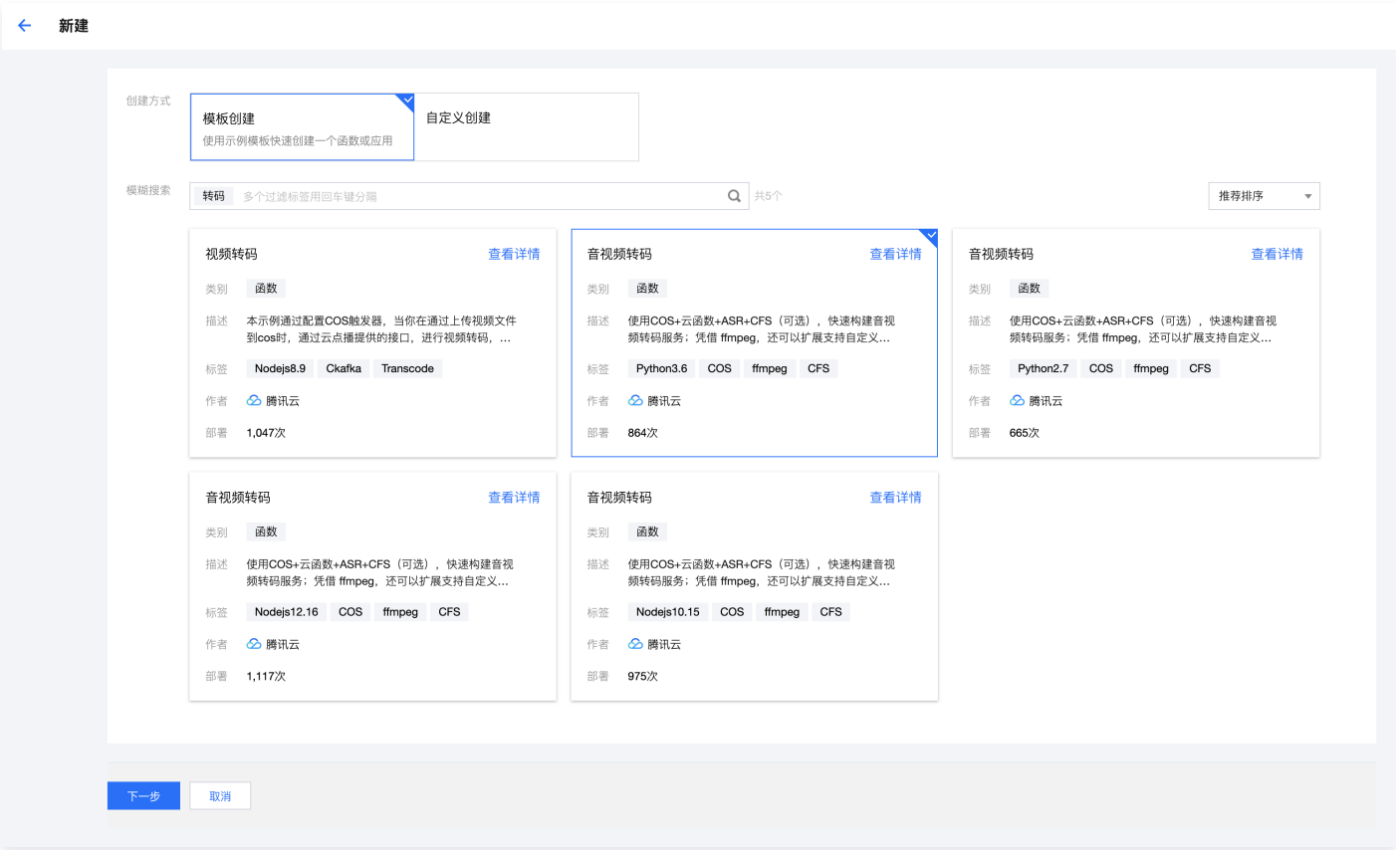
本文以广州地域为例：

- 前往 对象存储控制台 创建 COS Bucket，且 Bucket 权限设置为**公有读私有写**。
- （可选）当视频文件大于500M时，需前往文件存储控制台开通 CFS 服务，用于扩展云函数的本地存储空间。详情请参见 挂载 CFS 文件系统 。
- 前往访问管理控制台，创建云函数的运行角色，并授予该角色 COS、CFS 的读写权限，用于授权云函数访问相应服务。详情请参见 创建函数运行角色 。

操作步骤

创建云函数

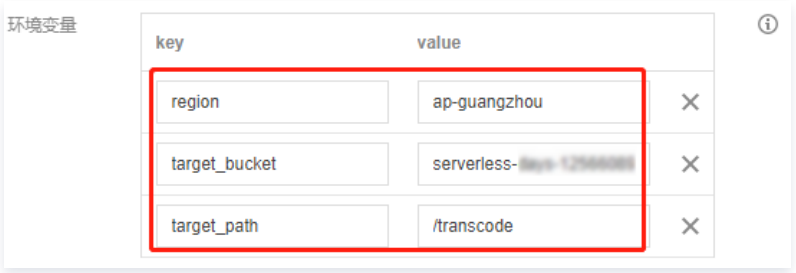
1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
3. 在“新建函数”页面根据以下信息选择函数模板。如下图所示：



- **创建方式**：选择**模板创建**。
- **模糊搜索**：输入“转码”，并进行搜索，本文以运行环境 Python3.6 为例。
  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，函数名称默认填充，可根据需要自行修改。按照引导配置环境变量和运行角色，如下图所示：**

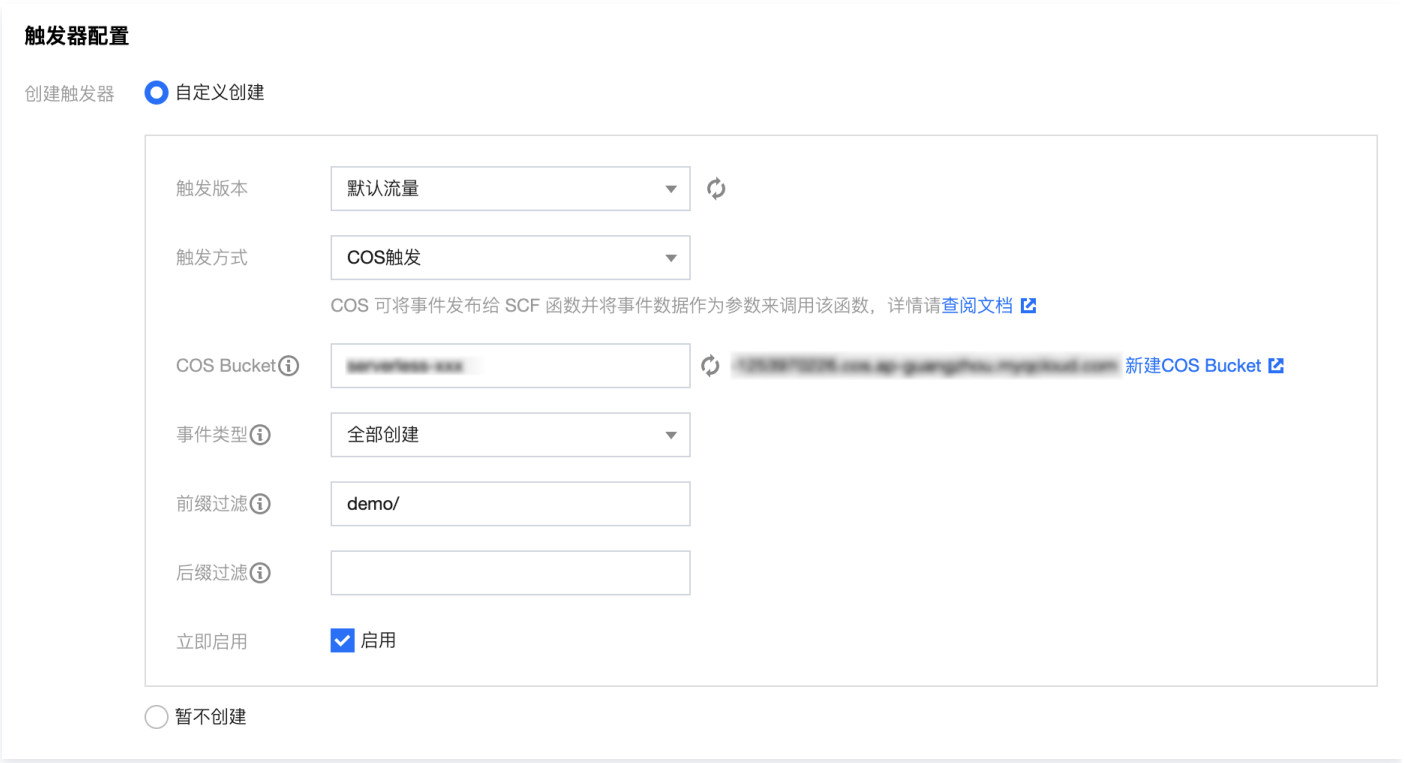
- **环境变量**：填写可参考下表。



key	value
region	COS Bucket 所在地域。
target_bucket	转码后的视频，上传到已创建好的 COS Bucket 中。
target_path	转码后的视频，上传到 Bucket 的指定目录中。

- **运行角色**：勾选“启用”，选择“配置并使用SCF模板运行角色”，将会自动创建并选择关联了 COS、CFS 全读写权限的 SCF 模板运行角色，或选择“使用已有角色”，在下拉列表中选择在 前提条件 中已创建的运行角色，本文以“配置并使用SCF模板运行角色”为例。
  云函数在运行时，会使用运行角色换取临时密钥，操作读取和写入 COS Bucket 的资源。

5. 在**触发器配置**中，参考以下信息创建 COS 触发器。如下图所示：



主要参数信息如下，其余选项请保持默认设置：

- **触发方式**：选择“COS触发”。
- **COS Bucket**：选择存储桶。若用的同一个 Bucket 存储源视频和转码后的视频，则需在触发器中配置前缀过滤规则/。例如 demo/。

6. 单击**完成**即可完成函数和触发器创建。

（可选）配置 CFS 挂载

> **说明**
> 若您开通了 CFS 挂载服务，则请参考以下步骤在云函数中同时启用私有网络和文件系统挂载能力。

1. 在的云函数“函数配置”页面，单击右上角**编辑**，参考以下信息进行配置。如下图所示：

- **私有网络**：勾选“启动”，并选择私有网络及子网。
- **文件系统**：勾选“启用”，选择选择在 前提条件 中已创建的文件系统。

2. 单击**函数代码**页签，进入函数代码编辑页面，修改文件上传路径。如下图所示：

```
75      logger.info('key: '+ key)
76      # upload_path = '/tmp/new-'+ key.split('/')[-1]
77      upload_path = '/mnt/new-'+ key.split('/')[-1]
78      logger.info('upload_path: '+ upload_path)
```

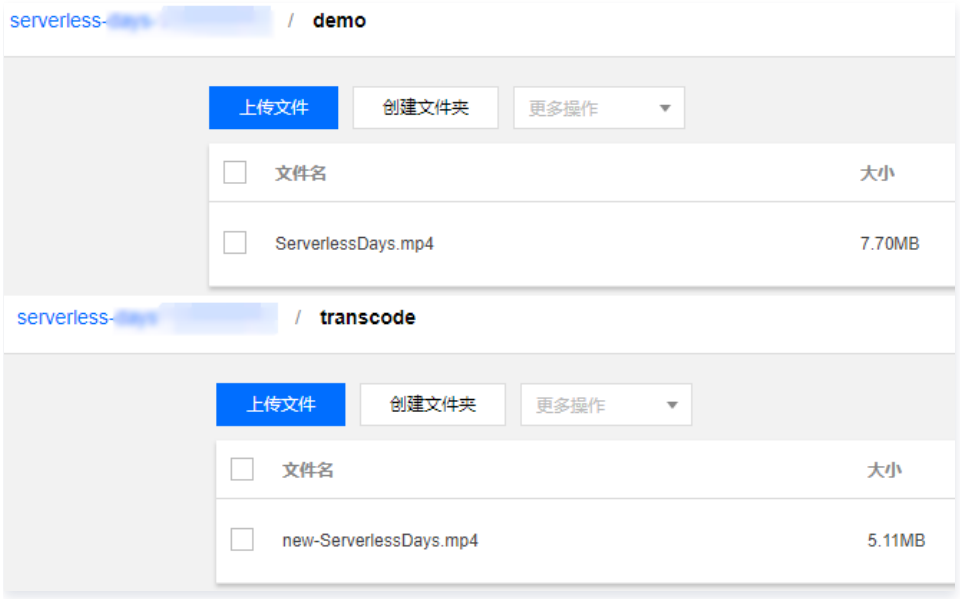
注释第76行代码，并在77行添加以下代码。

```
upload_path = '/mnt/new-'+ key.split('/')[-1]
```

测试功能

登录 对象存储控制台 ，进入对应的 Bucket 目录下，上传视频文件，并到对应的转码目录下查看，是否生成压缩的视频文件。如下图所示：

> **说明**
> 根据视频大小不同，压缩时间也不同。如果视频过大，则压缩时间也会延长，需要较长的时间才能查看到新视频。



扩展能力

基于本文示例，您还可扩展自动化 CDN 刷新/预热的能力。例如，转码后的视频在回传 COS Bucket 时，可触发新函数执行 CDN 刷新/预热功能。详情请参见 CDN 缓存刷新 。
您还可借助云函数的高并发能力，实现快速转码或者切片功能。例如，函数 A 进行调度任务，函数 B 进行实际的转码/切片工作。可借助 CFS 挂载能力，轻松实现跨函数的文件共享功能。

SCF + COS 实现日志分析写数据库

最近更新时间：2023-07-31 15:00:33

操作场景

在本文档示例中，我们用到了云函数 SCF、对象存储 COS、云数据库 MySQL。其中，COS 用来存储需要分析的日志文件，SCF 实现从 COS 下载日志文件并进行统计分析，把分析的结果写入到 MySQL 数据库中。

操作步骤

创建 COS Bucket

1. 登录 对象存储控制台，选择左侧导航栏中的**存储桶列表**。
2. 参考 创建存储桶 创建一个存储桶，主要参数信息设置如下：
   - **名称**：命名为 loganalysis。
   - **所属地域**：本案例地域选择北京。使用外网连接时 Bucket 可选与 MySQL 云数据库不同的地域。
   - **访问权限**：选择“私有读写”。

创建 MySQL 云数据库

1. 由于数据库需要付费购买，您可以选择在目标地域购买云数据库 MySQL 入门机型，本案例地域选择北京。
2. 参考 云数据库MySQL入门概述 创建一个 MySQL 云数据库。
3. 购买完成后，给数据库添加可访问的用户名和密码，并创建新实例，本案例实例名称使用 `mason_demo` 。

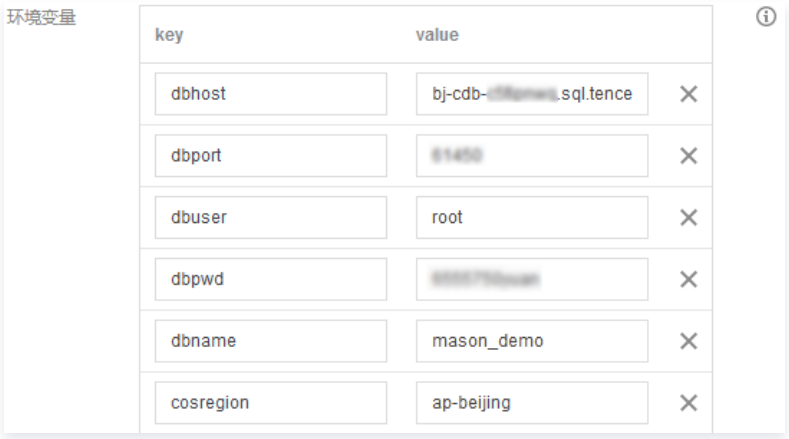
创建云函数 SCF

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面。
   设置以下参数信息，并单击**下一步**。如下图所示：
   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入“日志分析写数据库”，并进行搜索。
     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。

3. 函数名称默认填充，可根据需要自行修改。按照引导配置环境变量、运行角色和私有网络：

环境变量

在使用本模板函数时，您需要按照提示在函数配置中添加环境变量，填写方式可参考下图：



key	value
dbhost	请参考 访问 MySQL 数据库 获取。本文以外网为例，格式为 bj-cdb-xxxxx.sql.tencentcdb.com:00000。其中冒号后数字为 dbport。
dbport	
dbuser	已创建的 MySQL 数据库的用户名。
dbpwd	已设置的 MySQL 账号密码。
dbname	需进行写入的数据库实例名称，本文以 mason_demo 为例。
cosregion	Bucket 所在地域的简称。详情可参见 Bucket 地域和访问域名 。

运行角色

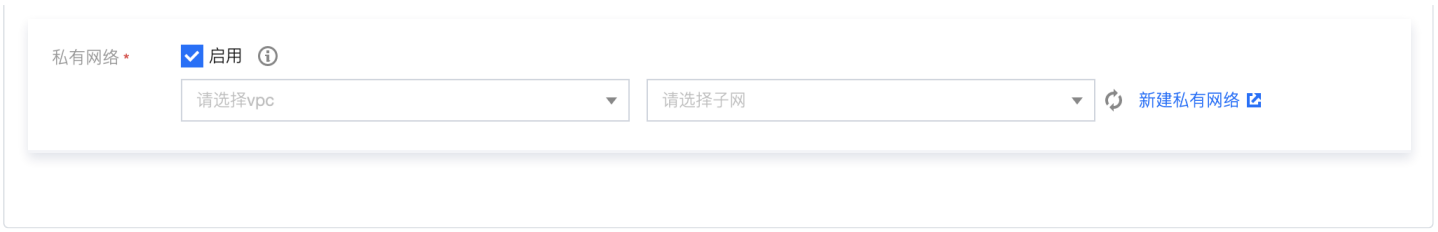
勾选“启用”，选择“配置并使用SCF模板运行角色”，将会自动创建并选择关联了 COS、CDB 全读写权限的 SCF 模板运行角色。或选择“使用已有角色”，在下拉列表中选择包含上述权限的已有角色。本文以“配置并使用SCF模板运行角色”为例。如下图所示：

注意

您也可以直接在函数代码中替换为账户实际使用的 SecretId 及 SecretKey，可前往 API密钥管理 页面获取。

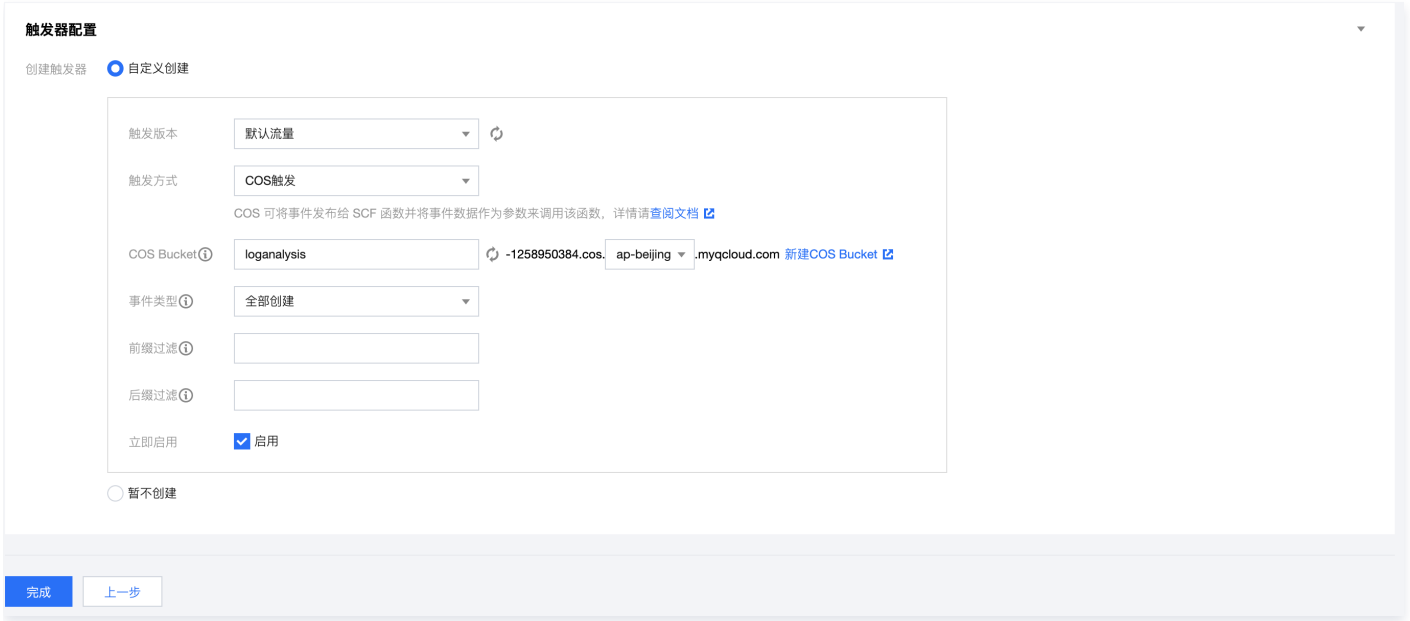
私有网络

如果数据库使用的是内网地址，则函数需要启用**私有网络**，并选择和数据库相同的 VPC 和子网。如下图所示：



配置 COS 触发器

在“触发器配置”中，选择“自定义创建”，并填写相关参数信息。如下图所示：



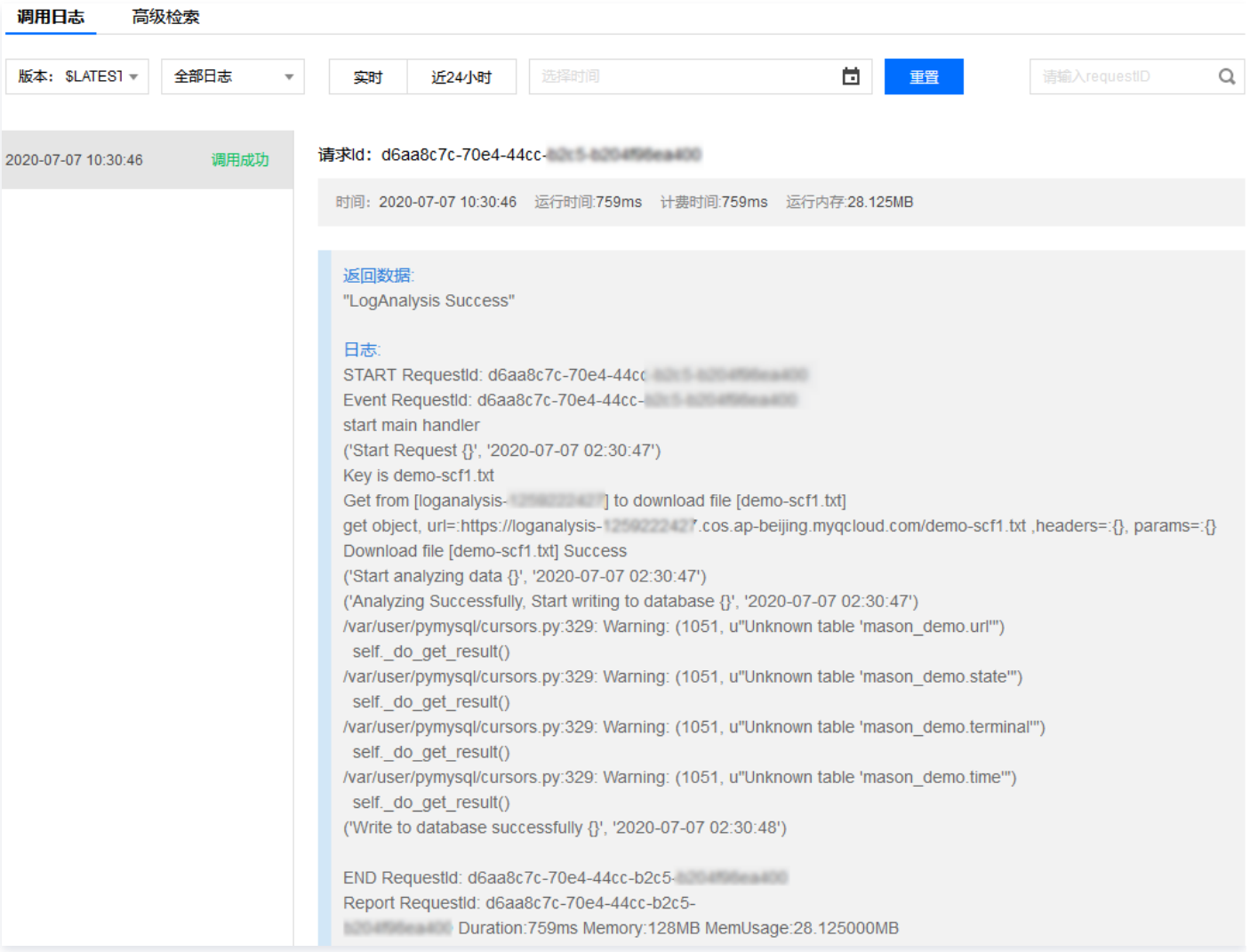
主要参数信息如下，其余配置项请保持默认：

- **触发方式**：选择 “COS触发”。
- **COS Bucket**：选择 创建 COS Bucket 步骤中已创建的存储桶 loganalysis。
- **事件类型**：选择“全部创建。”

单击**完成**，即可完成函数和触发器创建。

测试函数功能

1. 下载 测试样例 中的日志文件，并解压出 demo-scf1.txt。
2. 切换至 对象存储控制台，选择创建好的存储桶 loganalysis，单击**上传文件**。
3. 在弹出的“上传文件”窗口中，选择下载好的 demo-scf1.txt，单击**确定上传**。
4. 切换至 Serverless 控制台，查看执行结果。
   在函数详情页面中选择**日志查询**页签，可以看到打印出的日志信息。如下图所示：



5. 切换至 MySQL 管理界面，查看数据库中的分析结果。

> **说明**
> 您可以根据自身的日志格式编写具体的处理方法，数据库的写方法也可以修改为增量写。

SCF + COS 获取图像并创建缩略图示例说明

最近更新时间：2022-05-16 17:59:55

实现场景

> **注意**
>
> 1. 必须使用两个 COS Bucket。如果使用同一个存储桶作为源和目标，上传到源存储桶的每个缩略图都会触发另一个对象并创建事件，该事件将再次调用函数，从而产生无限的循环。
> 2. 请保证函数和 COS Bucket 位于同一个地域下。

本教程假设以下情况：

- 您的用户将上传照片至某个特定的 COS Bucket。
- 您要为用户上传的每个图像创建一个缩略图。
- 创建完缩略图后保存至另一个 COS Bucket。

实现概要

下面是该函数的实现流程：

- 创建函数与 COS Bucket 的事件源映射。
- 用户将对象上传到 COS 中的源存储桶（对象创建事件）。
- COS Bucket 检测到对象创建事件。
- COS 调用函数并将事件数据作为参数传递给函数，由此将 cos:ObjectCreated:* 事件发布给函数。
- SCF 平台接收到调用请求，执行函数。
- 函数通过收到的事件数据获得了 Bucket 名称和文件名称，从该源 Bucket 中获取该文件，使用图形库创建缩略图，然后将其保存到目标 Bucket 上。

请注意，完成本教程后，您的账户中将具有以下资源：

- 一个创建缩略图的 SCF 函数。
- 两个 COS Bucket，分别是源 Bucket 用于上传原始图片，目标 Bucket 用于存储裁剪后的图片。

步骤 1. 准备 COS Bucket

最近更新时间：2023-02-01 10:14:34

> **注意**
> 1. 源 Bucket、目标 Bucket 和函数必须位于同一个地域下。在本教程中，我们将使用广州区域。
> 2. 必须使用两个 COS Bucket。如果使用同一个 Bucket 作为源和目标，上传到源存储桶的每个缩略图都会再次触发函数，从而产生不必要的递归。

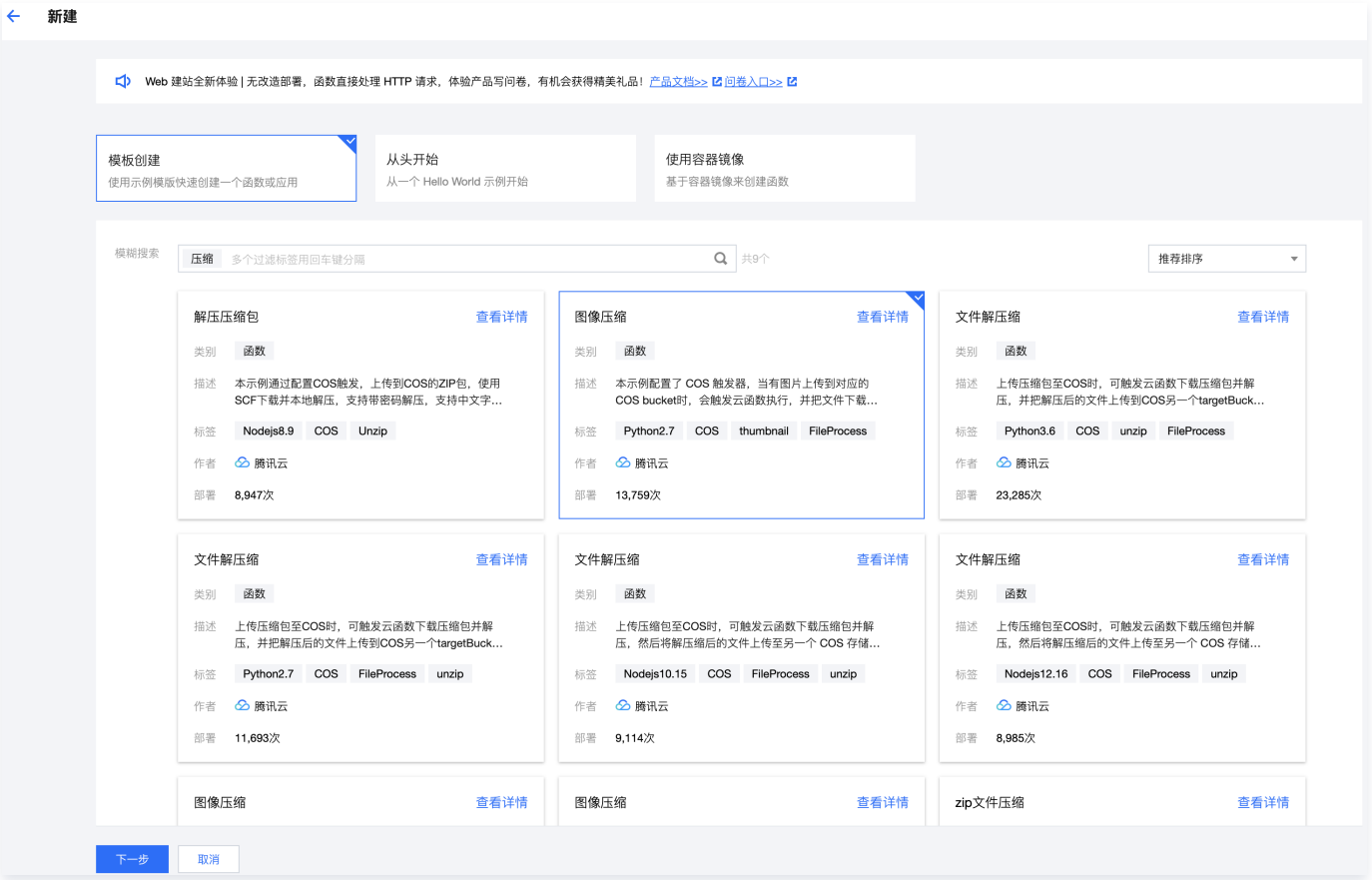
1. 登录 对象存储控制台 。
2. 单击**存储桶列表**选项卡下的**创建存储桶**按钮，参考 创建存储桶 新建源 COS Bucket。
   主要参数信息设置如下：
   - **名称**：命名为 `mybucket1` 。
   - **所属地域**：`广州` 。
   - **访问权限**：选择 `私有读写` 。
3. 按照相同的方式创建目标 Bucket `mybucket-resized1` 。

步骤 2. 创建 Thumbnail 函数并测试

最近更新时间：2024-11-07 10:23:52

创建 Thumbnail 函数

1. 登录 Serverless 控制台，选择左侧导航栏中的**函数服务**。
2. 在“函数服务”页面上方选择**广州**地域，并单击**新建**进入新建函数页面，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**模板创建**。
- **模板搜索**：输入“压缩”进行搜索，本文以运行环境 Python 2.7 为例。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。

3. 单击**下一步**，函数名称默认填充，可根据需要自行修改。按照引导配置运行角色：

- **运行角色**：勾选**启用**，本文以“配置并使用SCF模板运行角色”为例。详细说明如下：
- **配置并使用SCF模板运行角色**：选择该项将会自动创建并选择关联了 COS 全读写权限的 SCF 模板运行角色。
- **使用已有角色**：需在下拉列表中选择包含上述权限的已有角色。

> **说明：**
> 云函数在运行时，会使用运行角色换取临时密钥，操作相关云产品资源。

4. 在使用此模板函数时，您需要按照提示修改函数代码中的配置信息。
单击展开**函数代码**卡片，将函数代码中的 appid、secret_id、secret_key、region 和 resized_bucket 替换为您的 APPID、SecretId、SecretKey、region、resized_bucket。

> **说明：**
> - APPID 可在控制台 账户信息 中获得。
> - SecretId 和 SecretKey 可在控制台 API密钥管理 中获得。

配置 COS 触发器

1. 在**触发器配置**中，选择**自定义创建**，根据页面相关信息提示进行配置。如下图所示：

主要参数信息说明如下：

- **触发方式**：选择**COS 触发**。
- **COS Bucket**：选择 步骤1 中已创建的存储桶 mybucket。
- **事件类型**：选择**全部创建**。

2. 单击**完成**，完成函数和COS触发器创建。

测试函数功能

1. 切换至 对象存储控制台，选择已创建的存储桶 Bucket：`mybucket1`，单击**上传文件**，选择任意一张 .jpg 或 .png 的图片，并进行上传。
2. 切换至另外一个存储桶 Bucket：`mybucket-resized1`，查是否有同名的文件生成，并下载对比两张图片的大小。
3. 进入云函数控制台查看执行结果，在**运行日志**中可以看到打印出来的日志信息。

SCF + COS 实现身份证文字识别

最近更新时间：2023-12-08 14:14:42

创建 COS Bucket

1. 登录 对象存储控制台 。
2. 创建一个 Bucket，命名为 idcard-detect，并选择**北京**地域，权限选择 “私有读写” 。

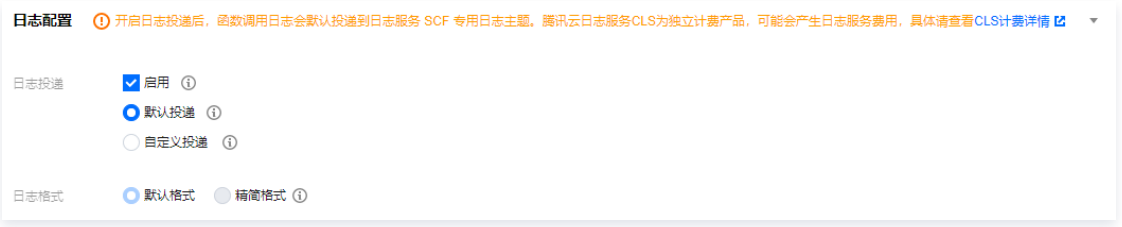
开通 AI 接口

前往 文字识别控制台 开通身份证识别功能，单击**开通服务**即可。

创建云函数

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在**函数服务**页面上方选择**北京**地域，并单击**新建**。
3. 在**新建函数**页面，设置以下参数信息，并单击**下一步**。如下图所示：
   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入 “身份证识别” ，并进行搜索。
     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. 在**基础配置**中，函数名称默认填充，可根据需要自行修改。
5. 在**日志配置**中，启用日志投递。如下图所示：



6. 在**高级配置** > **权限配置**中，勾选 “启用” ，在下拉列表中选择运行角色。如下图所示：

说明

> 云函数在运行时，会使用运行角色换取临时密钥，操作相关云产品资源。

7. 在**触发器配置**中，选择“自定义触发器”，根据页面的参数信息进行填写。
   - **触发方式**：选择 “COS 触发”。
   - **COS Bucket**：选择 “idcard-detect”。
   - **事件类型**：选择“全部创建”。
8. 单击**完成**，即可完成函数和 COS 触发器创建。

测试函数功能

1. 切换至 对象存储控制台，选择创建好的 Bucket：idcard-detect，单击**上传文件**，选择自己拍好的身份证照片，照片要清晰可读且尽可能大的占满图片，然后上传。
2. 进入云函数控制台查看执行结果，在**日志查询**中可以看到打印出来的日志信息，包含身份证识别的结果。

SCF + COS 数据入湖方案

最近更新时间：2022-04-18 10:48:49

操作场景

Serverless 架构的入湖方案是通过云函数触发器拉起数据调用后，通过云函数捕获并记录批次数据信息，在函数内闭环相关的结构转换和数据格式转换，数据压缩等能力。在本文档示例中，我们用到了云函数 SCF、对象存储 COS 对 TDMQ 消息进行备份。

操作步骤

创建 COS 储存桶

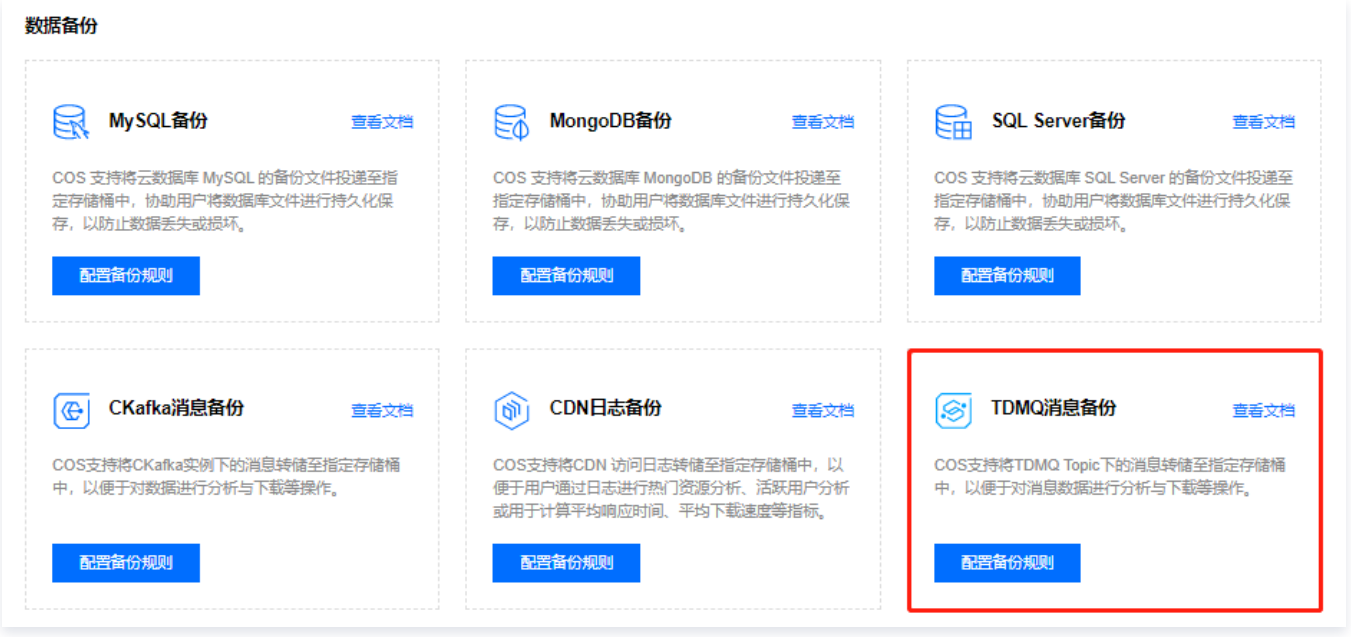
1. 登录 对象存储控制台 。
2. 选择左侧“存储桶列表”，单击**创建存储桶**，新建源 COS Bucket。
3. 设置 COS Bucket 的名称，例如 “ `scf-data-lake` ”，地域为“广州” ，设置访问权限为默认值“私有读写” ，单击**确定**完成创建。

创建消息队列 TDMQ

1. 通过 TDMQ 控制台创建集群和 Topic 等资源，详情可参见 TDMQ 资源创建与准备 。
2. TDMQ 集群需接入 VPC，详情可参见 VPC 接入 。

通过 COS 应用集成进行函数配置

1. 登录 对象存储控制台 。
2. 选择左侧“应用集成”，在“数据迁移与备份”中选择“TDMQ消息备份”。如下图所示：

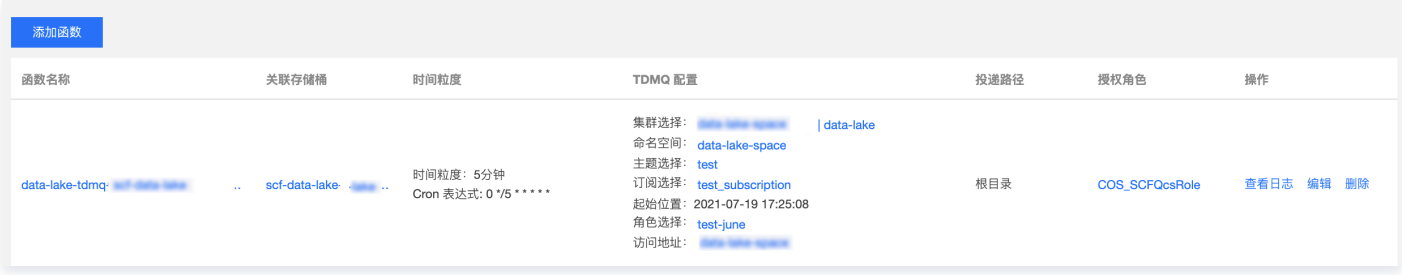


3. 在 “TDMQ消息备份”页中，选择已创建的 COS Bucket 所在的地域，并单击**添加函数**。
4. 在“创建TDMQ消息备份函数”弹窗中，进行函数基础配置。输入函数名称，例如 `data-lake` ，并关联已创建的 Bucket `scf-data-lake` ，勾选**授权SCF服务**后单击**下一步**。
5. 单击**下一步**，进行 TDMQ 配置，配置项说明如下：
   - **集群选择**：选择消息来源的 TDMQ 集群，仅支持同地域的 TDMQ 集群。
   - **命名空间**：选择集群中的命名空间。
   - **主题选择**：选择消息来源的主题。
   - **订阅选择**：选择对应的主题订阅，如现有的订阅不满足需求，可前往 TDMQ 控制台的 消费管理 新建订阅。
   - **起始位置**：历史消息的起始位置。
   - **角色选择**：选择 TDMQ 角色。TDMQ 的“角色”是 TDMQ 内专有的概念，区别于腾讯云的“角色”，是用户自行在 TDMQ 内部做权限划分的最小单位，用户可以添加多个角色并为其赋予不同命名空间下的生产和消费权限。

- **角色密钥**：选择 TDMQ 的角色密钥。TDMQ 的“密钥”是一种鉴权工具，用户可以通过在客户端中添加密钥来访问 TDMQ 进行消息的生产消费。密钥和角色一一对应，每种角色都有其对应的唯一密钥。
- **访问地址**：必须为 VPC 内网访问地址。

> **注意**
> 对应的 VPC 子网中必须有可用的 IP，且必须支持 DHCP。

6. 单击**下一步**，进行投递配置，配置项说明如下：
   - **投递的路径**：备份文件的投递路径前缀，不填写则默认保存在存储桶根路径，指定前缀必须以斜杠 `/` 为结尾。
7. 添加配置后，单击**确认**，即可看到函数已添加完成。如下图所示：



您可以对新创建的函数进行如下操作：

- 单击**查看日志**，查看 TDMQ 消息备份的历史运行情况。当备份出现报错时，您还可以通过单击**查看日志**，快速跳转到云函数控制台查看日志错误详情。
- 单击**编辑**，修改 TDMQ 消息备份规则。
- 单击**删除**，删除不使用的 TDMQ 消息备份规则。

测试云函数

1. 登录 TDMQ 控制台 。
2. 选择左侧“Topic管理”，选择地域、当前集群和命名空间。
3. 在“Topic管理”列表页，单击已关联的 Topic 主题右侧的**发送消息**。如下图所示：

4. 在弹出的对话框中输入消息内容。消息长度不超过64KB。
5. 单击**提交**，完成消息的发送。
6. 进入 COS 已关联桶详情页，例如 `scf-data-lake` 中已定义的**投递的路径**，查看是否有文件被创建。如有，则 TDMQ 消息备份成功。

SCF + COS 实现自定义计算文件哈希值

最近更新时间：2022-12-26 17:42:18

概述

数据在客户端和服务器间传输时可能会出现错误，对象存储（Cloud Object Storage，COS）结合云函数（Serverless Cloud Function，SCF）可以通过数据校验的方式保证上传数据的完整性，例如 MD5 码校验。用户在 COS 上传文件过程中，SCF 将帮助校验用户上传的对象，保证上传数据的完整性与正确性。

实践背景

业内现存公有云对象存储服务均不存在 MD5 码，用户上传文件后可能会出现以下情况：

- 文件重复、成本上升
- 文件错误、降低业务效率
- 文件缺失

方案优势

- 可视化操作：一键配置，简化开发流程，无需编码工作，大幅提升研发效率
- 多样化选择：支持 MD5 、SHA1 、SHA256、CRC64，满足各场景用户需求
- 自动化执行：文件上传到 COS 后，即可触发工作流开始计算校验码

操作步骤

1. 登录 对象存储控制台 。
2. 创建工作流，自定义格式过滤规则，及创建自定义函数节点。详细操作请参见 配置工作流 。

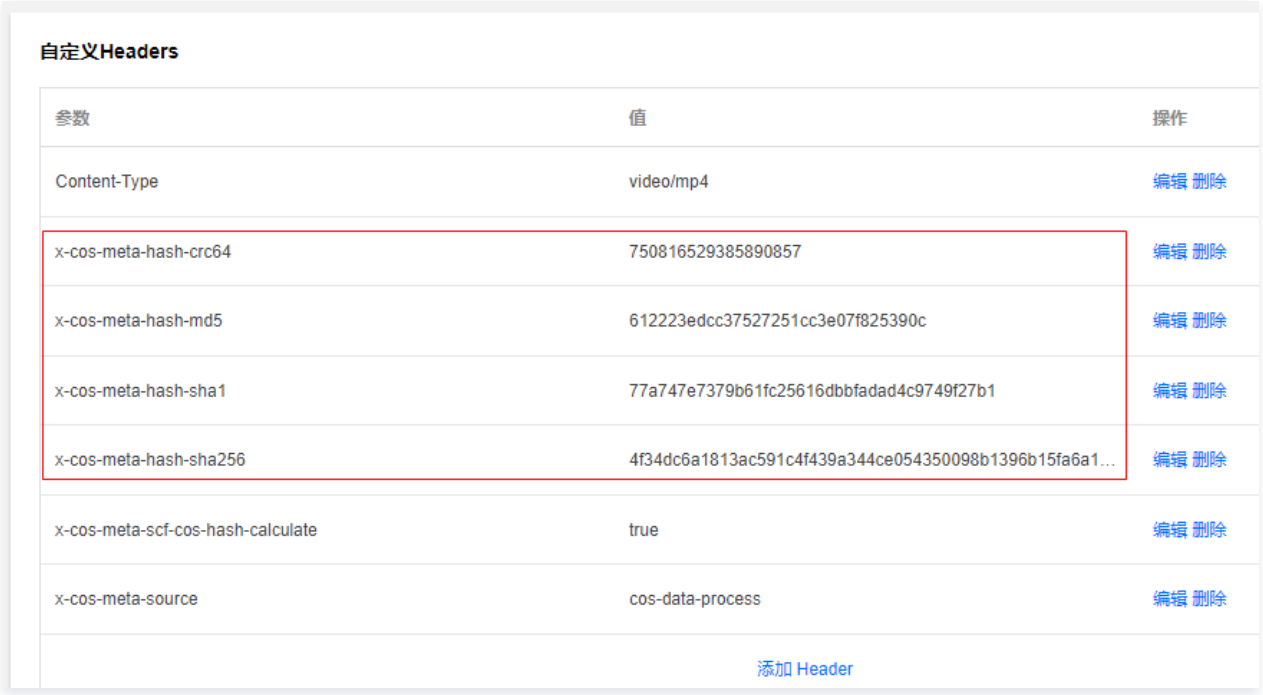
3. 在函数节点弹窗中，单击**新增函数**。

4. 在 SCF 的创建页面，选择**计算COS对象的哈希值**模板。

5. 根据用户文件大小，在基础配置项中配置执行超时时间，在高级配置中配置足够的内存。
6. 配置函数代码，该函数模板支持以下两个环境变量：
   - hashTypeList 指定要计算的算法，该项为可选，默认为["crc64", "md5", "sha1", "sha256"]。
   - caseType 指定哈希值大小写，该项为可选，默认为 lowercase，可以传入 uppercase。
7. 启用权限配置，绑定包含当前存储桶读写权限的角色，创建运行角色请参见 角色与策略 。
8. 单击**完成**。

9. 返回刚才的工作流页面，选中刚创建的自定义转码函数，并保存工作流。

10. 上传文件，待工作流处理成功后，即可看到上传的文件已成功添加多个哈希头部。

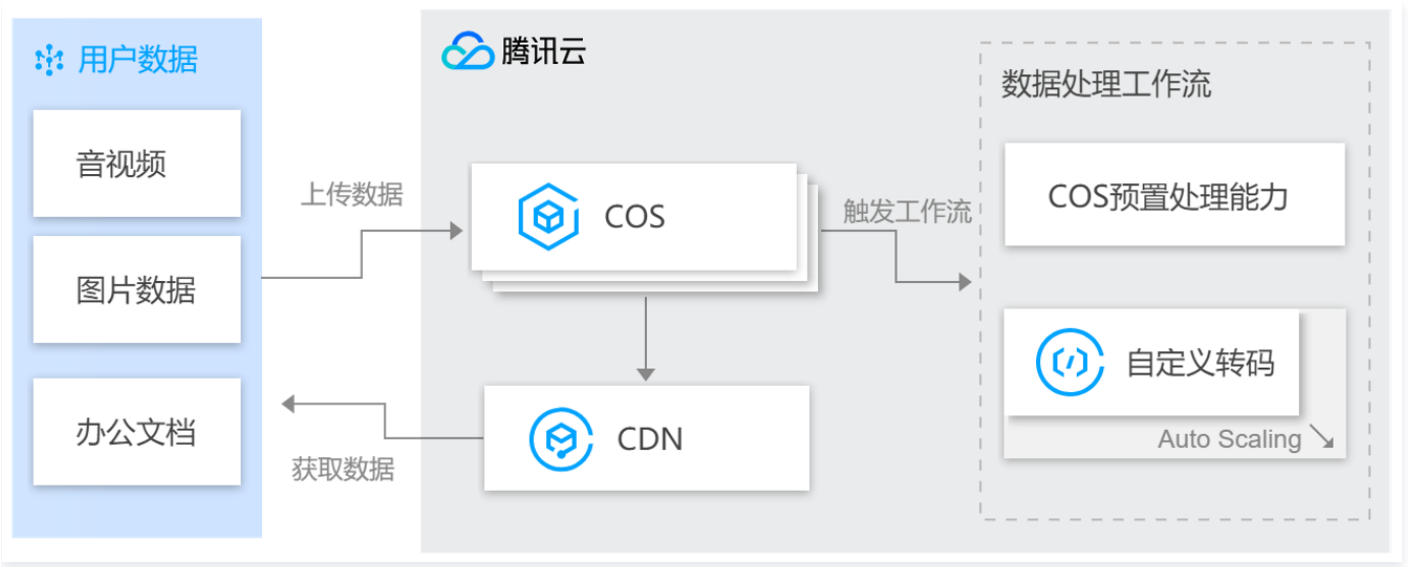


SCF + COS 实现自定义转码

最近更新时间：2022-12-26 17:44:42

概述

音视频作为信息传播中流量占比最大的部分在各行业的业务中都弥足重要，而不同的业务场景中对音视频的处理逻辑可能具备行业的特殊性。公有云虽然提供大量的视频处理服务供用户选择，但依然不能做到全面覆盖用户的特殊流程及定制化需求，使用对象存储（Cloud Object Storage，COS）工作流处理结合云函数（Serverless Cloud Function，SCF）定制逻辑此时就是一个绝佳选择，帮助用户快速创建满足需求的各种音视频处理服务。



应用场景

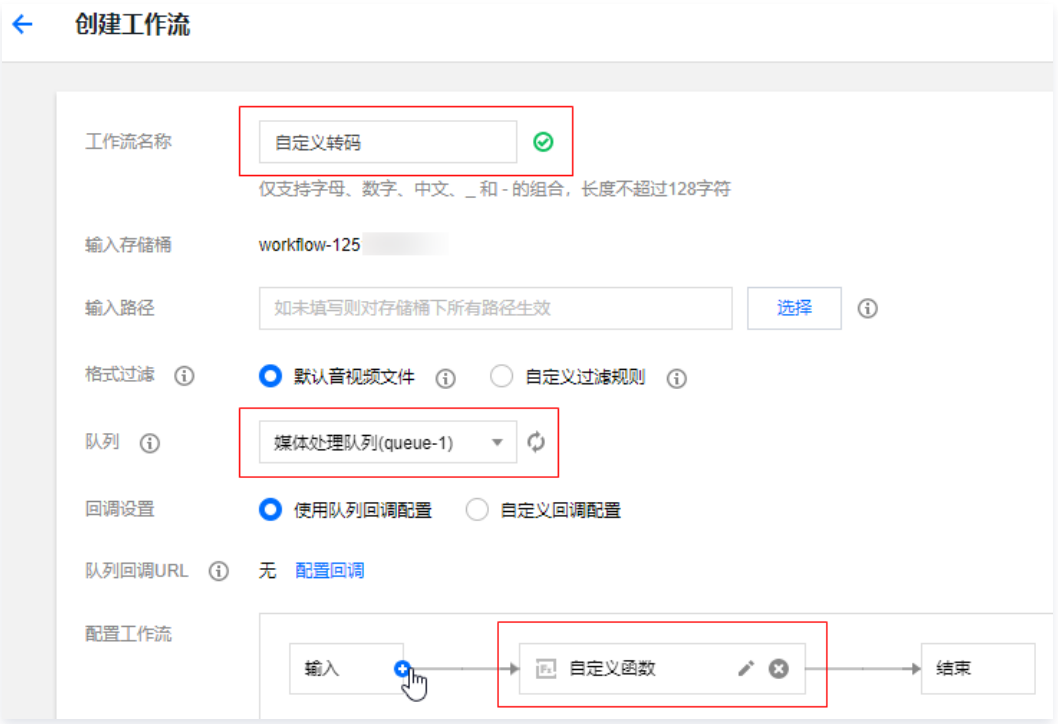
- 快速接入用户自建转码集群，兼容用户原有业务。
- 支持行业特殊格式与处理逻辑，接入电影、传媒等特殊行业。
- 支持用户自定义处理逻辑，满足各场景下定制流程需求。
- 触发工作流批量模板化处理，满足视频网站、教育、社交互联行业常见音视频处理需求。

方案优势

- 加速开发：不再需要关注资源运维与组件开销，极大地降低了服务架构搭建的复杂性。
- 降低开销：空闲时没有资源在运行，函数执行时按请求数和计算资源的运行时间收费，价格优势明显。
- 高可用、高扩展：根据请求自动平行调整服务资源，拥有近乎无限的扩容能力，且免除单可用区运行的故障风险。

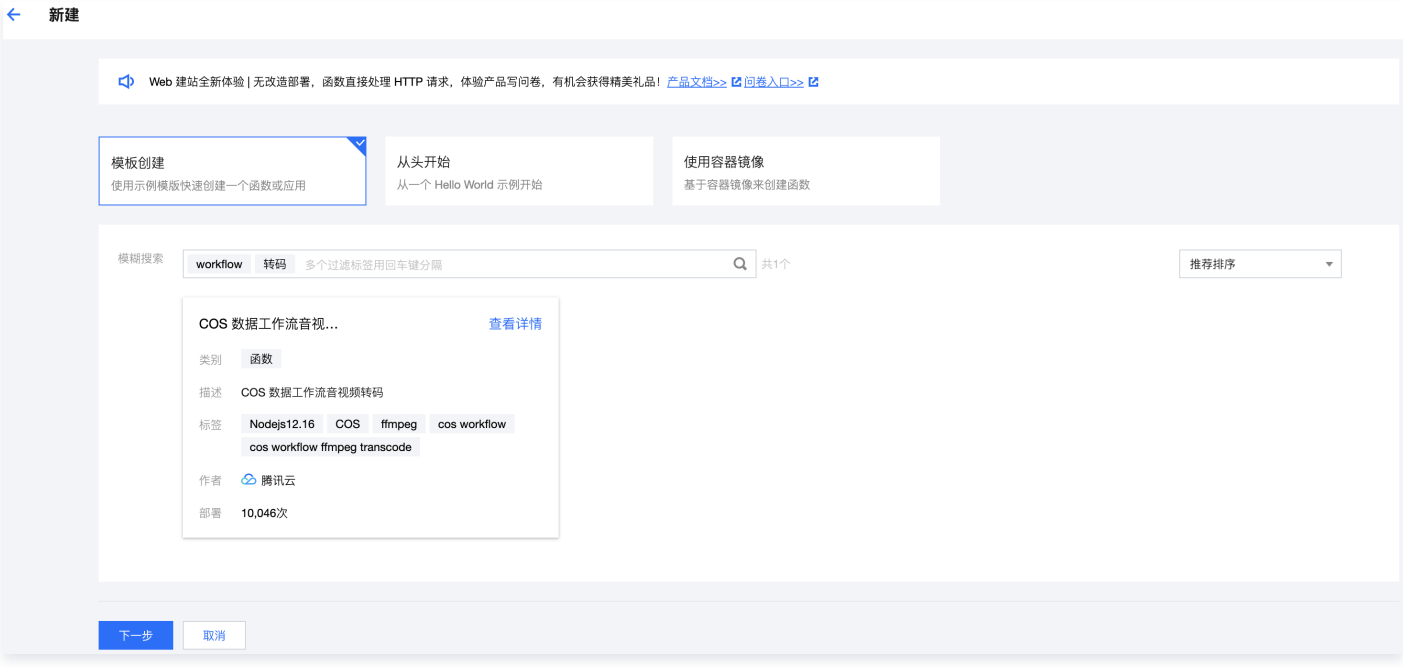
操作步骤

1. 登录 对象存储控制台，创建工作流、自定义过滤规则以及创建自定义函数节点。详细操作请参见 配置工作流 。

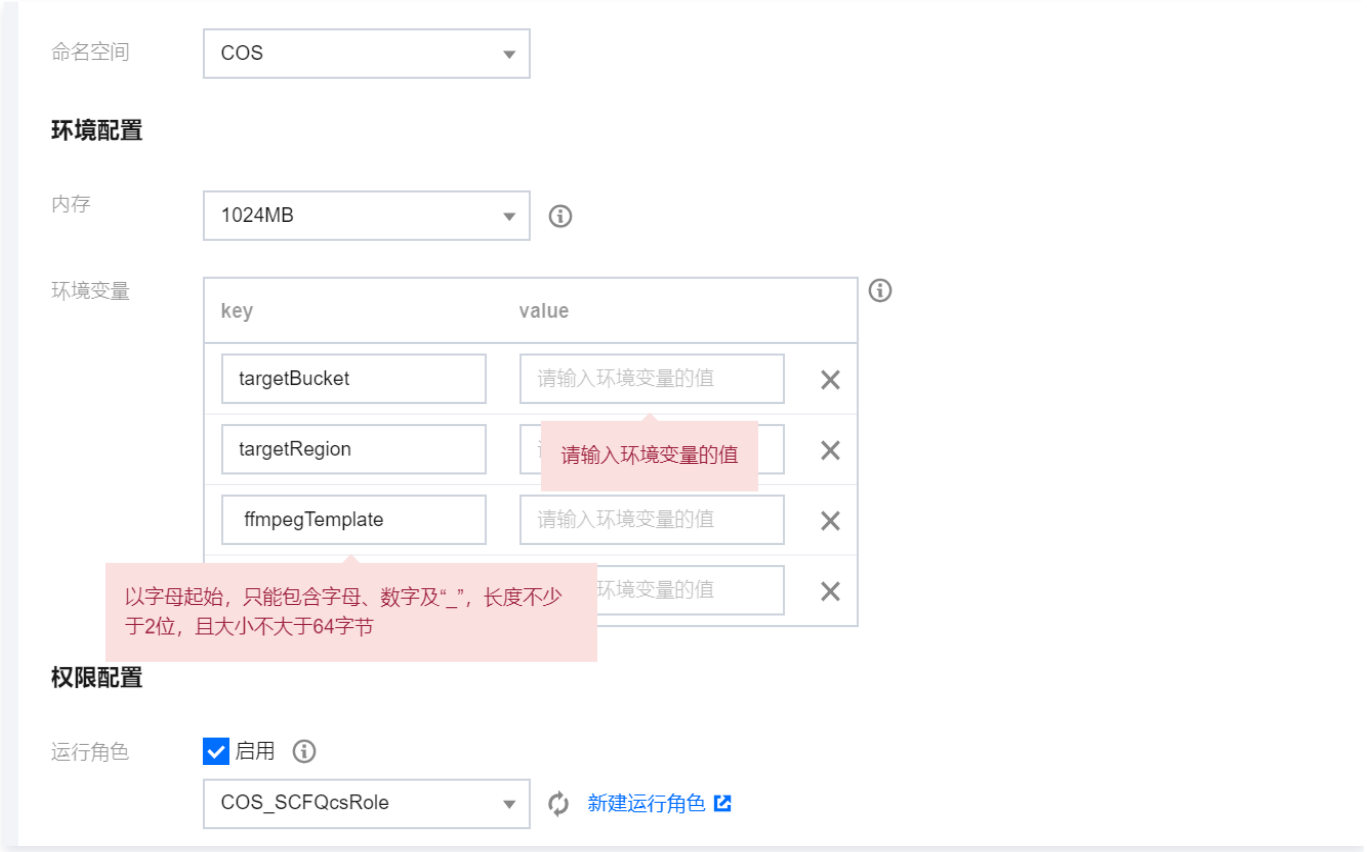


2. 在函数节点弹窗中，单击**新增函数**。

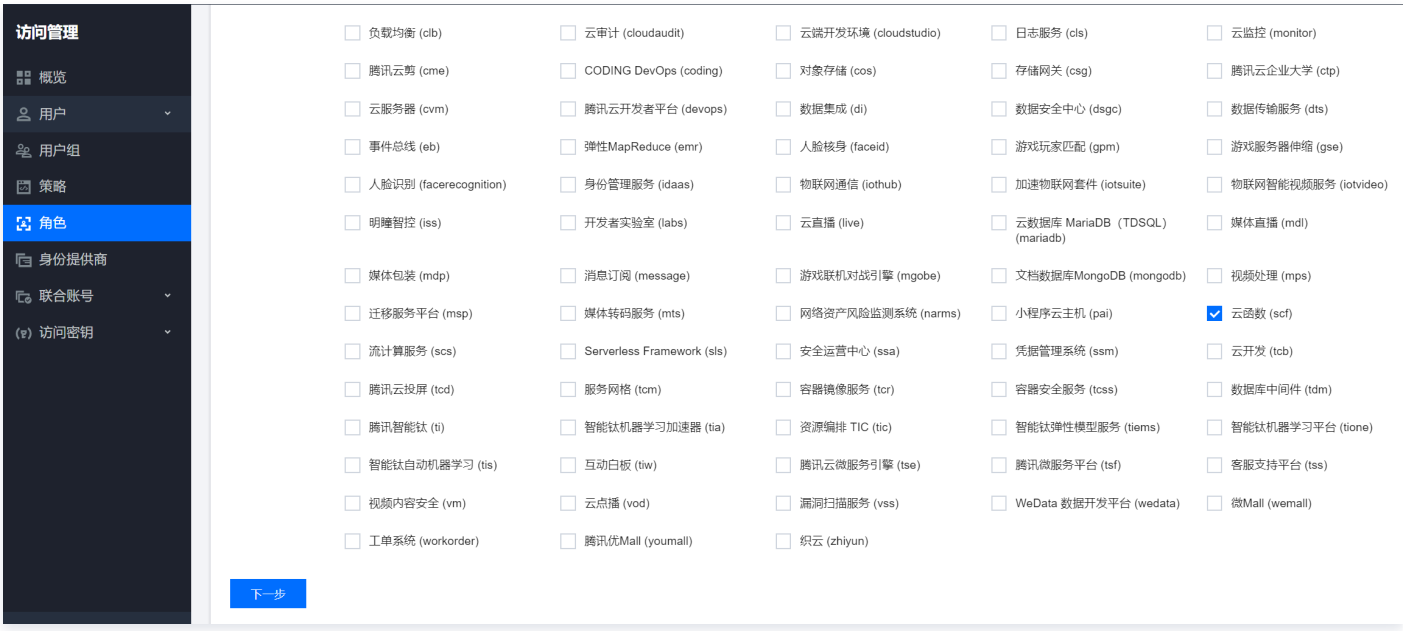
3. 在云函数的创建页面，选择**COS 数据工作流音视频转码**模板。



4. 根据用户文件大小，在基础配置项中配置执行超时时间，在高级配置中配置足够的内存。
5. 在**高级配置 > 环境配置**中配置环境变量，该函数模板支持以下环境变量：
   - targetBucket：目标存储桶，必填。
   - targetRegion：目标存储桶地域，必填。
   - targetKeyTemplate：目标路径模板，可选，默认为 `${InputPath}${InputName}_transcode.${ext}` 。
   - ffmpegTemplate：转码命令模板，必填，例如 `${ffmpeg} -loglevel error -i ${source} -r 10 -b:a 32k ${target}` 。
   - localTmpPath：临时保存路径，当绑定 CFS 时可以更改临时路径，可选，默认为 `/tmp` 。



6. 启用权限配置，绑定包含当前存储桶读写权限的角色，如需创建运行角色，请参见 角色与策略 。



7. 单击**完成**。

8. 返回刚才创建的工作流页面，选中刚创建的自定义转码函数，保存工作流，并在工作流列表页开启工作流。

9. 上传文件，待工作流处理成功后，即可看到上传的视频已成功转码并保存为新的文件。

SCF + COS 实现 MapReduce
示例说明

最近更新时间：2022-05-16 18:01:10

本教程假设以下情况：

- 您将不定时上传一些文本文件（例如日志等）至某个特定的 COS Bucket。
- 您要对这些文本文件进行字数统计。

实现概要

下面是该函数的实现流程：

- 创建函数与 COS Bucket。
- 用户将对象上传到 COS 中的源存储桶（对象创建事件）。
- COS Bucket 检测到对象创建事件。
- COS 调用函数并将事件数据作为参数传递给函数，由此将 cos:ObjectCreated:* 事件发布给函数。
- SCF 平台接收到调用请求，执行函数。
- 函数通过收到的事件数据获得了 Bucket 名称和文件名称，从该源 Bucket中获取该文件，根据代码中实现的 wordcount 进行字数统计，然后将其保存到目标 Bucket 上。

> ⚠ 注意
>
> - 本示例会使用三个 COS Bucket。如果使用同一个存储桶作为源和目标，上传到源存储桶的每个文件都会触发另一个对象创建事件，该事件将再次调用函数，从而产生无限的循环。
> - 请保证云函数和 COS Bucket 位于同一个地域下。

请注意，完成本教程后，您的账户中将具有以下资源：

- 两个 SCF 函数：Mapper 和 Reducer。
- 三个 COS Bucket：srcmr、middlestagebucket 和 destmr。
- Mapper 函数将会绑定 srcmr 触发，Reducer 函数将会绑定 middlestagebucket 触发，destmr 将会用来接收最终的统计结果。

步骤 1. 准备 COS Bucket

最近更新时间：2022-07-05 10:45:23

请确保您在执行此示例时，已经获得了 SCF 使用权限。

1. 登录腾讯云控制台，选择**云产品** > **存储** > **对象存储服务**。
2. 选择左侧导航栏中的 存储桶列表，进入“存储桶列表”页面。
3. 在“存储桶列表”页面，单击**创建存储桶**。
4. 在弹出的“创建存储桶”窗口中，参考以下信息新建源 COS Bucket。
   设置 COS Bucket 的名称例如 `srcmr`，选择地域为 中国 > 北京，设置访问权限为默认值 私有读写，单击**下一步**。
5. 其他可选配置保持默认值，单击**下一步**新建一个 COS Bucket。
6. 重复步骤1-4，创建中间阶段 Bucket `middlestagebucket` 和目标 Bucket `destmr`。

步骤 2. 创建 Mapper 和 Reducer 函数

最近更新时间：2022-07-05 10:45:31

创建 Mapper 函数

通过控制台模板函数创建

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面。根据页面相关信息进行配置。如下图所示：

- **创建方式**：选择**模板创建**。
- **模糊搜索**：输入“map_function”，并进行搜索，选择“map_function”模板。
  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。

3. 单击**下一步**，函数名称默认填充，可根据需要自行修改。按照引导配置运行角色：
   运行角色：勾选**启用**，本文以“配置并使用SCF模板运行角色”为例。如下图所示：

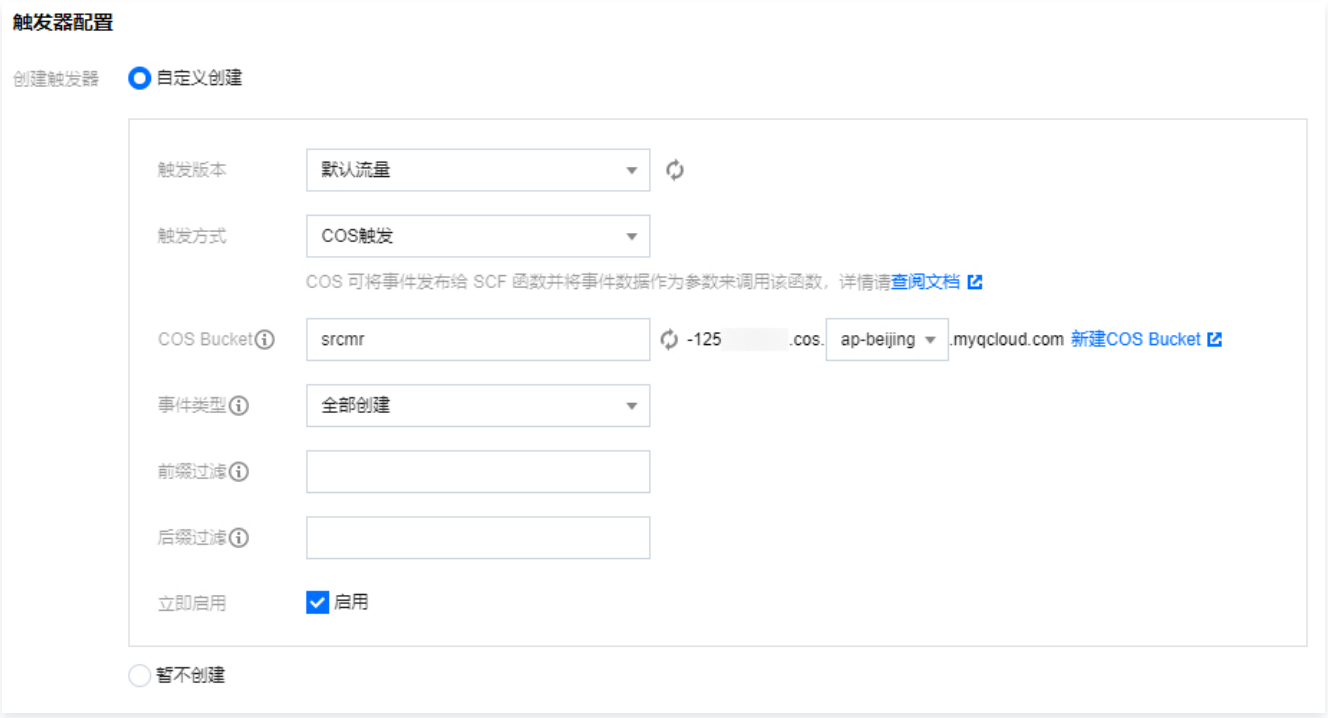
- **配置并使用SCF模板运行角色**：选择该项将会自动创建并选择关联了 COS 全读写权限的 SCF 模板运行角色。
- **使用已有角色**：需在下拉列表中选择包含上述权限的已有角色。

> **说明**
> 云函数在运行时，会使用运行角色换取临时密钥，操作相关云产品资源。

4. 单击展开**函数代码**卡片，可以浏览代码信息，如需修改变量等参数值，可以选择在线修改并保存。

配置 COS 触发器

1. 在**触发器配置**中，选择**自定义创建**，根据页面的参数信息进行填写。如下图所示：



主要参数信息说明如下：

- **触发方式**：选择**COS触发**。
- **COS Bucket**：选择**srcmr**。
- **事件类型**：选择**全部创建**。

2. 单击**完成**，完成函数和 COS 触发器创建。

创建 Reducer 函数

通过控制台模板函数创建

1. 登录 Serverless 控制台，进入函数服务页面。

2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面。根据页面相关信息进行配置。如下图所示：

- **创建方式**：选择**模板创建**。
- **模糊搜索**：输入 “reduce_function”，并进行搜索，选择 “reduce_function” 模板。
  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。

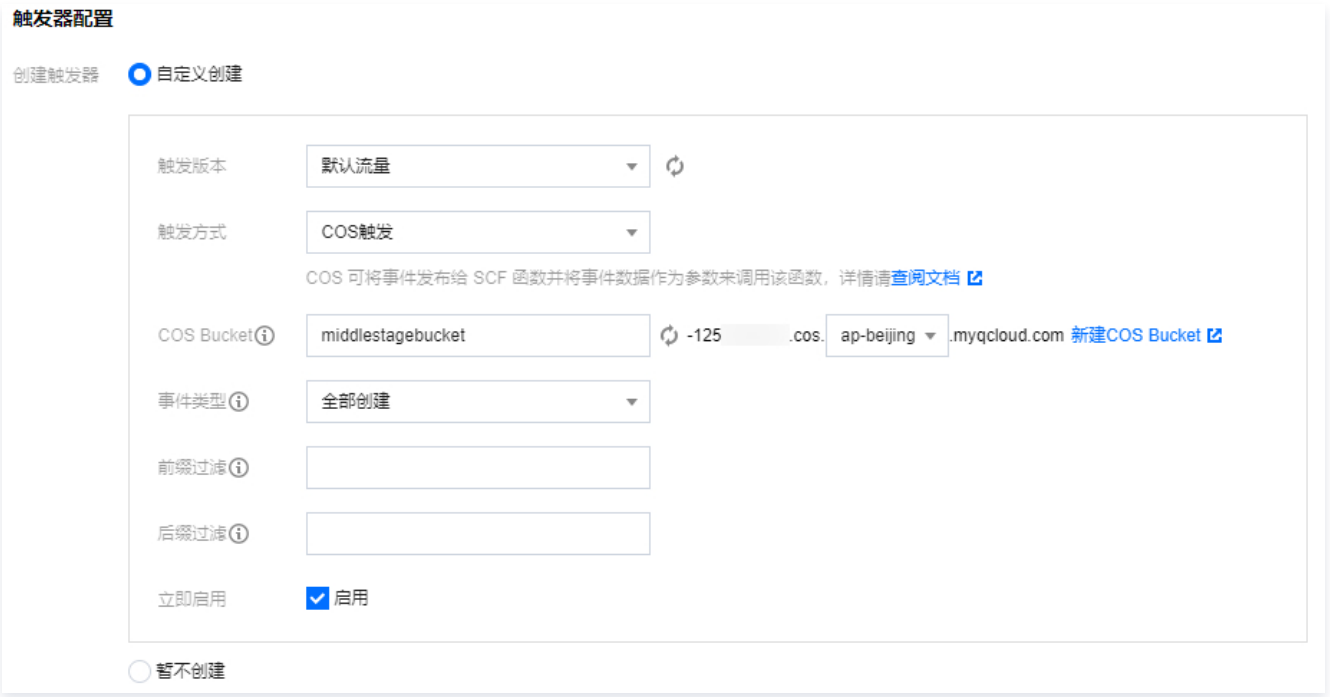
3. 单击**下一步**，函数名称默认填充，可根据需要自行修改。按照引导配置运行角色：
   运行角色：勾选**启用**，本文以“配置并使用SCF模板运行角色”为例。如下图所示：

- **配置并使用SCF模板运行角色**：选择该项将会自动创建并选择关联了 COS 全读写权限的 SCF 模板运行角色。
- **使用已有角色**：需在下拉列表中选择包含上述权限的已有角色。

4. 单击展开**函数代码**卡片，可以浏览代码信息，如需修改变量等参数值，可以选择在线修改并保存。

配置 COS 触发器

1. 在**触发器配置**中，选择**自定义创建**，根据页面的参数信息进行填写。如下图所示：



主要参数信息如下：

- **触发方式**：选择**COS触发**。
- **COS Bucket**：选择**middlestagebucket**。
- **事件类型**：选择**全部创建**。

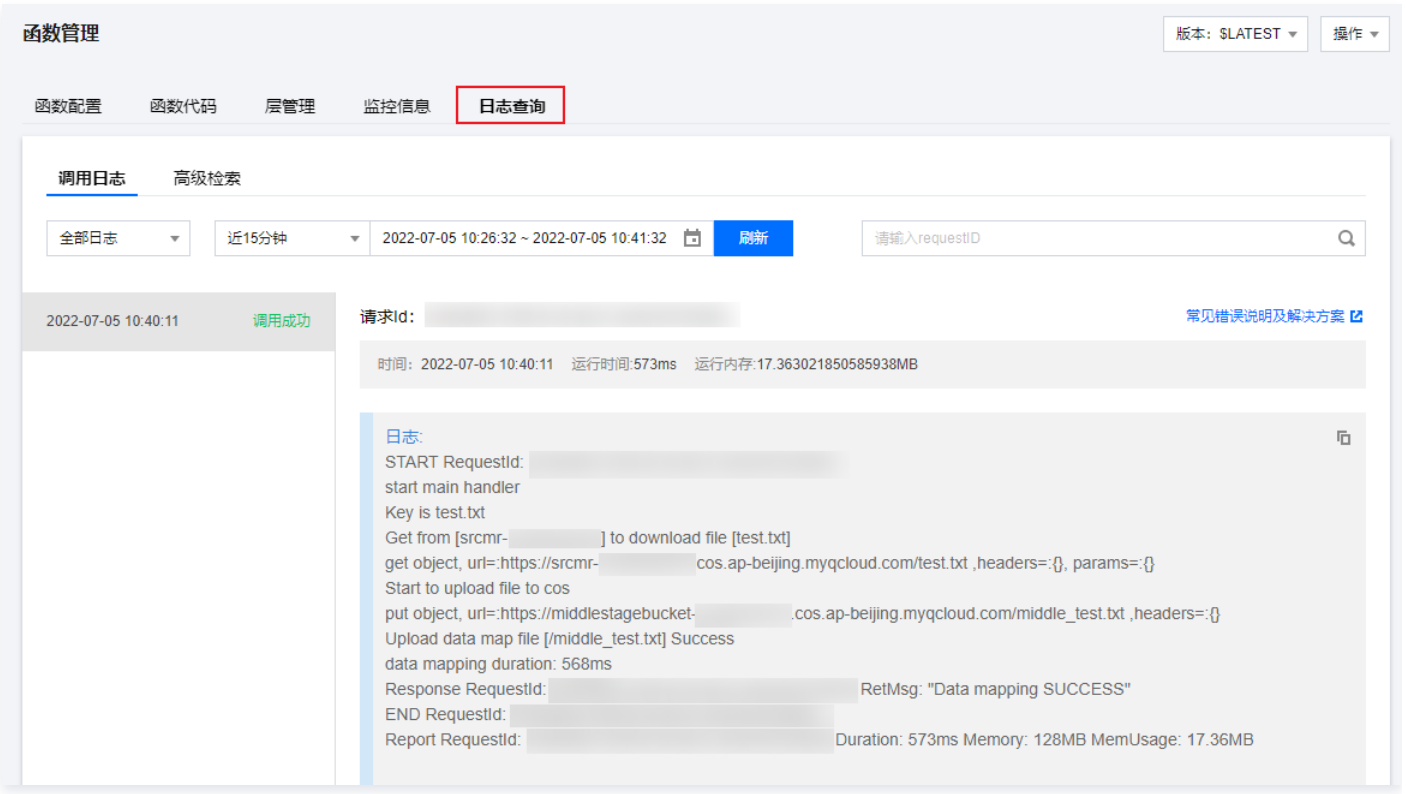
2. 单击**完成**，完成函数和 COS 触发器创建。

步骤 3. 测试函数

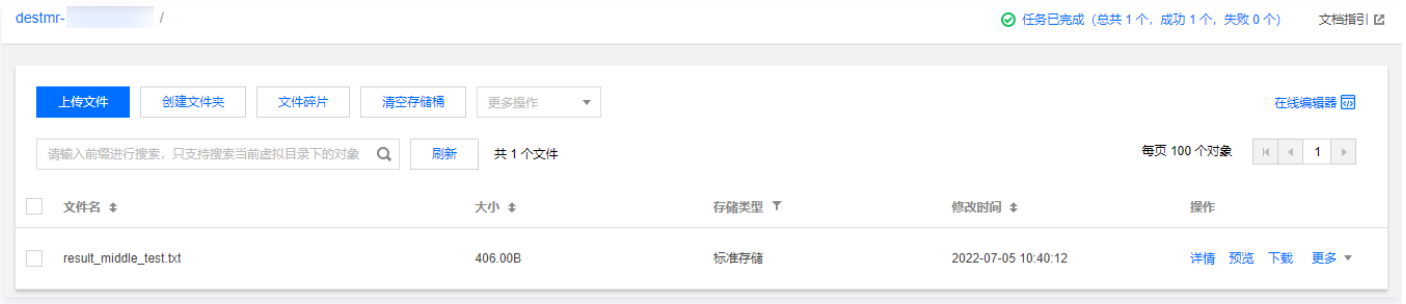
最近更新时间：2022-07-05 10:45:38

1. 下载 测试样例 中的文本文件，并解压出 test.txt。
2. 切换至 对象存储控制台，选择创建好的 Bucket：srcmr，单击**上传文件**。
3. 在弹出的 “上传文件” 窗口中，选择下载好的 test.txt，单击**上传**。如下图所示：

4. 切换至 Serverless 控制台，查看执行结果。在**日志查询**中可以看到打印出来的日志信息。如下图所示：



5. 切换至 对象存储控制台，选择创建好的 Bucket：destmr，查看生成的文件。如下图所示：



消息队列 CKafka
SCF + Ckafka 实现消息数据自定义处理并投递至 COS
示例说明

最近更新时间：2022-05-16 17:58:35

实现概要

本示例主要演示：

- 通过配置 Ckafka 触发器和 Ckafka 对接，实现消息的自定义逻辑处理。
- 将云函数处理后的消息以文本的形式投递到对象存储 COS。

实现步骤

1. 购买 Ckafka 实例，并创建 Topic。
2. 在云函数控制台中，部署业务处理函数，并配置 Ckafka 触发器，对接已有的 Ckafka Topic。
3. 创建 COS bucket，用于接收处理后的消息文本。

演示流程

在 Ckafka 控制台中的 Consumer Group 查看以 “scf_invoker” 开头的消费组。
当 Ckafka 产生消息后，云函数会被触发执行。同时，在 COS bucket 中也会产生基于时间维度的文件夹、文本文件。

函数部署

最近更新时间：2022-12-26 17:29:38

环境准备

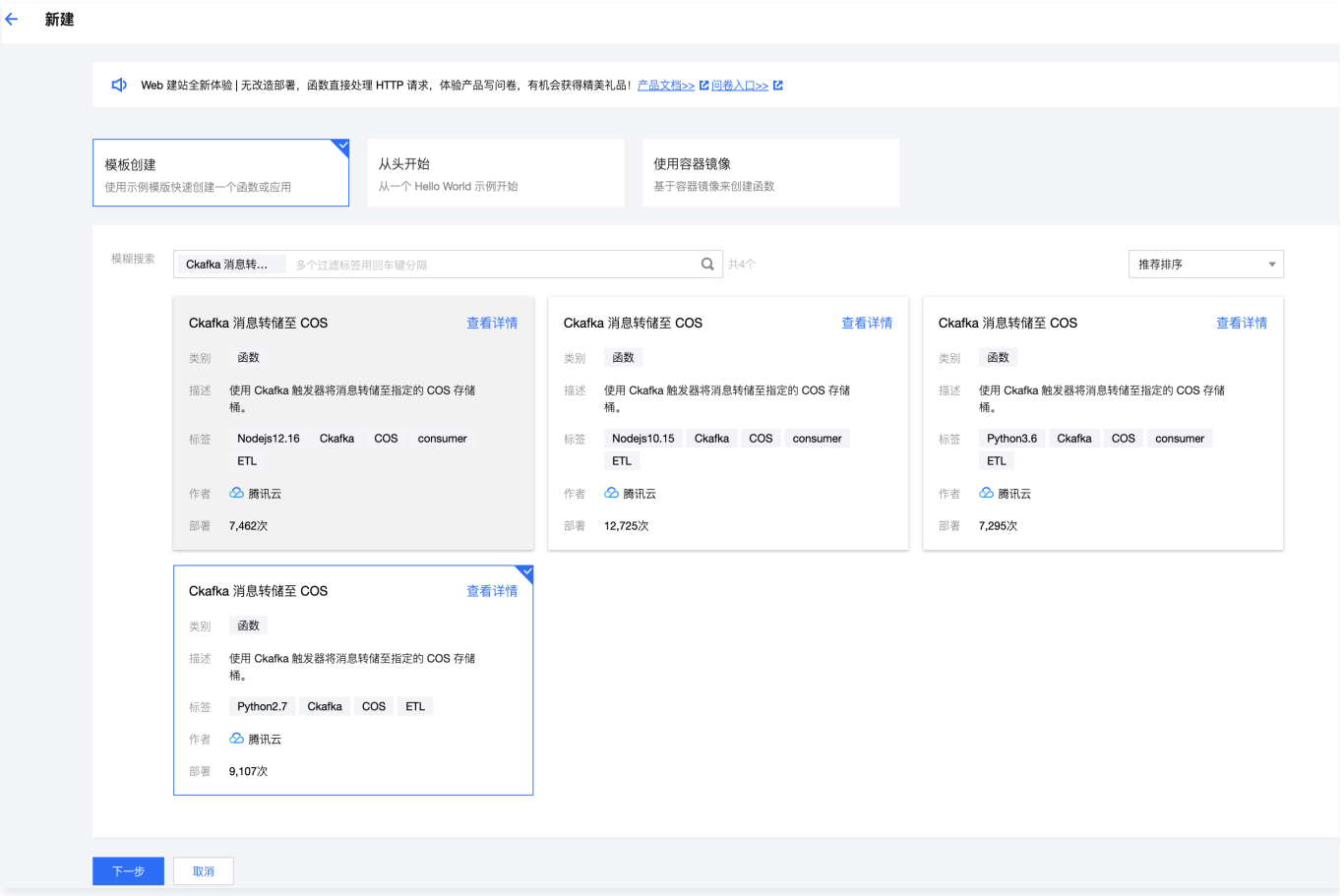
1. 登录 对象存储控制台，创建一个 Bucket，命名为 ckafka-backup，并选择**北京**地域，权限选择 “私有读写” 。该 bucket 用于存放云函数处理后的 Ckafka 消息。

> **注意**
> Bucket 和函数必须位于同一个地域下。本文将以**北京**区域为例。

2. 切换到 Ckafka 控制台，购买 Ckafka 实例，并创建 Topic。

创建云函数及 Ckafka 触发器

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面。
   设置以下参数信息，并单击**下一步**。如下图所示：
   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入 “Ckafka 消息转储至 COS”，并进行搜索，本文以运行环境 Python 2.7 为例。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。

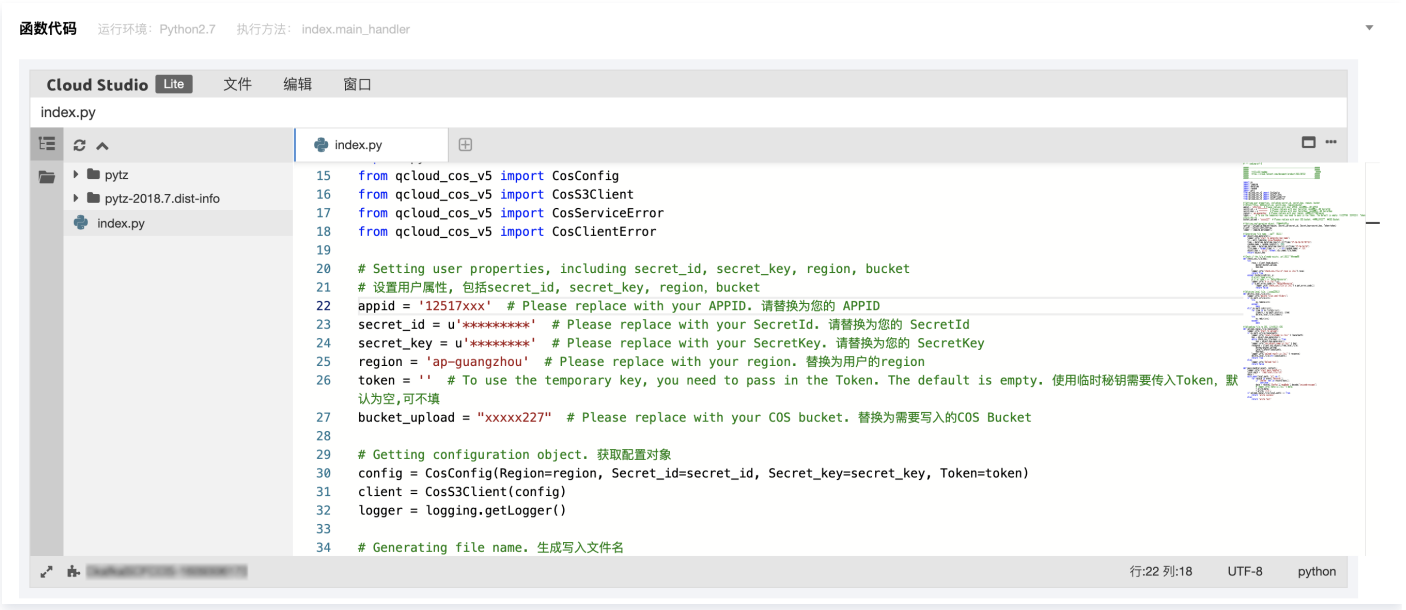


3. 单击**下一步**，函数名称默认填充，可根据需要自行修改。按照引导配置运行角色：
   - **运行角色**：勾选“启用”，选择“配置并使用SCF模板运行角色”，将会自动创建并选择关联了 COS、Ckafka 全读写权限的 SCF 模板运行角色，或选择“使用已有角色”，在下拉列表中选择包含上述权限的已有角色。本文以“配置并使用SCF模板运行角色”为例。如下图所示：

> **说明**

> 云函数在运行时，会使用运行角色换取临时密钥，操作相关云产品资源。

4. **在使用本模板函数时，您需要按照提示修改函数代码中的配置信息。**
   点击展开函数代码，将 appid、secret_id、secret_key、region 和 bucket_upload 替换为您的 APPID、SecretId、SecretKey、region、bucket_upload，并单击保存，完成创建。如下图所示：



5. 在**触发器配置**中，选择“自定义创建”，根据页面的参数信息进行填写。如下图所示：

主要参数信息如下：

- **触发方式**：选择 “Ckafka触发”。
- **Ckafka实例**：选择需要对接的 Ckafka 实例。
- **Topic**：选择需要对接的 Topic。
- **最大批量消息数**：请根据单条消息的大小和实际业务量进行配置，默认消息总数的大小要小于1MB。单次触发云函数运行时最大可处理的消息条数。例如，配置500，则最多会有500条消息会触发一次云函数。Ckafka 生产的消息过多时，消息组将自动触发多个函数实例运行，加速消息速度。

6. 单击**完成**，即可完成函数和触发器创建。

函数测试

最近更新时间：2022-04-18 10:50:05

> **注意**
> 如果您还未将实际数据接入消息队列 Ckafka，您可以通过 客户端工具 模拟消息生产。

检查 Consumer Group

登录 Ckafka 控制台，并在 “Consumer Group” 中检查消费组是否生成。 如下图所示：

- 是，表示部署成功。
- 否，表示部署失败，请 提交工单 反馈。

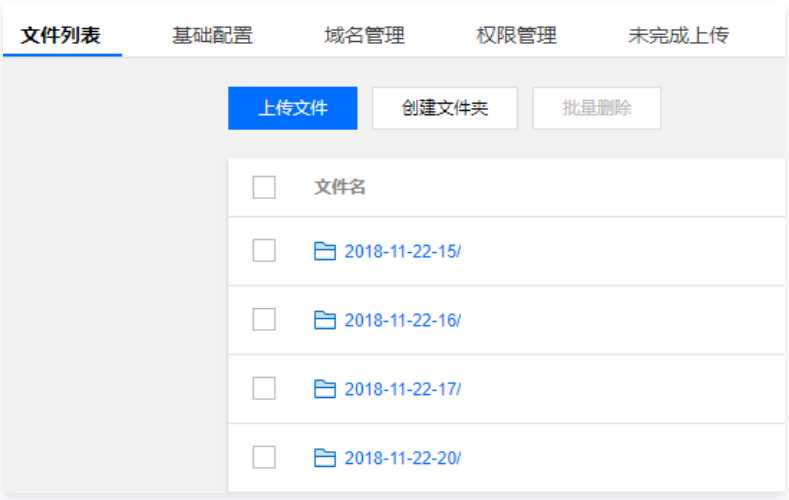
检查执行状态

触发器生效后，切换至 Serverless 控制台，查看函数的运行日志。如下图所示：

检查 Ckafka 消息

1. 切换至 对象存储 COS 的控制台，查看对应的 bucket 下是否有文件生成。
   - 是，执行下一步。
   - 否，表示部署失败，请 提交工单 反馈。

2. 下载并查看新生成的文件，确认 Ckafka 的消息是否已按照换行写入。如下图所示：



- 是，表示部署成功。
- 否，表示部署失败，请 提交工单 反馈。

调整函数代码

本示例根据触发器配置的最大批量消息数触发云函数运行。

> **说明**
> 如果生产的消息数很多，且不想在 COS bucket 中生成过多的文件，可在云函数中使用 COS 的 SDK，以追加写入的方式减少文件数量。

云函数运行的操作流程如下：

1. 创建一个名称格式为 “时间戳+随机数” 的文本文件，例如2018-11-22-20:43:55-127.txt。
2. 将拉取到的 Ckafka 消息换行写入文本文件中。
3. 将写好的文件上传到 COS bucket，并以小时为单位在 COS bucket 中创建文件夹。

SCF + Ckafka 实现数据转储至 ES

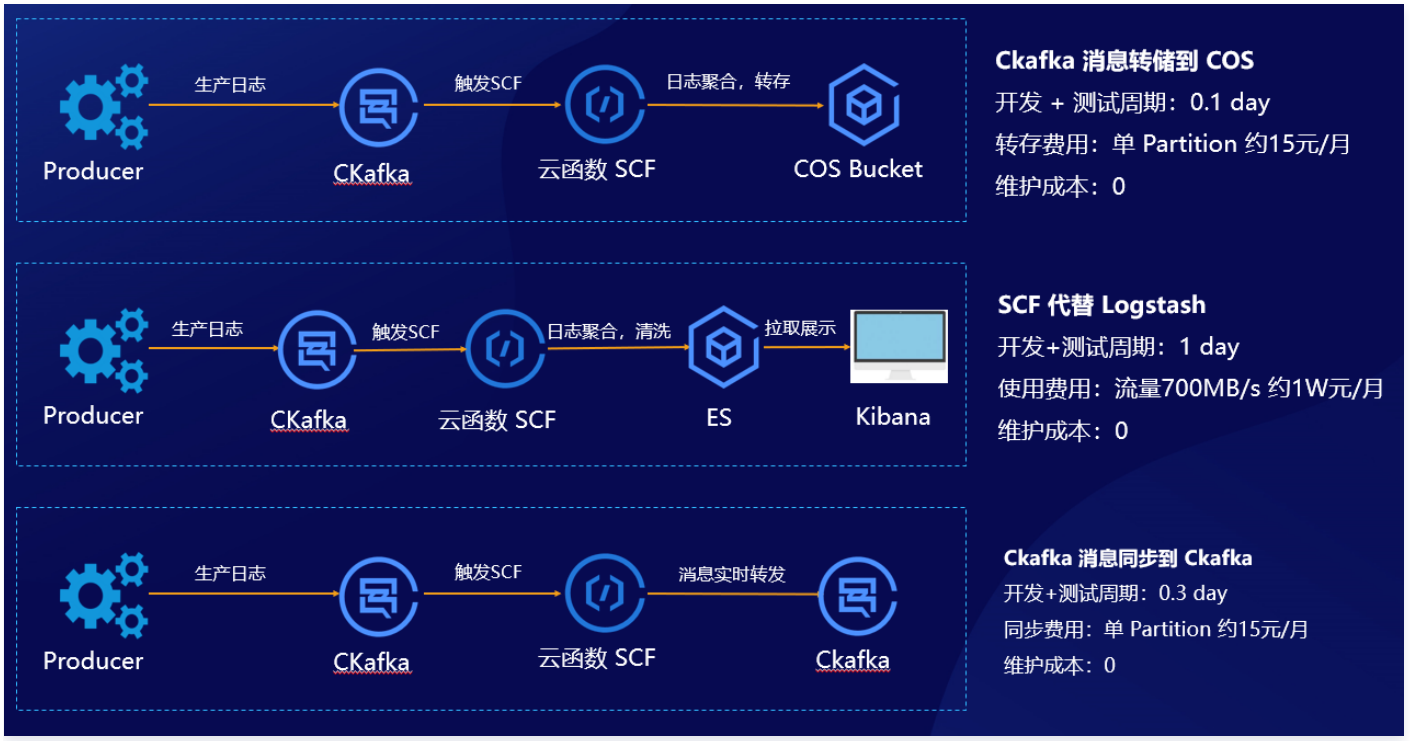
最近更新时间：2022-12-26 17:37:31

操作场景

随着 Kafka 社区的繁荣，越来越多的用户开始使用 Kafka 来进行日志收集、大数据分析、流式数据处理等操作。而腾讯云消息队列 Ckafka 也借助了开源社区的力量，进行了如下优化：

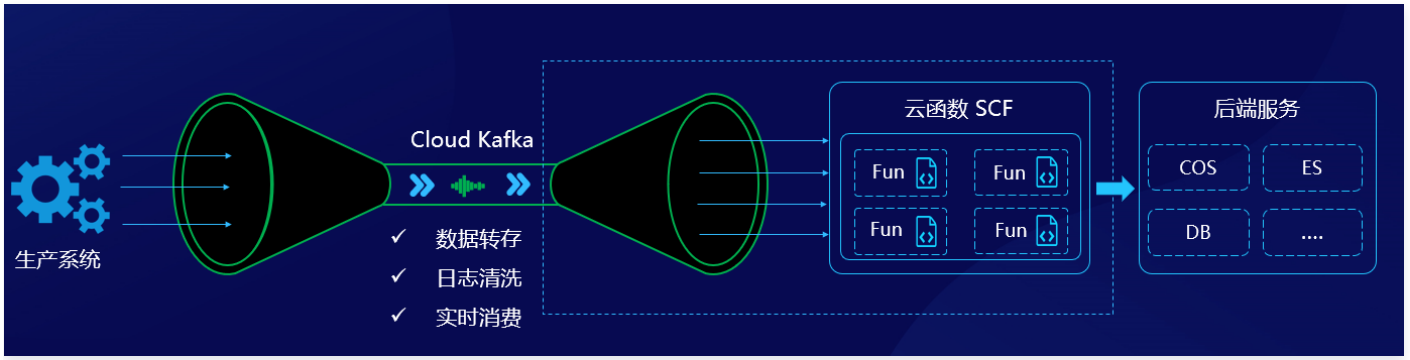
- 基于 ApacheKafka 的分布式、高可扩展、高吞吐。
- 100%兼容 Apache KafkaAPI（0.9及0.10）。
- 无需部署，直接使用 Kafka 所有功能。
- 封装所有集群细节，无需用户运维。
- 对消息引擎优化，性能比社区最高提升50%。

腾讯云云函数与 Ckafka 也进行了深度联动，并推出了很多实用的功能。借助云函数和 Ckafka 触发器，可以非常方便实现 CKafka 消息转存到 COS、ES、DB 等，本文介绍使用云函数替代 Logstash，实现 Ckafka 消息落盘 ES。如下图所示：



运行原理

云函数可以实时消费 Ckafka 中的消息。例如，做数据转存、日志清洗、实时消费等。且数据转存的功能已集成到 Ckafka 控制台，用户可以一键开启使用，降低了用户使用的复杂度。如下图所示：



方案优势

对比使用云服务器自建 Ckafka Consumer 的方式，云函数具备以下优势：

- 云函数控制台支持一键开启 Ckafka 触发器，帮助用户自动创建 Consumer，并由云函数平台来维护组件的高可用。
- Ckafka 触发器自身支持很多实用的配置：支持配置 offset 位置、支持配置1 – 1万消息聚合条数、支持配置1 – 1万次重试次数等。
- 基于云函数开发的业务逻辑，天然支持弹性伸缩，无需额外搭建和维护服务器集群等。

使用云函数和使用云服务器 CVM 自建 Logstash 对比，云函数具备以下优势：

- 云函数自带 Consumer 组件，可自行聚合。
- 云函数的模板函数已经实现了消息聚合和部分清洗能力，支持自行扩展。
- 云函数集群自带高可用和监控日志能力，业务上线速度更快。
- 云函数采用按实际使用收费，比自建集群费用更低廉，可以节省50%的费用。

前提条件

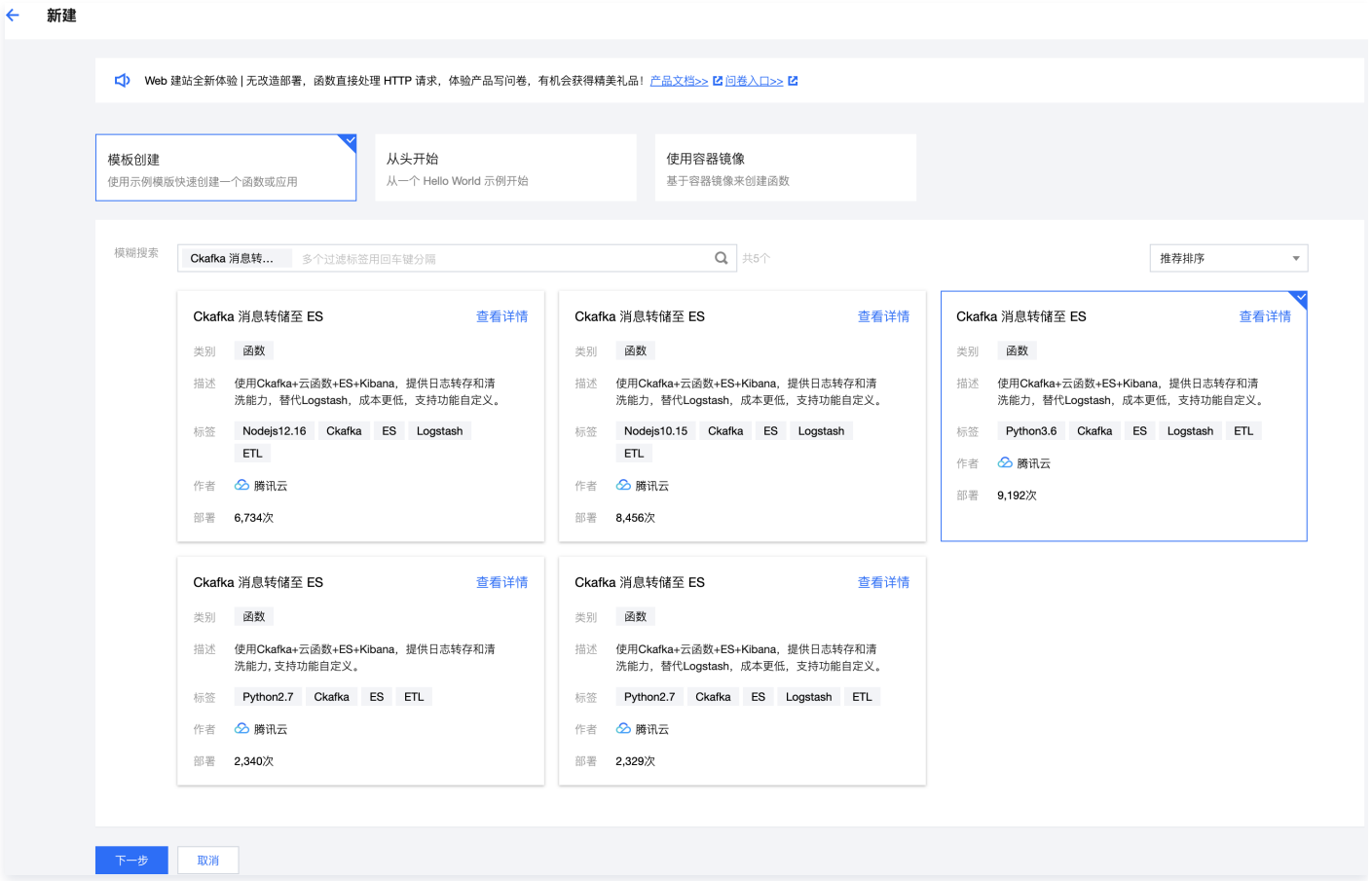
本文以**广州**地域为例：

- 需开启 Elasticsearch Service 服务。
- 需开启 Ckafka 服务。

操作步骤

创建云函数及 Ckafka 触发器

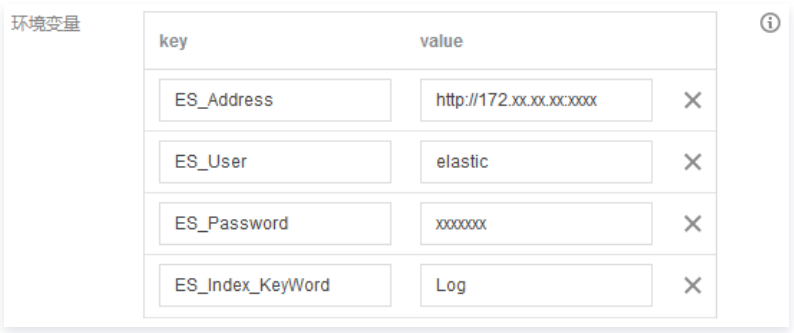
1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在“函数服务”上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
3. 在“新建函数”页面根据以下信息选择函数模板，并单击**下一步**。如下图所示：



- **创建方式**：选择**模板创建**。
- **模糊搜索**：输入“Ckafka 消息转储至 ES”，并进行搜索。本文以运行环境 Python3.6 为例。 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. 在**基础配置**中，函数名称已经自动生成，可根据需要自行修改。按照引导配置环境变量、运行角色和私有网络，如下图所示：

- **环境变量**：新增如下环境变量，参考表格进行填写。如下图所示：



key	value	是否必填
ES_Address	ES 服务地址	是
ES_User	ES 用户名，默认为 elastic。	是
ES_Password	ES 用户登录密码。	是
ES_Index_KeyWord	ES 关键词索引。	是
ES_Log_IgnoreWord	需要删除的关键词，缺省则全量写入。例如，填写 name 或 password。	否
ES_Index_TimeFormat	按照天或者小时设置 Index，缺省则按照天建立索引。例如填写 hour。	否

- **运行角色**：勾选“启用”，选择“配置并使用SCF模板运行角色”，将会自动创建并选择关联了 ES、Ckafka 全读写权限的 SCF 模板运行角色，或选择“使用已有角色”，在下拉列表中选择包含上述权限的已有角色。本文以“配置并使用SCF模板运行角色”为例。
- **私有网络**：勾选“启用”，并选择与 ES 相同的 VPC。

5. 在**触发器配置**中，选择“自定义创建”，根据页面的参数信息进行填写。如下图所示：

主要参数信息如下，其余参数请保持默认配置：

- **触发方式**：选择 “Ckafka触发”。
- **Ckafka实例及 Topic**：按需选择对应的 Topic。
- **起始位置**：选择“从最开始位置开始消费”。

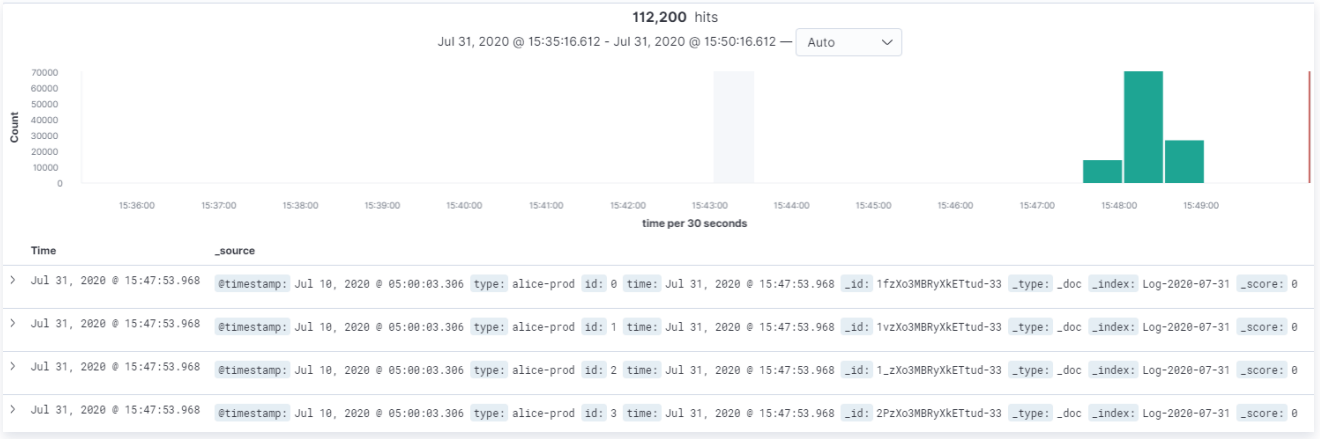
6. 单击**完成**，即可完成函数和触发器创建。

查看 ES 和函数运行日志

> **注意**
> 如果您还未将实际数据接入消息队列 Ckafka，您可以通过 客户端工具 模拟消息生产。

- 选择函数侧边栏**日志查询**，即可查看函数运行日志。如下图所示：

- 查看 Kibana。详情请参见 通过 Kibana 访问集群 。



扩展能力

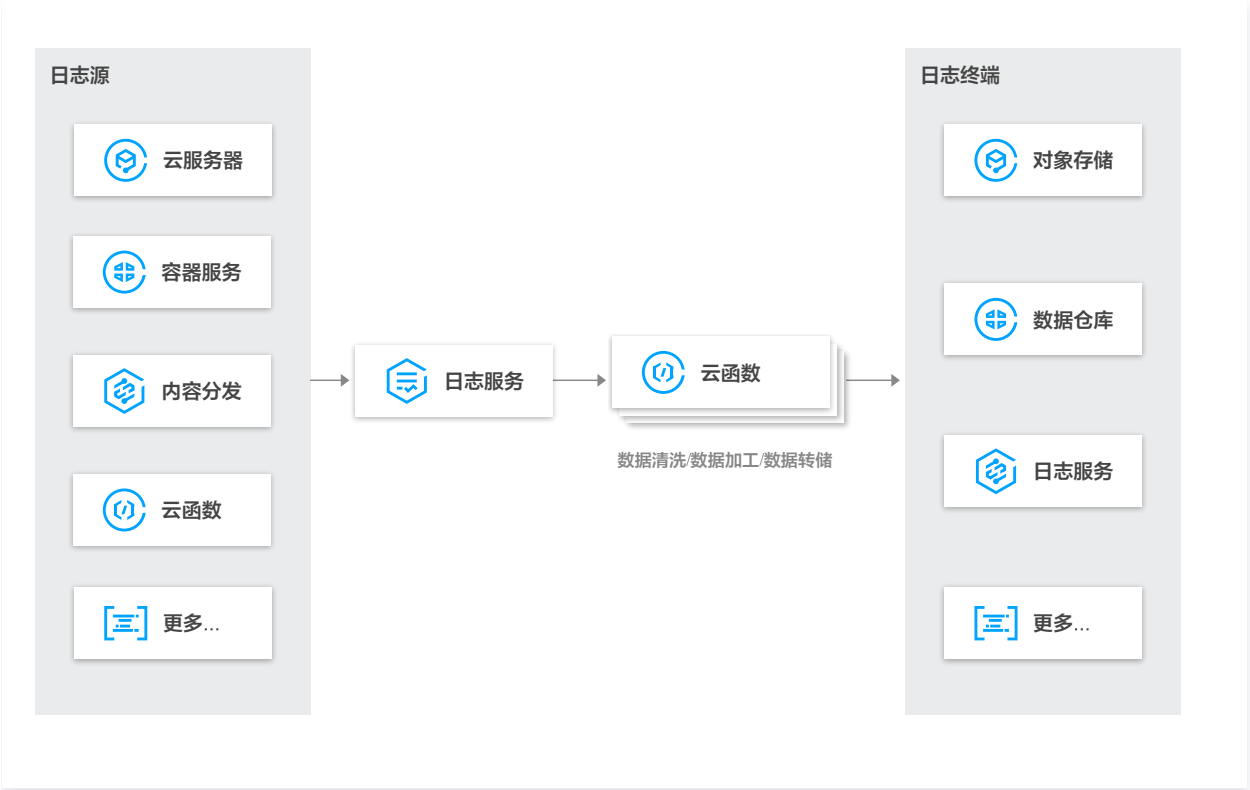
若您需实现高级日志清洗逻辑，可在如下图所示的代码位置中修改逻辑：

```
#日志清洗逻辑，可自行修改
def deallog(log):
    if ES_Log_IgnoreWord != None:
        for key in ES_Log_IgnoreWord.split(","):
            log.pop(key, None)
    log["time"] = datetime.datetime.now().isoformat()
    return log
```

日志服务 CLS
CLS 函数处理概述

最近更新时间：2024-04-17 16:23:11

通过函数处理服务，可以快速完成云服务器 CVM 等云上资源的运行日志采集、ETL（Extraction-Transformation-Loading）加工和消息转储等复杂日志处理任务。函数处理为异步过程，凡是收集到日志服务的数据，均能通过配置将数据投递到云函数进行消费处理，您只需要在日志服务控制台进行简单的配置即可完成日志服务 CLS 对接云函数消费。
通过 CLS 触发器 将日志源信息提交到 SCF，再通过 Serverless 无服务架构的函数计算提供数据加工与分析、事件触发、弹性伸缩，无需运维，按需付费。
整体数据处理流程如下所示：



使用函数处理优势

- 提供一站式采集、存储、加工、分析和展示。
- 全托管日志加工任务，按时间周期进行触发执行，自动重试。
- 持续增加内置函数模板，降低主流需求下的日志处理开发成本。
- 基于云函数提供数据加工及自定义代码逻辑。
- 基于云函数提供计算能力，拥有弹性伸缩、免运维、按需付费等特性。

多场景函数处理实践

日志服务可以将日志主题中的数据通过 CLS 日志触发器 投递至云函数进行处理，以满足日志加工/日志清洗等应用场景需求，场景及具体说明如下表所示：

函数处理场景	描述说明
CLS 转储至 Ckafka	日志数据通过云函数进行日志清洗等操作并投递至 Ckafka
CLS 转储至 COS	日志数据通过云函数进行日志清洗等操作并投递至 COS

> ⚠ **注意：**
> 数据投递至云函数，云函数侧将产生相应的计算费用，计费详情请参见 云函数 SCF 计费概述 。

SCF + CLS 日志转存至消息队列 CKafka

最近更新时间：2024-11-15 17:24:13

操作场景

本文为您介绍如何通过云函数 SCF 将 CLS 日志转储至消息队列 Ckafka。其中，CLS 主要用于日志采集，SCF 主要提供数据加工的节点计算能力，Ckafka 主要提供数据流终端转储能力。数据处理流程图请参见 函数处理概述 。

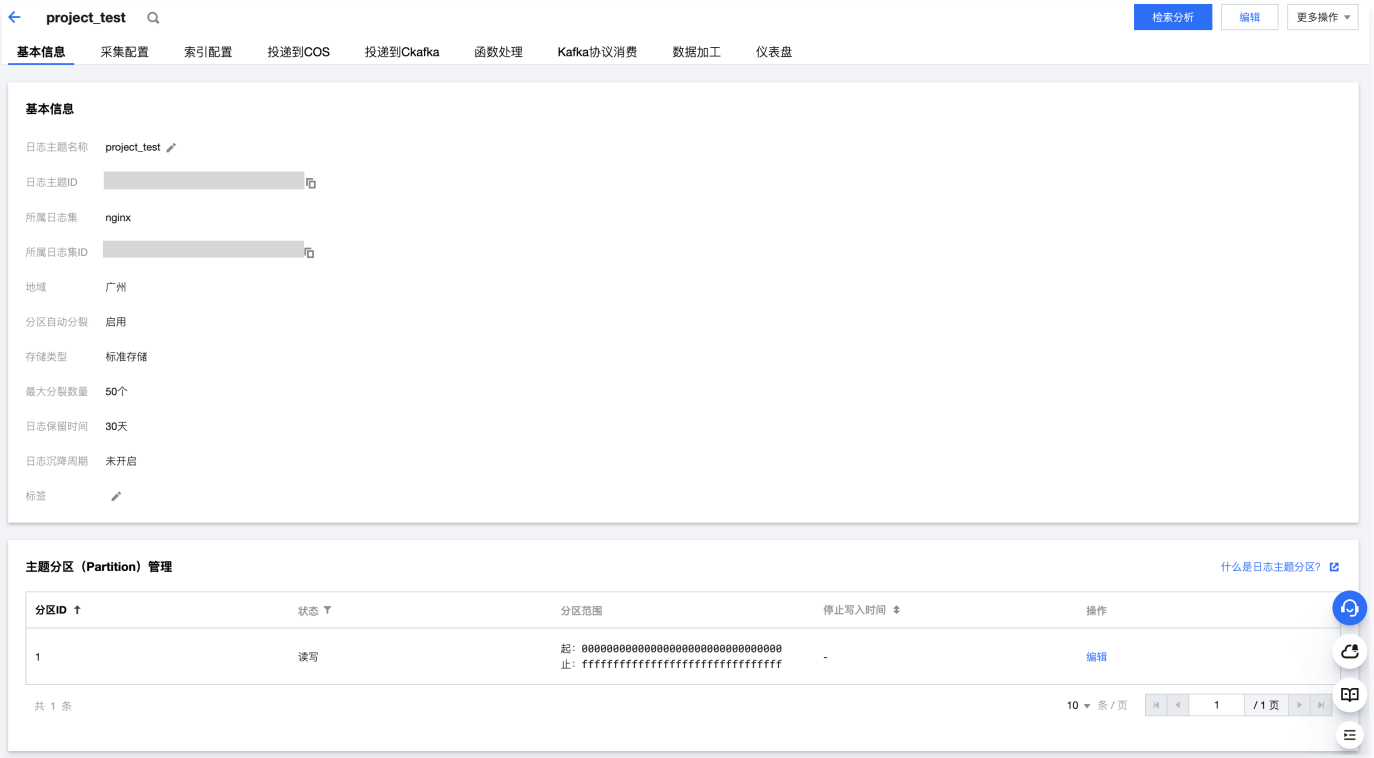
操作步骤

创建日志集和主题

1. 登录 日志服务控制台，在左侧导航栏中单击**日志主题**。
2. 进入日志集管理页面，在页面上方选择日志集的地域。
3. 单击**创建日志主题**，在弹出的创建日志集窗口中，填写相关信息：
   - 日志主题名称：例如 project_test
   - 日志集名称：例如 nginx

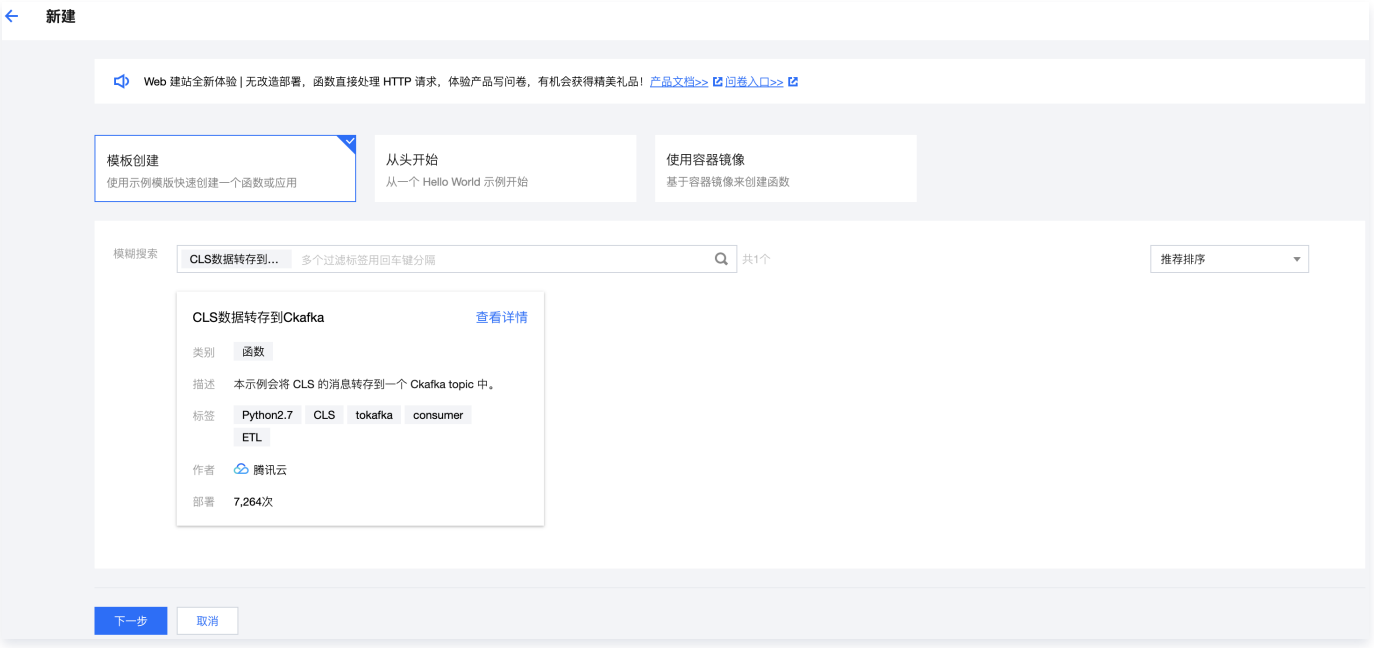
4. 单击**确定**，即可创建日志集和主题。

5. 日志主题新增成功，将进入日志主题管理页，如下图所示：



创建云函数 SCF

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面，配置以下参数：
   - **函数名称**：命名为 “CLSdemo”。
   - **运行环境**：选择 “Python 2.7”。
   - **创建方式**：选择**模板函数**。
   - **模糊搜索**：输入“CLS数据转存到CKAFKA”，并进行搜索。
3. 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. 基本信息配置完成之后，单击**下一步**，进入函数配置页面。
5. 函数配置保持默认配置，单击**完成**，完成函数的创建。

> **注意：**
> 函数需要在**函数配置**页面中，选择和 Ckafka 相同的 VPC 和子网。如下图所示：
>
> 
>

配置 CLS 触发器

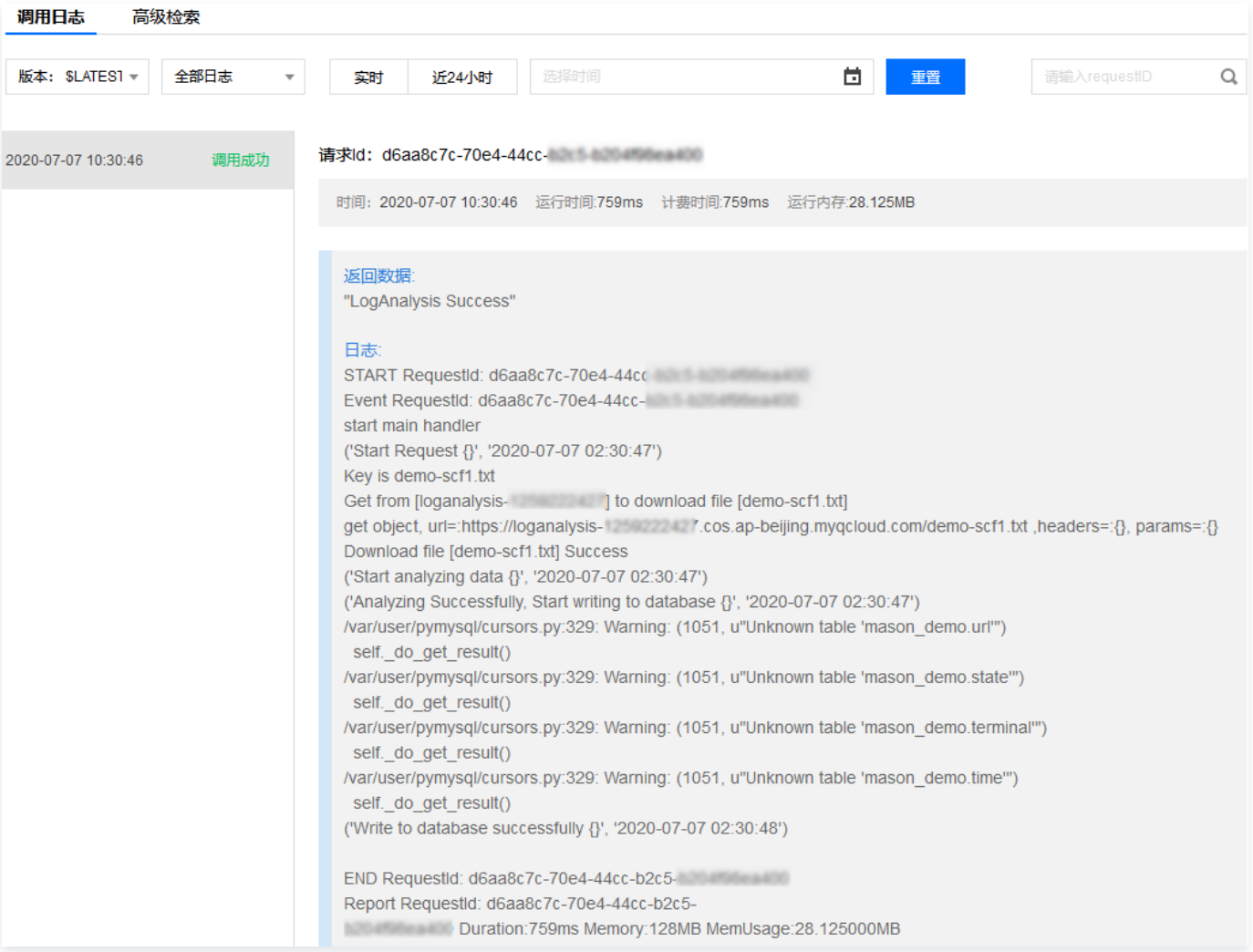
1. 登录 日志服务控制台，在左侧导航栏中单击**日志主题**。
2. 找到已创建的日志集，例如“project_test”，在其右侧操作栏中，单击**查看**，进入日志集详情页面。
3. 在日志主题详情页面，选择**函数处理**并单击右上角的**新建**。在弹出的“函数处理”窗口中添加已创建完成的函数。如下图所示：

主要参数信息如下，其余配置项请保持默认：

- **命名空间**：选择函数所在的命名空间。
- **函数名**：选择 创建云函数 SCF 步骤中已创建的云函数。
- **别名**：选择函数别名。
- **最长等待时间**：单次事件拉取的最长等待时间，默认60s。

测试函数功能

1. 下载 测试样例 中的日志文件，并解压出 demo-scf1.txt，导入至源端 CLS 服务。
2. 切换至 Serverless 控制台，查看执行结果。
   在函数详情页面中选择**日志查询**页签，可以看到打印出的日志信息。如下图所示：



3. 切换至 Ckafka 控制台，查看数据转储及加工结果。

> **说明：**
> 您可以根据自身的需求编写具体的数据加工处理方法。

SCF + CLS 日志转存至对象存储 COS

最近更新时间：2023-05-22 16:51:30

操作场景

本文为您介绍如何通过云函数 SCF 将 CLS 日志转储至对象存储 COS。其中，CLS 主要用于日志采集，SCF 主要提供数据加工的节点计算能力，COS 主要提供终端永久性存储能力。数据处理流程图请参见 函数处理概述 。

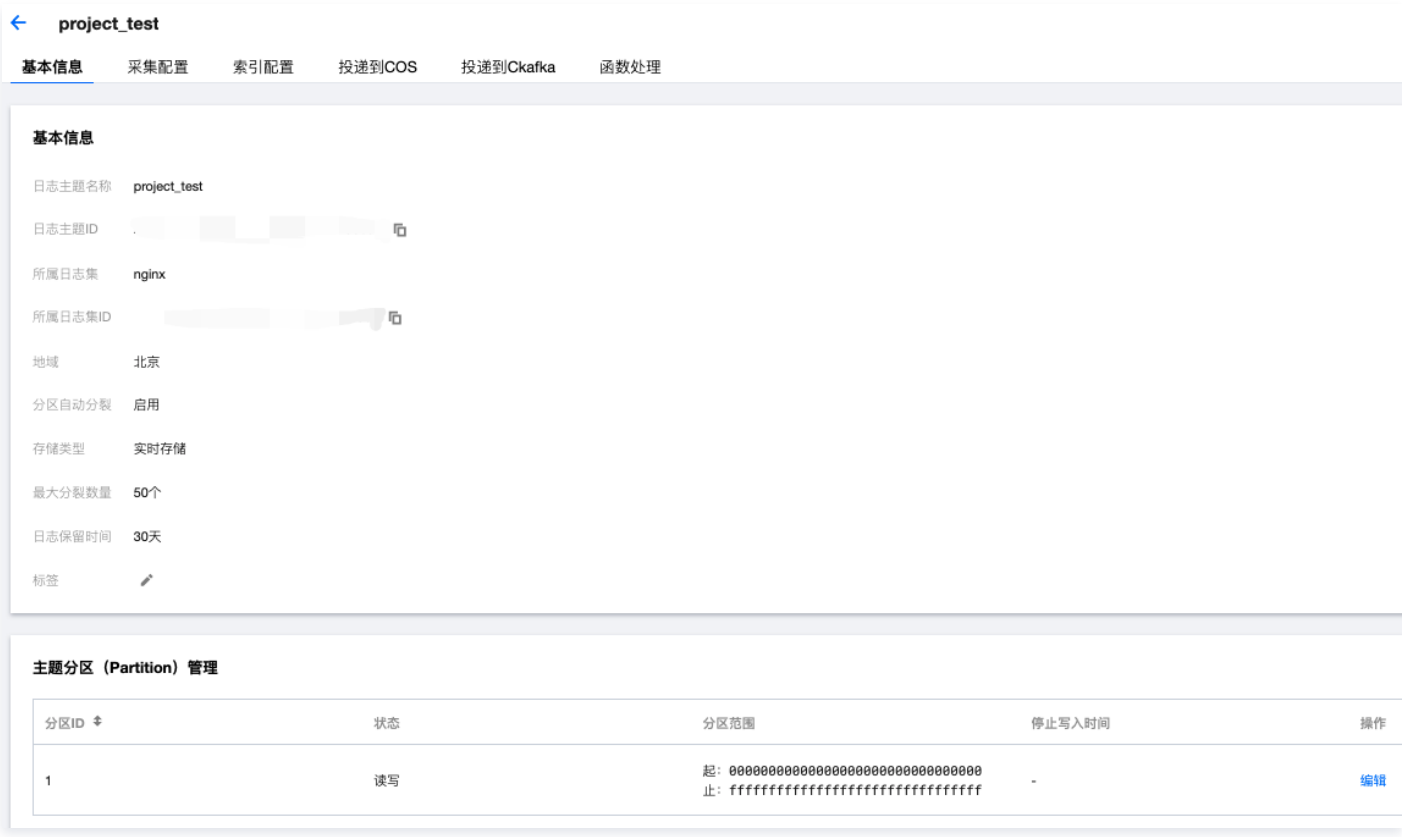
操作步骤

创建日志集和主题

1. 登录 日志服务控制台，在左侧导航栏中单击**日志主题**。
2. 进入日志集管理页面，在页面上方选择日志集的地域。
3. 单击**创建日志主题**，在弹出的创建日志集窗口中，填写相关信息：
   - 日志主题名称：例如 project_test
   - 日志集名称：例如nginx

4. 单击**确定**，即可创建日志集和主题。

5. 日志主题新增成功，将进入日志主题管理页，如下图所示：



创建云函数 SCF

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面，配置以下参数：
   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入“CLS 消息转储至 COS”，并进行搜索。
3. 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. 基本信息配置完成之后，单击**下一步**，进入函数配置页面。
   - **函数名称**：命名为 “CLSdemo”。
   - 选择**北京**地域
5. 函数配置保持默认配置，单击**完成**，完成函数的创建。

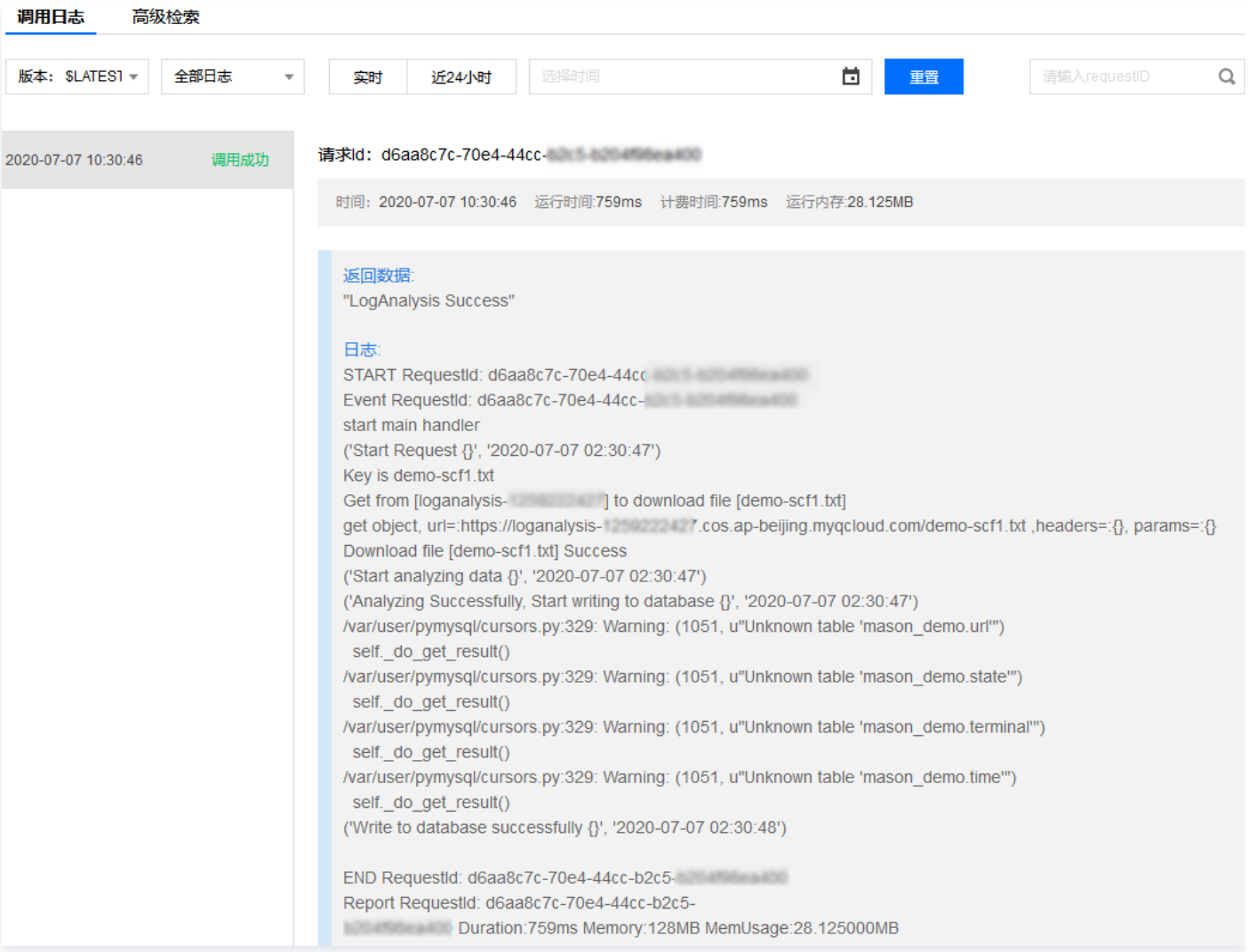
配置 CLS 触发器

1. 登录 日志服务控制台，在左侧导航栏中单击**日志集管理**。
2. 找到已创建的日志集，在其右侧操作栏中，单击**查看**，进入日志集详情页面。
3. 在日志主题详情页面，选择**函数处理**并单击**新建**。在弹出的“函数处理”窗口中添加已创完成的函数。如下图所示：

主要参数信息如下，其余配置项请保持默认：
- **命名空间**：选择函数所在的命名空间。
- **函数名**：选择 创建云函数 SCF 步骤中已创建的云函数。
- **别名**：选择函数别名。
- **最长等待时间**：单次事件拉取的最长等待时间，默认60s。

测试函数功能

1. 下载 测试样例 中的日志文件，并解压出 demo-scf1.txt，导入至源端CLS服务。
2. 切换至 Serverless 控制台，查看执行结果。
   在函数详情页面中选择**日志查询**页签，可以看到打印出的日志信息。如下图所示：



3. 切换至 对象存储 COS 控制台 ，查看数据转储及加工结果。

> **说明**
> 您可以根据自身的需求编写具体的数据加工处理方法。

负载均衡 CLB
SCF + CLB 快速部署 Web 服务

最近更新时间：2023-08-21 09:41:22

操作场景

本文将实践如何使用负载均衡 CLB 作为 Serverless 服务的访问入口，配合 SCF 快速部署 Web 服务。拓展 Serverless 服务低成本、免运维等优势，为开发者平滑迁移应用上云提供参考。

操作步骤

创建私有网络 VPC

登录 私有网络控制台，创建私有网络与子网。详情可参见 快速搭建私有网络 。

> **注意**
> 私有网络 VPC 需要与负载均衡 CLB 和云函数 SCF 网页部署在相同地区，本文以上海地域为例。

创建 CLB 实例

1. 登录 负载均衡控制台，创建负载均衡实例。详情可参见 创建负载均衡实例 。
   本文以上海地域为例，所属网络选择 上一步 中已创建的 VPC。
2. 创建完成后，您可以在实例管理页面中找到目标负载均衡实例，并为其配置监听器，详情请参见 配置 HTTP 监听器 。
   本文案例中以监听器名称为 clb-scf-web，监听协议端口为81为例。

创建云函数服务

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在**函数服务**页面上方选择**上海**地域，并单击**新建**进入新建函数页面。
   设置以下参数信息，并单击**下一步**。如下图所示：
   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入 “Web 静态页面托管” 和 “Python3.6” 进行搜索。
     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。

3. 在**基础配置**中，填写**函数名称**，选择**函数地域**。
   - **函数名称**：例如 `clb-scf-web` 。
   - **地域**：需要与 CLB 地域相同，例如 “上海” 。

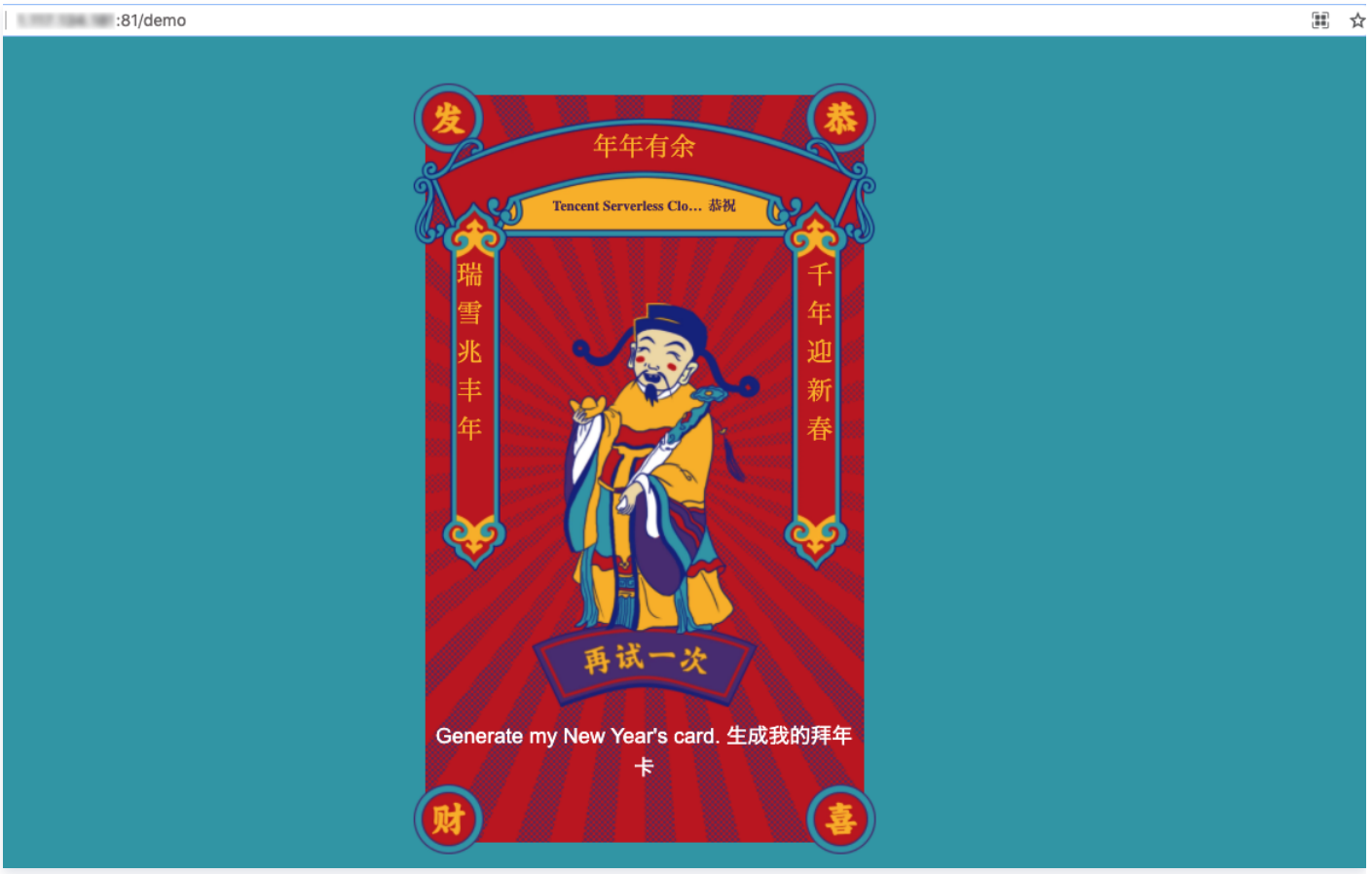
4. 在**触发器配置**中，选择“自定义创建”，使用触发器绑定 CLB 至 SCF。
   - 触发版本：选择“默认流量”。
   - 触发方式：选择“CLB触发”。
   - 实例ID：选择 上一步 中已创建的 CLB 实例，例如 `clb_serverless_web`。
   - 监听器：选择已配置的监听器，在本案例中监听器监听了端口 `81`。
   - 域名/主机：选择“新建规则”。
   - 新增域名：将 CLB 实例中的 “VIP” 填入新增域名。

     > **说明**
     > VIP 即负载均衡向客户端提供服务的 IP 地址。

   - URL路径：以 `/` 为开头添加本网站 URL 路径，例如 `/demo`。
5. 单击**完成**，跳转到**部署日志**中查看函数和触发器创建进度。

测试负载均衡入口

1. 登录 Serverless 控制台，进入函数服务页面。
2. 在**函数服务**页面上方单击已创建的函数 `clb-scf-web`。
3. 在该函数的详情页面，选择**触发管理**。
4. 在**触发管理**页中获取 API 网关触发器访问路径，查看 Web 页面。如下图所示：



视频处理 MPS
MPS 函数处理概述

最近更新时间：2021-11-15 14:47:01

通过函数处理服务，可以快速完成对 视频处理 MPS 产生的回调事件进行处理及操作。通过 MPS 触发器 将事件推送到云函数 SCF ，再通过 Serverless 无服务架构的函数计算提供回调事件的处理及响应。
整体数据处理流程如下图所示：

函数处理场景实践

日志服务可以将日志主题中的数据通过 MPS 日志触发器 投递至云函数进行处理，以满足对视频进行事件通知、状态监控、告警处理等应用场景需求，场景及具体说明如下表所示：

函数处理场景	描述说明
视频任务回调备份 COS	将 MPS 产生的回调任务通过 SCF 及时备份至 COS
视频任务回调通知工具	实时接收 MPS 数据消息，并将消息推送至企业微信、邮件等

注意
数据投递至云函数，云函数侧将产生相应的计算费用，计费详情请参见 SCF 计费概述 。

SCF + MPS 视频任务回调备份 COS

最近更新时间：2025-06-13 15:22:02

操作场景

本文为您介绍如何将 视频处理 MPS 产生的回调任务通过云函数 SCF 及时备份至 对象存储 COS 。其中，SCF 主要提供回调消息处理，MPS 主要用于视频处理任务，COS 主要提供终端永久性存储能力。

操作步骤

创建云函数

1. 登录 Serverless 控制台，单击左侧导航栏的函数服务。
2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面，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**模板创建**。
- **模糊搜索**：输入“MPS 消息转储至 COS”，并进行搜索。
  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。

3. 单击**下一步**，函数名称默认填充，可根据需要自行修改。

配置 MPS 触发器

1. 在**触发器配置**中，选择**自定义创建**，根据页面的参数信息进行填写。如下图所示：

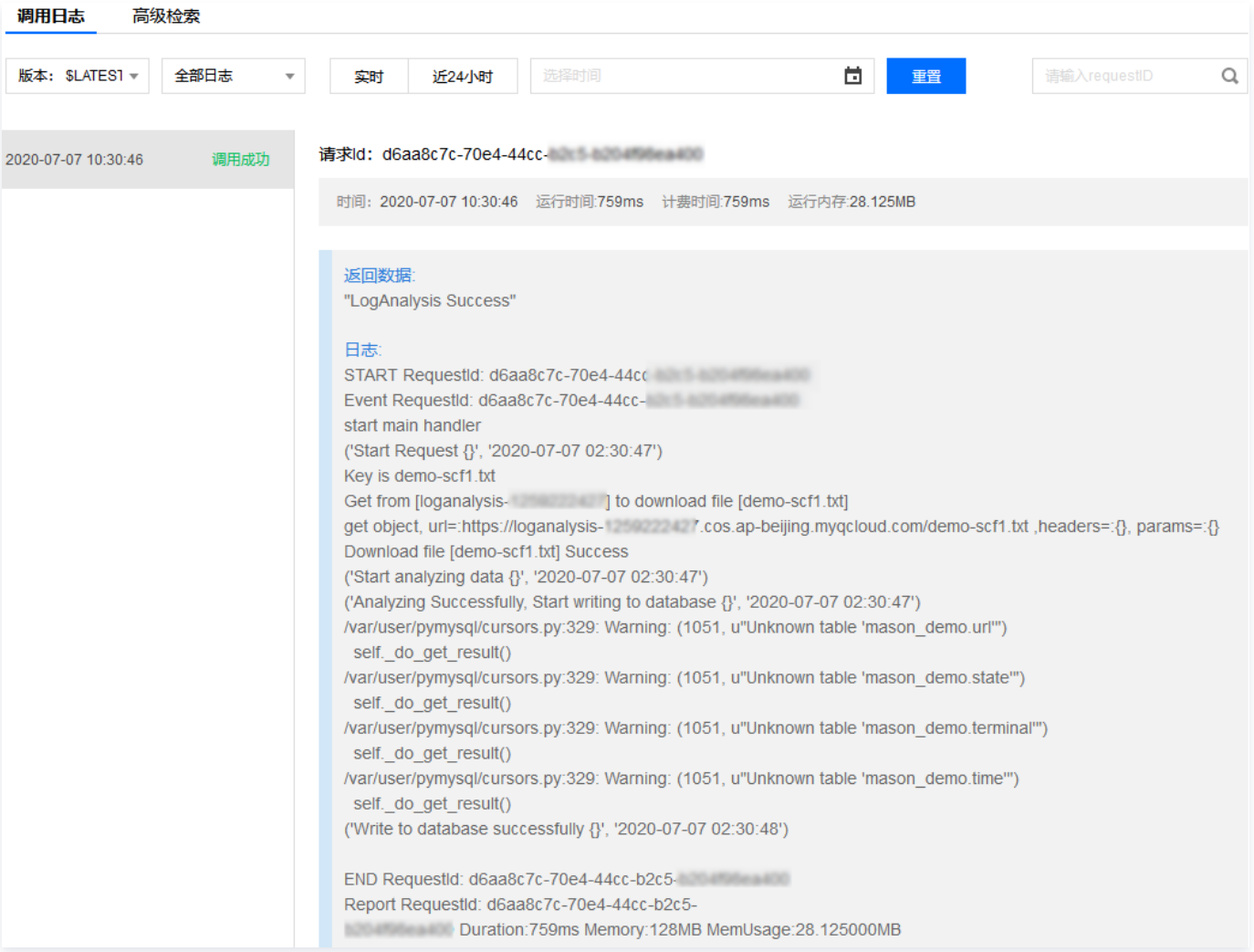
- **触发版本**：选择**默认流量**。
- **触发方式**：选择**MPS触发**。
- **事件类型**：选择**工作流任务**。

> **说明：**
> - 初次创建 MPS 触发器，需单击**SCF_QcsRole**、**MPS_QcsRole**对相关服务角色进行授权。
> - **事件类型**：MPS 触发器以账号维度的事件类型推送 Event 事件，目前支持工作流任务（WorkflowTask）和视频编辑任务（EditMediaTask）两种事件类型触发。
> - **事件处理**：MPS 触发器以服务维度产生的事件作为事件源，不区分地域、资源等属性。每个账号只允许两类事件分别绑定单个函数。如需多个函数并行处理任务，请参见 函数间调用 SDK 。

2. 单击**完成**即可完成函数创建和 MPS 触发器创建。

测试函数功能

1. 登录 视频处理控制台，执行视频处理工作流。
2. 在云函数控制台 函数服务 页面中，单击上述 创建云函数 步骤中创建的云函数名称，进入函数详情页面。
3. 在函数详情页面中选择**日志查询**页签，可以查看到打印出的日志信息。如下图所示：



4. 切换至 对象存储控制台，查看数据转储及加工结果。

> **说明：**
> 您可以根据自身的需求编写具体的数据加工处理方法。

SCF + MPS 视频任务回调通知工具

最近更新时间：2025-06-13 15:22:02

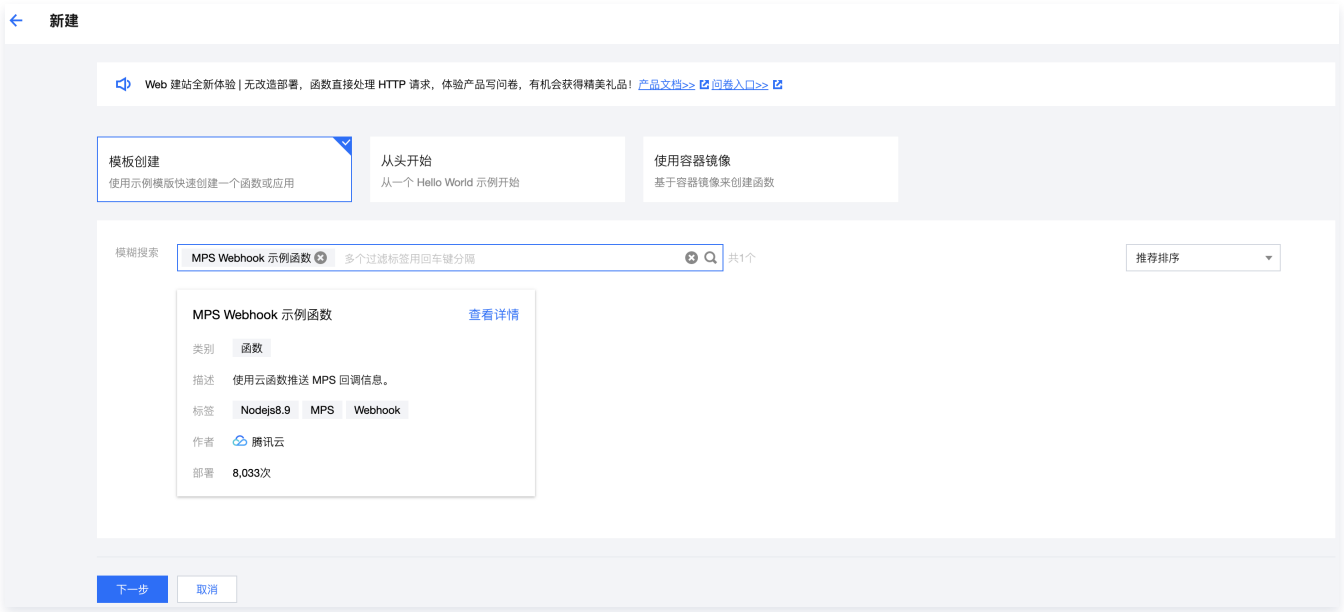
操作场景

本文为您介绍如何使用云函数 SCF 推送 视频处理 MPS 回调信息。其中，SCF 主要提供回调消息处理，MPS 主要用于视频处理任务。

操作步骤

创建云函数

1. 登录 Serverless 控制台，单击左侧导航栏的函数服务。
2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面，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**模板创建**。
   - **模糊搜索**：输入“MPS Webhook 示例函数”，并进行搜索。
     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。
3. 单击**下一步**，函数名称默认填充，可根据需要自行修改。

配置 MPS 触发器

1. 在**触发器配置**中，选择**自定义创建**，根据页面的参数信息进行填写。如下图所示：

   - **触发版本**：选择**默认流量**。
   - **触发方式**：选择**MPS触发**。
   - **事件类型**：选择**工作流任务**。

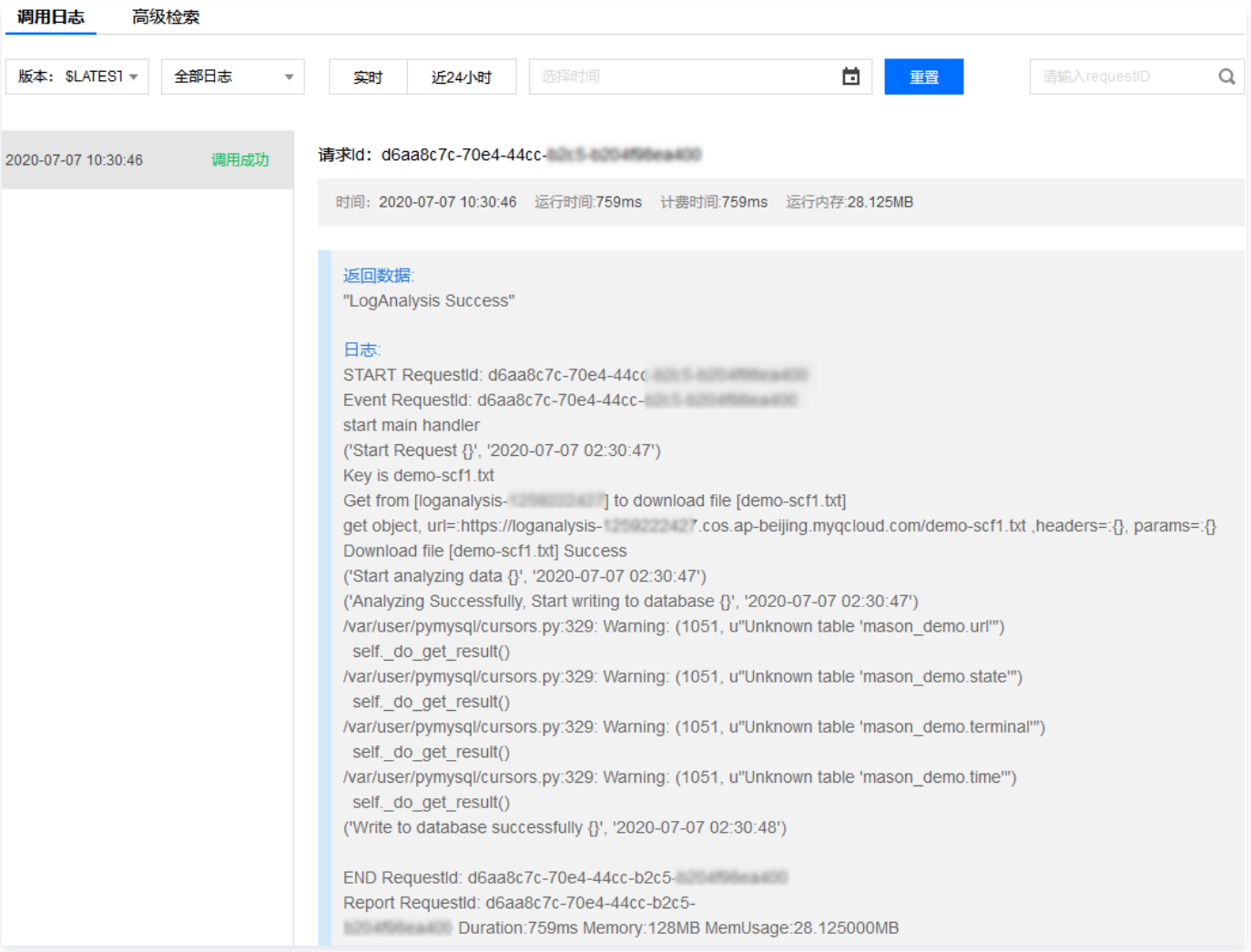
> **说明**

- 初次创建 MPS 触发器，需单击**SCF_QcsRole**、**MPS_QcsRole**对相关服务角色进行授权。
- **事件类型**：MPS 触发器以账号维度的事件类型推送 Event 事件，目前支持工作流任务（WorkflowTask）和视频编辑任务（EditMediaTask）两种事件类型触发。
- **事件处理**：MPS 触发器以服务维度产生的事件作为事件源，不区分地域、资源等属性。每个账号只允许两类事件分别绑定单个函数。如需多个函数并行处理任务，请参见 函数间调用 SDK 。

2. 单击**完成**即可完成函数创建和 MPS 触发器创建。

测试函数功能

1. 登录 视频处理控制台，执行视频处理工作流。
2. 在云函数控制台 函数服务 页面中，单击上述 创建云函数 步骤中创建的云函数名称，进入函数详情页面。
3. 在函数详情页面中选择**日志查询**页签，可以查看到打印出的日志信息。如下图所示：



4. 切换至**企业微信**，查看回调通知结果。

说明

> 您可以根据自身的需求编写具体的数据加工处理方法。

内容分发网络 CDN
SCF + CDN 实现定时预热刷新

最近更新时间：2023-09-11 21:26:32

定时刷新预热通过腾讯云 SCF 云函数，设置定时触发的刷新/预热任务。定时刷新预热任务被包括在每日刷新/预热的配额之内，执行当天如超过当日配额可能导致任务失败。

配置说明

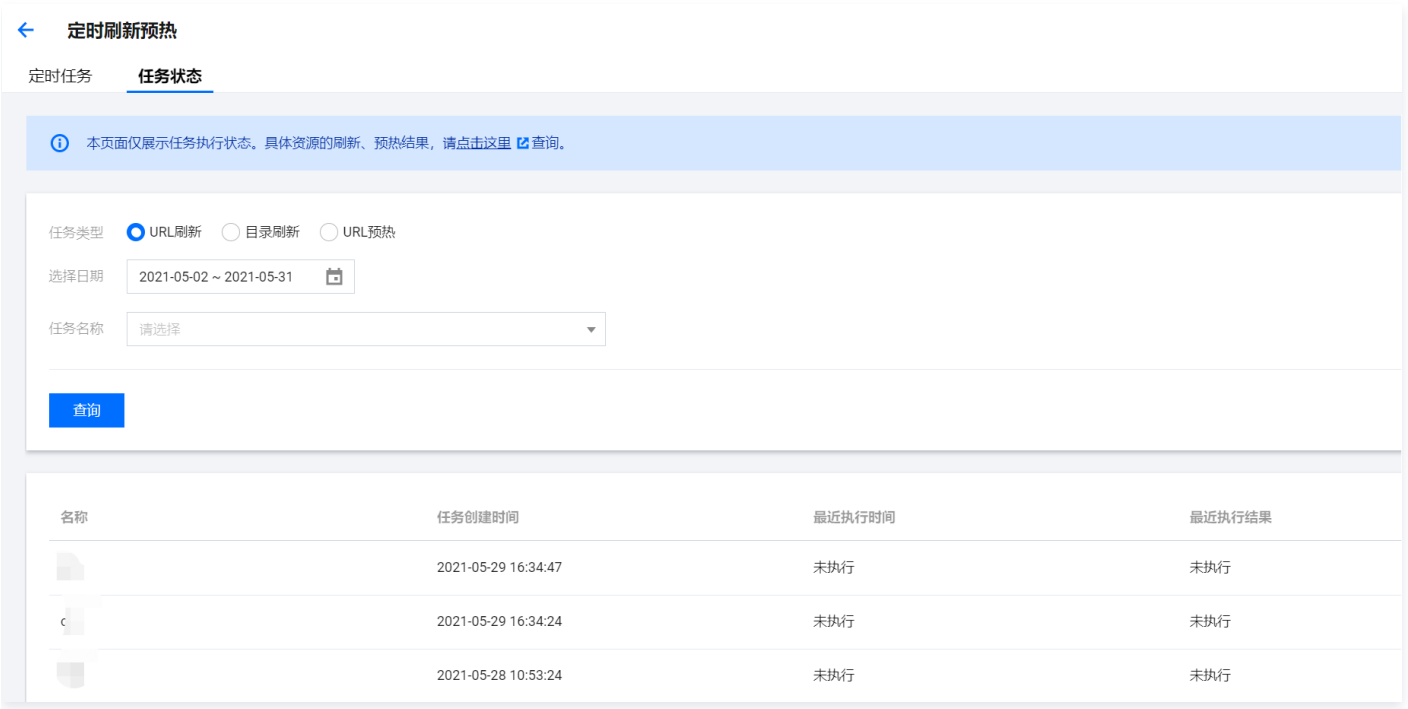
登录 CDN 控制台，在菜单栏里选择**插件中心**，单击定时刷新预热插件功能卡片，开通定时刷新预热，即可进入任务配置页面。首次开通之后，也可以单击卡片底部的**基础配置**进入定时刷新预热的任务列表页面进行配置。

在新建定时任务界面，选择相应的任务类型、设置 Cron 定时表达式（见下文）、输入对应的刷新/预热 URL，并进行 SCF 授权，系统即可自动生成对应的 SCF 云函数，并按时触发对应的任务。

注意

请不要在 SCF 控制台删除该云函数！

在任务状态页面，可以查看定时任务最近一次的执行情况。



Cron 表达式

Cron 表达式一共包含7个位值，每个位值之间必须用空格隔开。

位数	字段	取值范围	通配符
第一位	秒	0 – 59的整数	, – * /
第二位	分钟	0 – 59的整数	, – * /
第三位	小时	0 – 23的整数	, – * /
第四位	日	1 – 31的整数（需要考虑月的天数）	, – * /
第五位	月	1 – 12的整数或 JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC	, – * /
第六位	星期	0 – 6的整数或 SUN,MON,TUE,WED,THU,FRI,SAT。其中0指星期日，1指星期一，以此类推	, – * /
第七位	年	1970 – 2099的整数	, – * /

通配符的意义如下：

通配符	含义
,（逗号）	代表取用逗号隔开的字符的并集。例如：在“小时”字段中 1,2,3 表示1点、2点和3点
–（破折号）	包含指定范围的所有值。例如：在“日”字段中，1– 15包含指定月份的1号到15号
*（星号）	表示所有值。在“小时”字段中，* 表示每个小时
/（正斜杠）	指定增量。在“分钟”字段中，输入1/10以指定从第一分钟开始的每隔十分钟重复。例如，第11分钟、第21分钟和第31分钟，以此类推

注意：
在 Cron 表达式中的“日”和“星期”字段同时指定值时，两者为“或”关系，即两者的条件分别均生效。

示例

一次性任务

- `33 22 11 6 7 * 2021` 表示在 2021-7-6 11:22:33 触发任务
- `00 00 20 25 10 * 2021` 表示在 2021-10-25 20:00:00 触发任务

周期性任务

- `*/5 * * * * * *` 表示每5秒触发一次任务。
- `0 0 2 1 * * *` 表示在每月的1日的凌晨2点触发任务。
- `0 15 10 * * MON-FRI *` 表示在周一到周五每天上午10:15触发任务。
- `0 0 10,14,16 * * * *` 表示在每天上午10点，下午2点，4点触发任务。
- `0 */30 9-17 * * * *` 表示在每天上午9点到下午5点每半小时触发任务。
- `0 0 12 * * WED *` 表示在每个星期三中午12点触发任务。

费用说明

定时刷新预热功能本身免费，但是会调用 SCF 创建定时任务，超过 SCF 免费试用额度可能会产生云函数费用，具体请见 SCF免费额度 。

腾讯云数据仓库 TCHouse-P
SCF + TCHouse-P 导入 Ckafka 数据

最近更新时间：2024-10-21 14:56:11

背景说明

云函数是腾讯云为企业和开发者们提供的无服务器执行环境，具体可参见 云函数 SCF，下文简称 SCF。
腾讯云数据仓库 TCHouse-P 常见使用场景是将消息中间件的信息同步到腾讯云数据仓库 TCHouse-P 后再进行分析。本文提供了一种便捷的方法，即使用 SCF 实时的将 Kafka 中的数据导入到腾讯云数据仓库 TCHouse-P，无需用户维护任何服务。

注意事项

- 该云函数目前只能将腾讯云 CKafka 作为数据源，暂不支持自建 Kafka。
- 该云函数目前只能将腾讯云数据仓库 TCHouse-P 中的某一张表作为目标数据写入，如果有多张表的需求，请按照以下流程每张表创建对应的云函数。

使用步骤

步骤一：创建函数

1. 登录 云函数控制台，选择左侧导航中的**函数服务**。
2. 在**函数服务**页面，单击**新建**。
3. 在**新建**页面**模糊搜索**中搜索关键词“ckafka数据加载到CDW”，设置完成后单击**下一步**。

4. 进入**函数配置**页面后，配置参数请参见 创建函数，其中**环境配置**和**网络配置**，配置参数如下：
   - **环境配置**
     - 内存：根据实际运行情况来设置，默认为128MB。当导入过程中出现内存不足的问题时，需调大内存。
     - 环境变量：

参数	必填	说明
DB_DATABASE	是	数据库名
DB_HOST	是	如果函数是私有网络，并且和腾讯云数据仓库 TCHouse-P 是在同一子网，则可以填写腾讯云数据仓库 TCHouse-P 的内网 IP，否则需要填写外网 IP，并配置白名单
DB_USER	是	用户名
DB_PASSWORD	是	用户密码
DB_SCHEMA	是	模式名，如果创建表的时候未指定，通常是 public
DB_TABLE	是	表名

DB_PORT	否	腾讯云数据仓库 TCHouse-P 端口，默认为5436
MSG_SEPARATOR_ASCII	否	CKafka 中数据分隔符的 ASCII 码，默认为39（逗号），由于逗号经常会出现在业务数据中，这里建议使用11（Vertical tab）
MSG_NULL	否	CKafka 中消费的 NULL 值，默认是 \N
REPLACE_0X00	否	是否替换字符串中的0x00，默认是0（1表示替换）
ENABLE_DEBUG	否	是否打印错误的记录，默认是0（1表示打印）
ENABLE_COS	否	是否把未写入记录转储到 COS 上，默认是0（1表示转储）
COS_SECRET_ID	否	访问 COS 的 secret_id，ENABLE_COS 如果为1，该字段必填
COS_SECRET_KEY	否	访问 COS 的 secret_key，ENABLE_COS 如果为1，该字段必填
COS_BUCKET	否	COS 存储桶名称，ENABLE_COS 如果为1，该字段必填
STATMENT_TIMEOUT	否	查询超时时间，默认是50秒

- **网络配置**
  - 私有网络：建议**启用**私有网络，并将 VPC 和子网的值配置的与腾讯云数据仓库 TCHouse-P 相同。

下图为腾讯云数据仓库 TCHouse-P 对应的值。

  - 公网访问：建议同时**启用**公网访问。

5. 单击**完成**。

步骤二：配置触发器

1. 登录 云函数控制台，选择左侧导航中的**函数服务**。
2. 在**函数服务**页面，选择函数名，进入函数详情页面。
3. 选择左侧导航中的**触发管理**，单击**创建触发器**。
4. 在**创建触发器**页面，将**触发方式**设置为“Ckafka 触发”，如下图所示：

关于触发器参数配置可以参考 CKafka 触发器 。

5. 单击**提交**。

云点播 VOD
SCF + VOD 实现接收事件通知

最近更新时间：2024-12-02 21:19:43

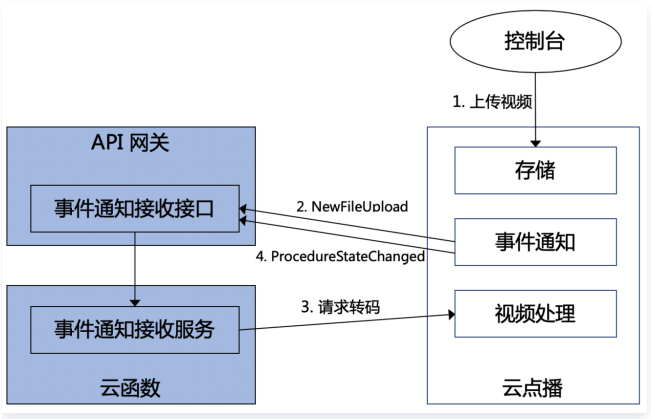
使用须知

Demo 功能介绍

本文以一个视频的上传、转码流程为例，向开发者展示云点播（VOD）事件通知机制 的使用方法。

架构和流程

Demo 基于云函数（SCF）搭建了一个 HTTP 服务，用于接收来自 VOD 的事件通知请求。该服务通过对 NewFileUpload（视频上传完成事件通知）和 ProcedureStateChanged（任务流状态变更）的处理，实现发起视频转码和获取转码结果。
系统主要涉及四个组成部分：控制台、API 网关、云函数和云点播，其中 API 网关和云函数即是本 Demo 的部署对象，如下图所示：



具体业务流程为：
1. 在控制台上传一个视频到 VOD。
2. VOD 后台发起 NewFileUpload 事件通知请求给 Demo。
3. Demo 解析事件通知内容，调用 VOD 的 ProcessMedia 接口对刚上传的视频发起转码，使用的转码模板为 系统预置模板 100010和100020。
4. VOD 完成转码任务后，发起 ProcedureStateChanged 事件通知请求给 Demo。
5. Demo 解析事件通知内容，将转码输出文件的 URL 打印到 SCF 日志中。

> **说明：**
> Demo 中的 SCF 代码使用 Python3.6 进行开发，此外 SCF 还支持 Python2.7、Node.js、Golang、PHP 和 Java 等多种编程语言，开发者可以根据情况自由选择，具体请参考 SCF 开发指南 。

费用

本文提供的云点播事件通知接收服务 Demo 是免费开源的，但在搭建和使用的过程中可能会产生以下费用：
- 购买腾讯云云服务器（CVM）用于执行服务部署脚本，详见 CVM 计费 。
- 使用 SCF 提供签名派发服务，详见 SCF 计费 和 SCF 免费额度 。
- 使用腾讯云 API 网关为 SCF 提供外网接口，详见 API 网关计费 。
- 消耗 VOD 存储用于存储上传的视频，详见 存储计费 和 存储资源包 。
- 消耗 VOD 转码时长用于对视频进行转码，详见 转码计费 和 转码资源包 。

避免影响生产环境

事件通知接收服务 Demo 的业务逻辑使用到 VOD 事件通知机制，因此部署过程中需要开发者配置事件通知地址。如果该账号已有基于 VOD 的生产环境，那么变更事件通知地址可能造成业务异常。**操作前请务必确认不会影响生产环境，如果您无法确定，请更换一个全新账号来部署 Demo。**

快速部署事件通知接收服务

步骤1：准备腾讯云 CVM

部署脚本需要运行在一台腾讯云 CVM 上，要求如下：

- 地域：任意。
- 机型：官网最低配置（1核1GB）即可。
- 公网：需要拥有公网 IP，带宽1Mbps或以上。
- 操作系统：官网公共镜像 `Ubuntu Server 16.04.1 LTS 64位` 或 `Ubuntu Server 18.04.1 LTS 64位` 。

购买 CVM 的方法请参见 操作指南 – 创建实例 。重装系统的方法请参见 操作指南 – 重装系统 。

> **注意**
> - 事件通知接收服务 Demo 本身并不依赖于 CVM，仅使用 CVM 来执行部署脚本。
> - 如果您没有符合上述条件的腾讯云 CVM，也可以在其它带外网的 Linux（如 CentOS、Debian 等）或 Mac 机器上执行部署脚本，但需根据操作系统的区别修改脚本中的个别命令，具体修改方式请开发者自行搜索。

步骤2：开通云点播

请参考 快速入门 – 步骤1 开通云点播服务。

步骤3：获取 API 密钥和 APPID

事件通知接收服务 Demo 的部署和运行过程需要使用到开发者的 API 密钥（即 SecretId 和 SecretKey）和 APPID。

- 如果还未创建过密钥，请参见 创建密钥文档 生成新的 API 密钥；如果已创建过密钥，请参见 查看密钥文档 获取 API 密钥。
- 在控制台 账号信息 页面可以查看 APPID，如下图所示：

步骤4：部署事件通知接收服务

登录 步骤1准备的 CVM（登录方法详见 操作指南 – 登录 Linux ），在远程终端输入以下命令并运行：

```
ubuntu@VM-69-2-ubuntu:~$ export SECRET_ID=AKxxxxxxxxxxxxxxxxxxxxx; export
SECRET_KEY=xxxxxxxxxxxxxxxxxxx;export APPID=125xxxxxxx;git clone https://github.com/tencentyun/vod-
server-demo.git ~/vod-server-demo; bash ~/vod-server-demo/installer/callback_scf.sh
```

> **说明：**
> 请将命令中的 SECRET_ID、SECRET_KEY 和 APPID 赋值为 步骤3 中获取到的内容。

该命令将从 Github 下载 Demo 源码并自动执行安装脚本。安装过程需几分钟（具体取决于 CVM 网络状况），期间远程终端会打印如下示例的信息：

```
[2020-06-05 17:16:08]开始检查npm。
[2020-06-05 17:16:12]npm 安装成功。
[2020-06-05 17:16:12]开始安装 ServerLess。
[2020-06-05 17:16:13]serverless 安装成功。
[2020-06-05 17:16:14]开始部署云点播事件通知接收服务。
[2020-06-05 17:16:24]云点播事件通知接收服务部署完成。
[2020-06-05 17:16:26]服务地址: https://service-xxxxxxxx-125xxxxxxx.gz.apigw.tencentcs.com/release/callback
```

复制输出日志中的事件通知接收服务地址（示例中的 `https://service-xxxxxxxx-125xxxxxxx.gz.apigw.tencentcs.com/release/callback` ）。

> **注意：**
> 如果输出日志中出现如下所示的警告，一般是由于 CVM 无法立即解析刚部署好的服务域名，可尝试忽略该警告。
>
> ```
> [2020-04-25 17:18:44]警告：事件通知接收服务测试不通过。
> ```

步骤5：配置事件通知地址

如 避免影响生产环境 一节所述，操作之前请先确认您的线上业务不依赖于 VOD 事件通知。
登录 云点播控制台，单击**设置**，回调模式选择**普通回调**，回调 URL 填写 步骤4 中获得的事件通知接收服务地址，回调事件全部勾选，然后单击**确定**。如下图所示：

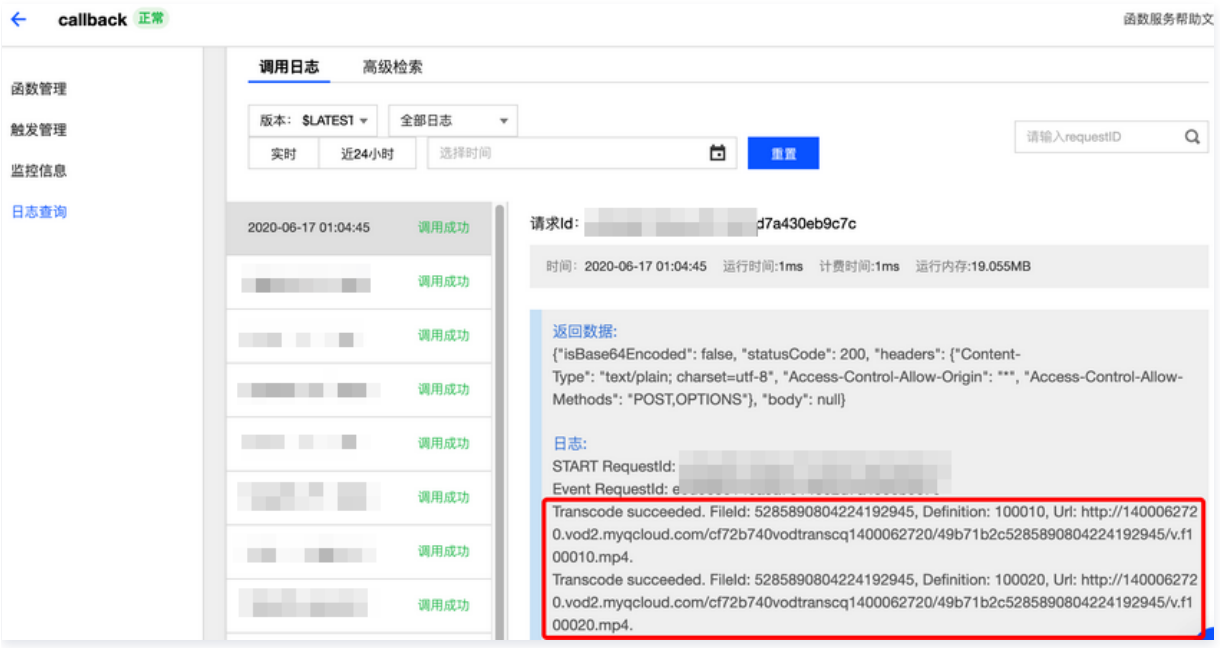
> **注意：**
> 如果您在控制台同时看到两个回调 URL 设置（2.0版本格式和3.0版本格式），请填写3.0版本。如下图所示：
>
> 
>

步骤6：测试 Demo

按照 上传视频 - 本地上传步骤 的说明，上传一个测试视频到云点播，注意上传过程选择默认的**只上传，暂不进行视频处理**。上传完成后，在**已上传**标签页可以看到该视频的状态为**处理中**，说明 Demo 接收到了 NewFileUpload 事件通知并发起了转码请求。

等待视频处理完成（状态变为“正常”）后，单击**快捷查看**，在页面右侧可以看到该视频有两个转码视频。如下图所示：

登录 SCF 控制台日志页面 查看 SCF 日志记录，在最新的一条日志中，可以看到两个转码文件的 URL 已经打印出来，在实际应用场景中，开发者可以通过 SCF 将 URL 记录在自己的数据库，或者通过其它渠道发布给观众。



> **说明：**
> SCF 日志可能会有些许延迟，如果在页面上没有看到日志，请耐心等待一两分钟，然后单击**重置**刷新。

系统设计说明

接口协议

事件通知接收云函数通过 API 网关对外提供接口，具体接口协议请参考文档 视频上传完成事件通知 和 任务流状态变更 。

事件通知接收服务代码解读

1. `main_handler()` 为入口函数。
2. 调用 `parse_conf_file()` ，从 `config.json` 文件中读取配置信息。配置项说明如下：

字段	数据类型	功能

secret_id	String	API 密钥
secret_key	String	API 密钥
region	String	云 API 请求地域，对于 VOD 可随意填写
definitions	Array of Integer	转码模板
subappid	Integer	事件通知是否来自 云点播子应用

3. 针对 NewFileUpload 类型的事件通知，调用 `deal_new_file_event()` 解析请求，从中取出新上传视频的 FileId：

```
        if event_type == "NewFileUpload":
            fileid = deal_new_file_event(body)
            if fileid is None:
                return ERR_RETURN
```

4. 调用 `trans_media()` 发起转码，输出云 API 的回包到 SCF 日志，并回包给 VOD 的事件通知服务：

```
            rsp = trans_media(configuration, fileid)
            if rsp is None:
                return ERR_RETURN
            print(rsp)
```

5. 在 `trans_media()` 中，调用云 API SDK 发起 `ProcessMedia` 请求：

```
    cred = credential.Credential(conf["secret_id"], conf["secret_key"])
    client = vod_client.VodClient(cred, conf["region"])

    method = getattr(models, API_NAME + "Request")
    req = method()
    req.from_json_string(json.dumps(params))

    method = getattr(client, API_NAME)
    rsp = method(req)
    return rsp
```

6. 针对 ProcedureStateChanged 类型的事件通知，调用 `deal_procedure_event()` 解析请求，从中取出转码输出视频的 URL 并打印到 SCF 日志：

```
        elif event_type == "ProcedureStateChanged":
            rsp = deal_procedure_event(body)
            if rsp is None:
                return ERR_RETURN
```

短信 SMS
SCF + SMS 实现短信验证码功能

最近更新时间：2024-11-27 15:21:02

通过手机短信发送验证码，是应用验证用户真实身份的常用方式。目前，短信验证码广泛应用于用户注册、密码找回、登录保护、身份认证、随机密码、交易确认等应用场景。
本文以使用 云函数 开发一个短信验证码登录注册服务为例，帮助您了解如何实现短信验证码功能。
除云函数之外，还可通过使用 发送短信接口 实现。

准备工作

- 已 注册腾讯云 账号，并完成 企业实名认证 。
- 已 购买 短信套餐包。
- 准备短信签名归属方资质证明文件，详细的文件清单以及规范请参见 签名审核标准 。本文以使用企业营业执照作为资质证明文件为例。
- 了解短信正文内容审核规范，详情请参见 正文模板审核标准 。
- 已获取短信应用的 SDKAppID。

相关资料

- Demo 源码
- 私有网络产品文档
- 云数据库 Redis 产品文档
- 云函数产品文档

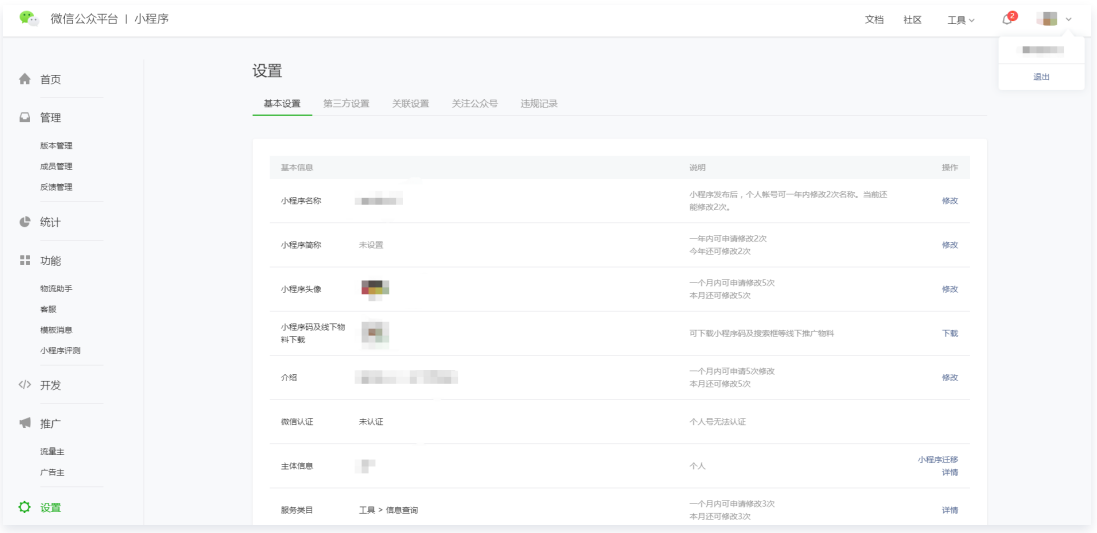
步骤1：配置短信内容

国内短信由签名+正文组成，签名符号为【 】（注：全角），发送短信时，必须选择签名+正文模板。短信签名、短信正文模板提交后，我们会在2个小时左右完成审核，您可以 配置告警联系人 并设置接收模板和签名审核通知，便于及时接收审核通知。

步骤1.1：创建签名

1. 登录 短信控制台 。
2. 在左侧导航栏选择**国内短信** > **签名管理**，单击**创建签名**。
3. 结合实际情况和 短信签名审核标准 设置以下参数：

参数	取值样例
签名用途	自用（签名为本账号实名认证的公司、网站、产品名等）
签名类型	App
签名内容	测试 Demo
证明类型	小程序设置页面截图

证明上传	

4. 单击**确定**。等待签名审核，当状态变为**已通过**时，短信签名才可用。

步骤1.2：创建正文模板

1. 登录 短信控制台 。
2. 在左侧导航栏选择**国内短信** > **正文模板管理**，单击**创建正文模板**。
3. 结合实际情况和 短信正文模板审核标准 设置以下参数：

参数	取值样例
模板名称	验证码短信
短信类型	普通短信
短信内容	您的注册验证码：{1}，请于{2}分钟内填写，如非本人操作，请忽略本短信。

4. 单击**确定**。等待正文模板审核，当状态变为**已通过**时，正文模板才可用，请记录模板 ID。

步骤2：设置短信发送频率限制（可选）

> **注意：**
> 个人认证用户不支持修改频率限制，如需使用该功能，请将 “个人认证” 变更为 “企业认证”，具体操作请参见 实名认证变更指引 。

为了保障业务和通道安全，减少业务被刷后的经济损失，建议 设置发送频率限制 。另外，您也可以结合使用 腾讯云验证码 以便最大程度地保护业务安全。
本文以短信的默认频率限制策略为例。

- 同一号码同一内容30秒内最多发送1条。
- 同一手机号一个自然日最多发送10条。

步骤3：配置私有网络和子网

默认情况下，云函数部署在公共网络中，只可以访问公网。如果开发者需要访问腾讯云的 TencentDB 等资源，需要建立私有网络来确保数据安全及连接安全。

1. 按需 规划网络 。
2. 创建私有网络，具体操作请参见 创建 VPC 。

> **注意：**
> 私有网络和子网的 CIDR 创建后不可修改。

参数	取值样例
所属地域	华南地区(广州)

名称	Demo VPC
IPv4 CIDR	10.0.0.0/16
子网名称	Demo 子网
IPv4 CIDR	10.0.0.0/16
可用区	广州三区

步骤4：配置 Redis 数据库

云数据库 Redis 实例需与 步骤3 配置私有网络的地域和子网的可用区保持一致。

1. 购买云数据库 Redis 实例，具体操作请参见 购买方式 。

参数	取值样例
计费模式	按量计费
地域	广州
数据库版本	Redis 4.0
架构	标准架构
网络	Demo VPC，Demo 子网
实例名	立即命名：Demo 数据库
购买数量	1

步骤5：新建云函数

云函数目前支持 Python、Node.js、PHP、Java 以及 Golang 语言开发，本文以 Node.js 为例。

1. 在 步骤3 创建的 VPC 所属地域中新建函数，具体操作请参见 编写函数 。

参数	取值样例
函数名称	Demo
运行环境	Nodejs 8.9
创建方式	模板函数：helloworld

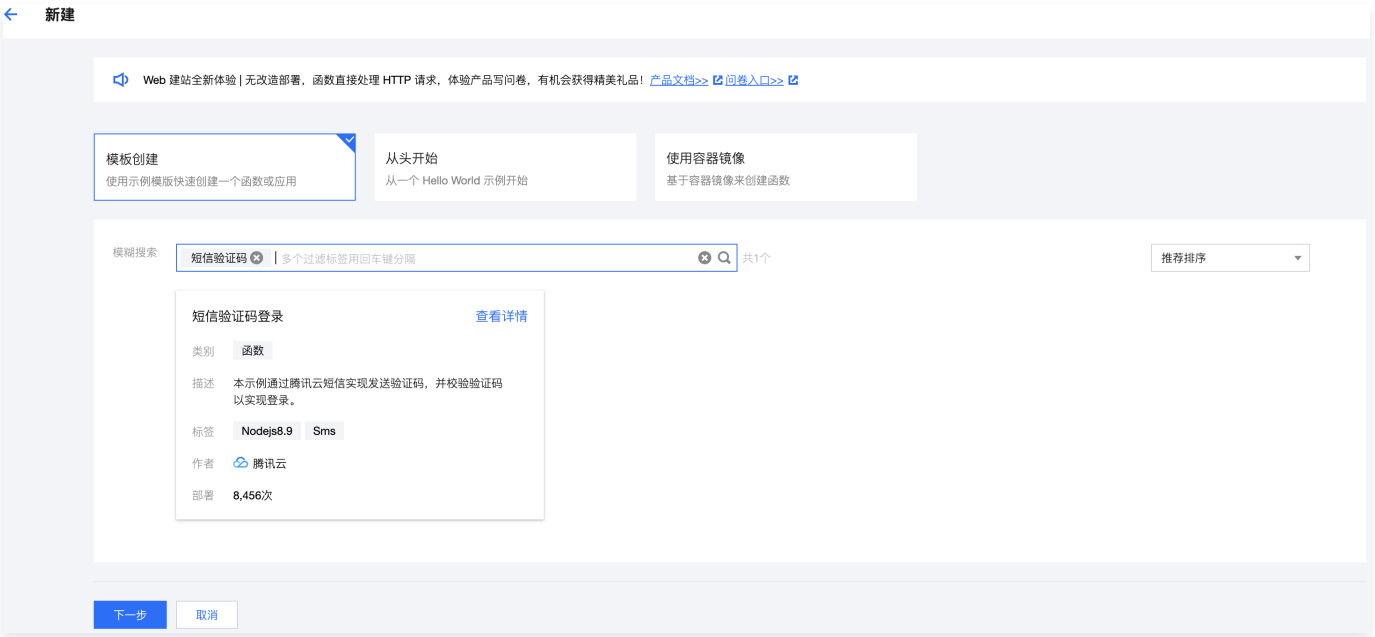
2. 部署函数并配置触发方式为**API网关触发器**，具体操作请参见 部署函数 。

步骤6：启用公网访问配置（可选）

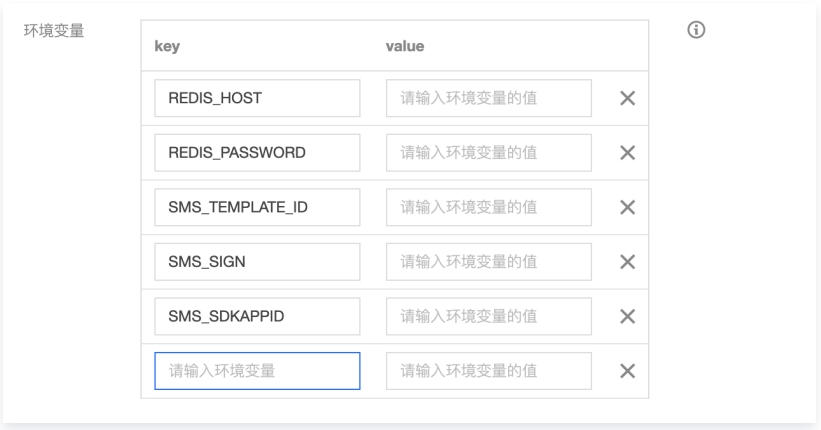
- 2020年4月29日前，部署在 VPC 中的云函数默认隔离外网。若需使云函数同时具备内网访问和外网访问能力，可通过启用公网配置方式实现。
  登录 云函数控制台，选择**函数服务**，在云函数列表中单击目标函数名进入函数配置页。单击**编辑**，勾选**公网访问**并单击**保存**保存配置。
- 2020年4月29日及以后，新部署的云函数默认已启用公网访问，无需额外操作。

步骤7：部署短信 Demo

1. 前往 云函数控制台 并选择 SMS Demo 进行部署。

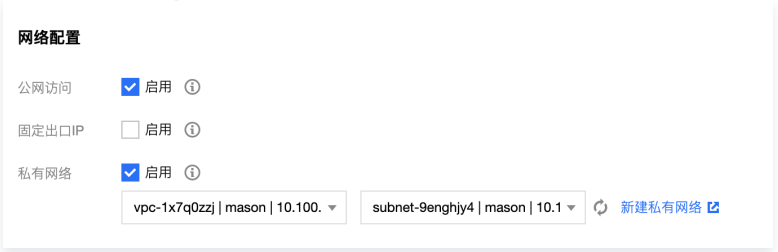


2. 在**高级配置**中设置 Demo 的环境变量。

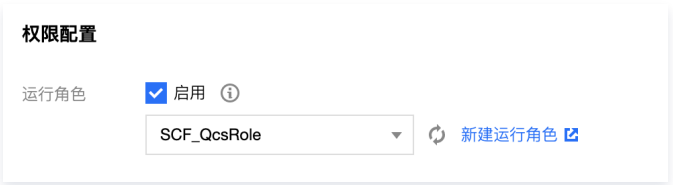


字段	说明
REDIS_HOST	Redis 数据库地址
REDIS_PASSWORD	Redis 数据库密码
SMS_TEMPLATE_ID	模板 ID，必须填写已审核通过的模板 ID。模板 ID 可登录 短信控制台 查看。
SMS_SIGN	短信签名内容，使用 UTF-8 编码，必须填写已审核通过的签名，签名信息可登录 短信控制台 查看。注：国内短信为必填参数。
SMS_SDKAPPID	短信 SdkAppid 在 短信控制台 添加应用后生成的实际 SdkAppid，示例如1400006666。

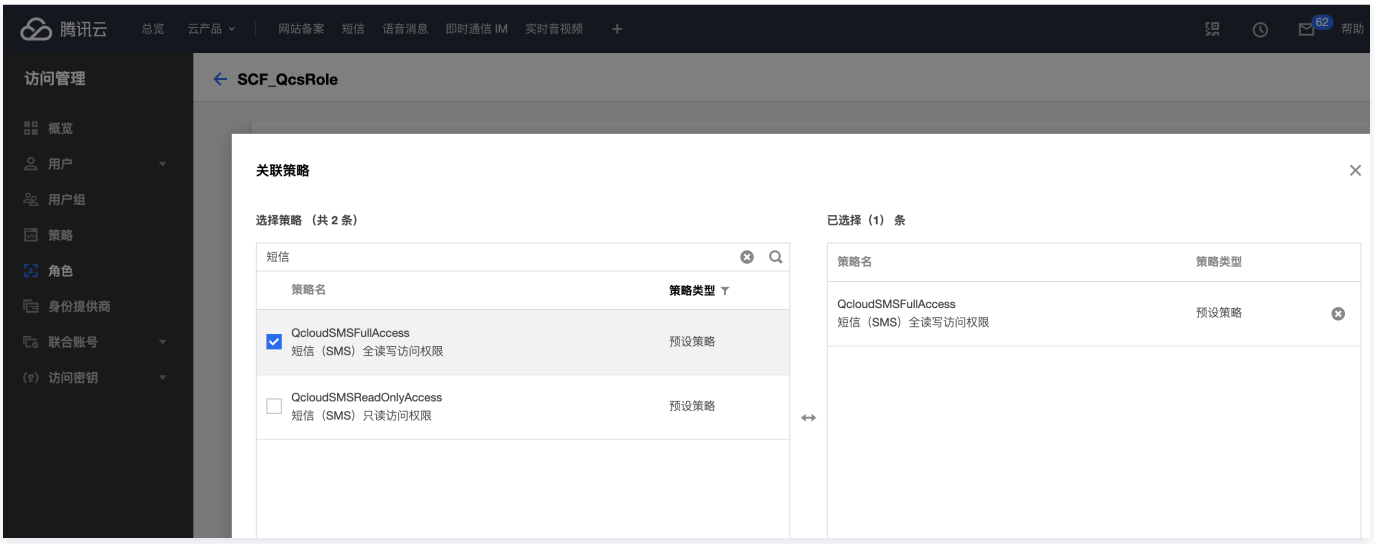
3. 在**高级配置**中设置与 Redis 数据库相同的 VPC 环境。



4. 在**高级配置**中设置 SCF **运行角色**权限。



需要在 访问管理 控制台给 SCF_QcsRole 角色添加短信 QcloudSMSFullAccess 权限。



这样代码里就能获取到 `TENCENTCLOUD_SECRETID`、`TENCENTCLOUD_SECRETKEY`、`TENCENTCLOUD_SESSIONTOKEN` 环境变量了，发送短信的 sdk 会用到这些环境变量。

5. 单击**完成**，即可完成函数部署。
6. 创建函数 **API网关触发器**，请求该触发器地址，即可使用短信相关能力。

步骤8：功能使用及说明

验证码的时效性要求较高，您可以把验证码存在内存中或存在云数据库 Redis 中。以手机号作为 key，存储发送时间、验证码、验证次数、是否已验证过等信息。

功能

发送短信验证码

请求参数：

字段	类型	说明
method	string	请求方法，值为 getSms
phone	string	手机号，值为区号+手机号，例如86185662466**

校验验证码（登录）

请求参数：

字段	类型	说明
method	string	请求方法，值为 login
phone	string	手机号，值为区号+手机号，例如86185662466**
code	string	值为6位数字验证码

错误码

字段	说明
InValidParam	缺少参数
MissingCode	缺少验证码参数
CodeHasExpired	验证码已过期
CodeHasValid	验证码已失效
CodeIsError	请检查手机号和验证码是否正确

如有任何疑问，请联系 腾讯云短信小助手，将有专人为您解答。

Elasticsearch Service
SCF + ES 快速构建搜索服务

最近更新时间：2024-08-21 17:10:02

搜索服务

搜索服务广泛的存在于我们身边，例如我们生活中用的百度、工作中用的 wiki 搜索、淘宝时用的商品搜索等。这些场景的数据具有数据量大、结构化、读多写少等特点，而传统的数据库的事务特性在搜索场景并没有很好的使用空间，并且在全文检索方面速度慢（如 like 语句）。因此，Elasticsearch 应运而生。
Elasticsearch 是一个广泛应用于全文搜索领域的开源搜索引擎，它可以快速地索引、搜索和分析海量的文本数据。腾讯云 ES 是基于 Elasticsearch 构建的高可用、可伸缩的云端托管 Elasticsearch 服务，对结构化和非结构化的数据都有良好的支持，同时还提供了简单易用的 RESTful API 和各种语言的客户端，方便快速搭建稳定的搜索服务。
本文将针对搜索场景，使用腾讯云 ES 官方文档作为语料，介绍如何使用腾讯云 ES+SCF 快速搭建搜索服务。搜索服务界面示例如下：

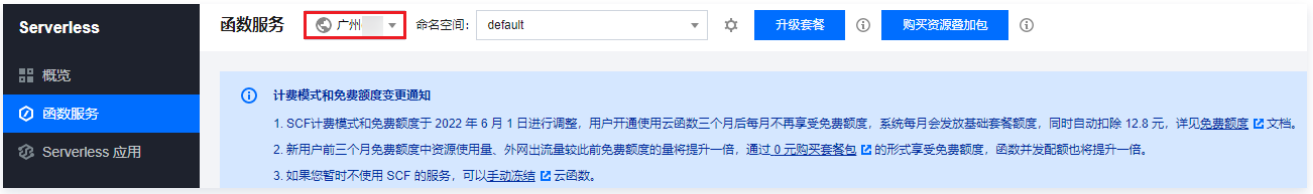
资源准备

只需要一个 ES 集群。在腾讯云购买一个 ES 集群，集群的规模根据搜索服务的 QPS 和存入的文档的数据量而定，详情请参见 集群规格和容量配置评估 。

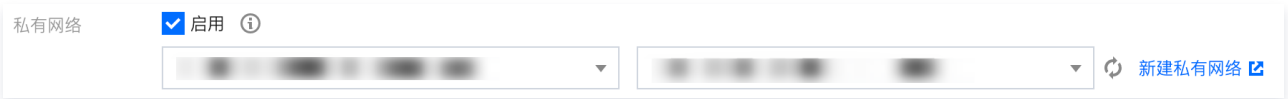
部署搜索服务

使用腾讯云 SCF 部署搜索服务的前端界面和后台服务。

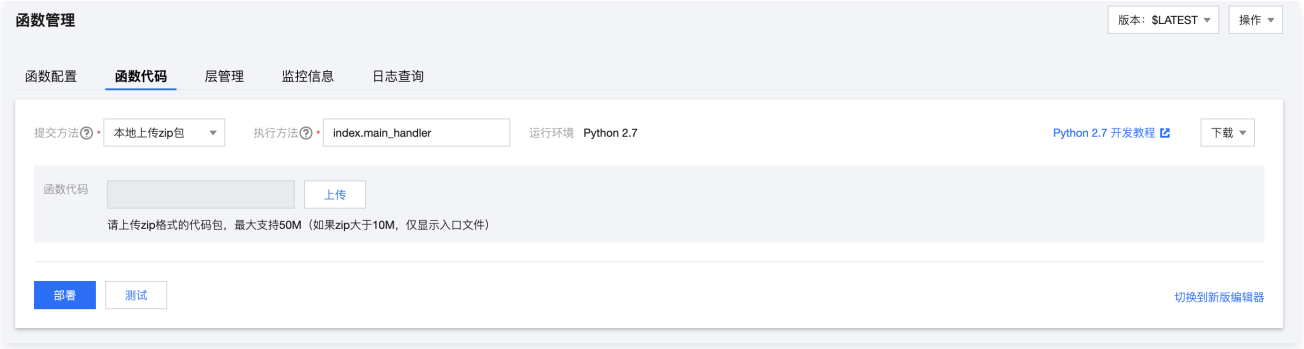
1. 在**云函数** > **函数服务** 界面左上角首先选择您购买 ES 集群的地域。



2. 单击**新建**，创建一个函数服务，操作详情见 创建函数 。
3. 函数创建完成后，单击**函数名称**进入函数详情页。
4. 在**函数配置**页单击右上角的**编辑**，开启**内网访问**，并选择您创建 ES 所选的 VPC，然后单击**保存**。

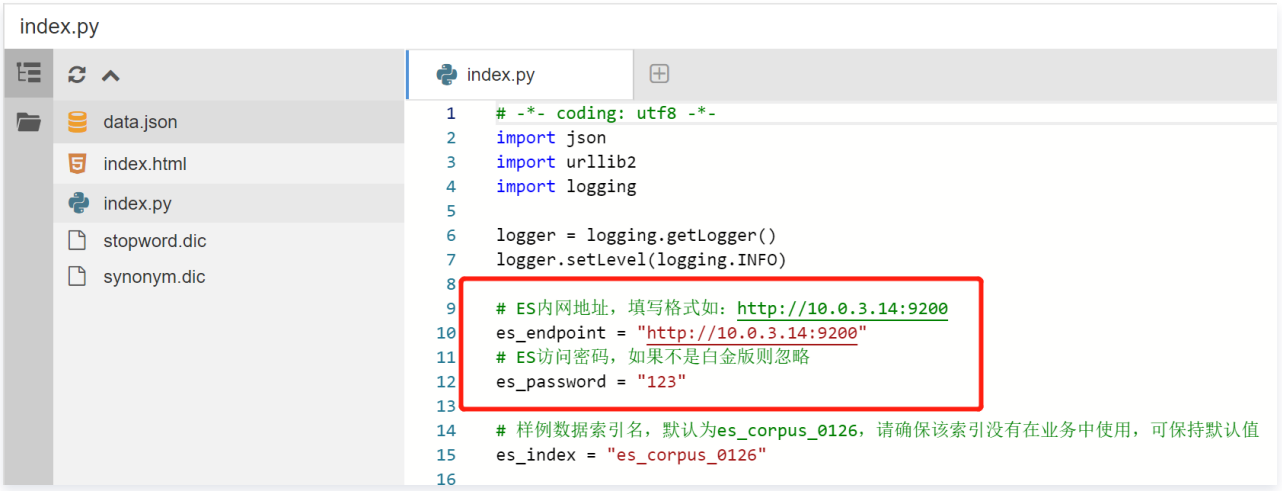


5. 单击 代码 zip 包 将示例文件下载到本地。
6. 在**函数代码**页的**提交方法**中选择上传本地 zip 包，并选择刚下载的 zip 包，单击**部署**。



7. 在**函数代码**页修改代码。需要修改的文件有 `index.py` 和 `index.html`：
   - `index.py` 中的 `es_endpoint` 修改为您的 ES 集群的内网地址，填写格式如：`http://10.0.3.14:9200`。
   - `index.py` 中的 `es_password` 修改为白金版 ES 密码，如果不是白金版则不修改。

index.py



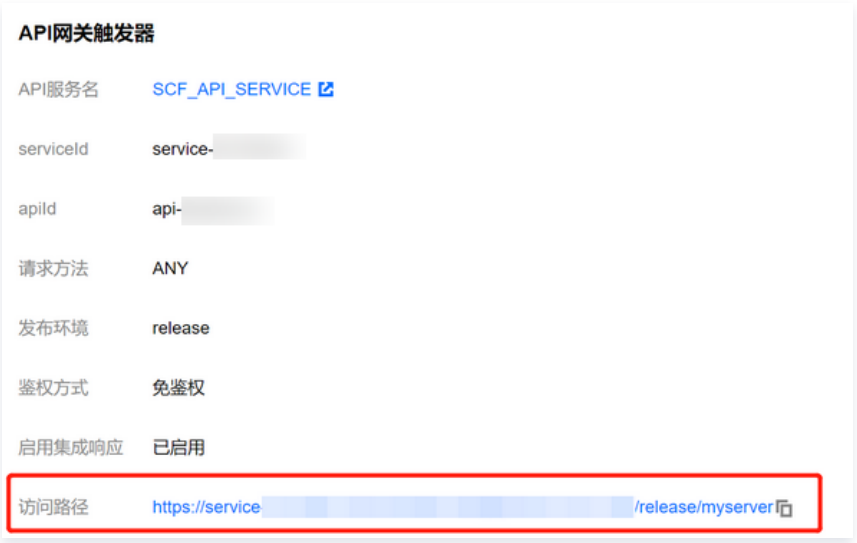
   - `index.html` 中的 `server_name` 修改为您创建的 SCF 函数的函数名称，默认为 `myserver`。

index.html

> **注意：**
> 样例默认使用 `es_corpus_0126` 作为索引名，请确保该索引没有业务在使用。如需修改，可在 `index.py` 中修改 `es_index` 变量。

8. 在**触发方式**页单击**添加触发方式**，按下图添加 API 网关触发器，并启用集成响应，然后单击**保存**。

9. 在**触发方式**中查看函数的“访问路径”，单击此路径即可访问页面。



10. 上传 腾讯云 ES 官方文档 样例数据。单击搜索框上方的文字，自动导入数据。

11. 至此，一个简单的基于腾讯云 ES 的问答搜索服务后台已部署完成。

了解更多

停用词和用户词典导入

停用词不会被 ES 检索，用户词典在分词时将保留该词。在上面的案例中，我们导入了默认的 停用词库 和 用户词典，您也可以在 ES 集群详情页的**插件列表 > 更新词典**导入自己的停用词和用户词典。

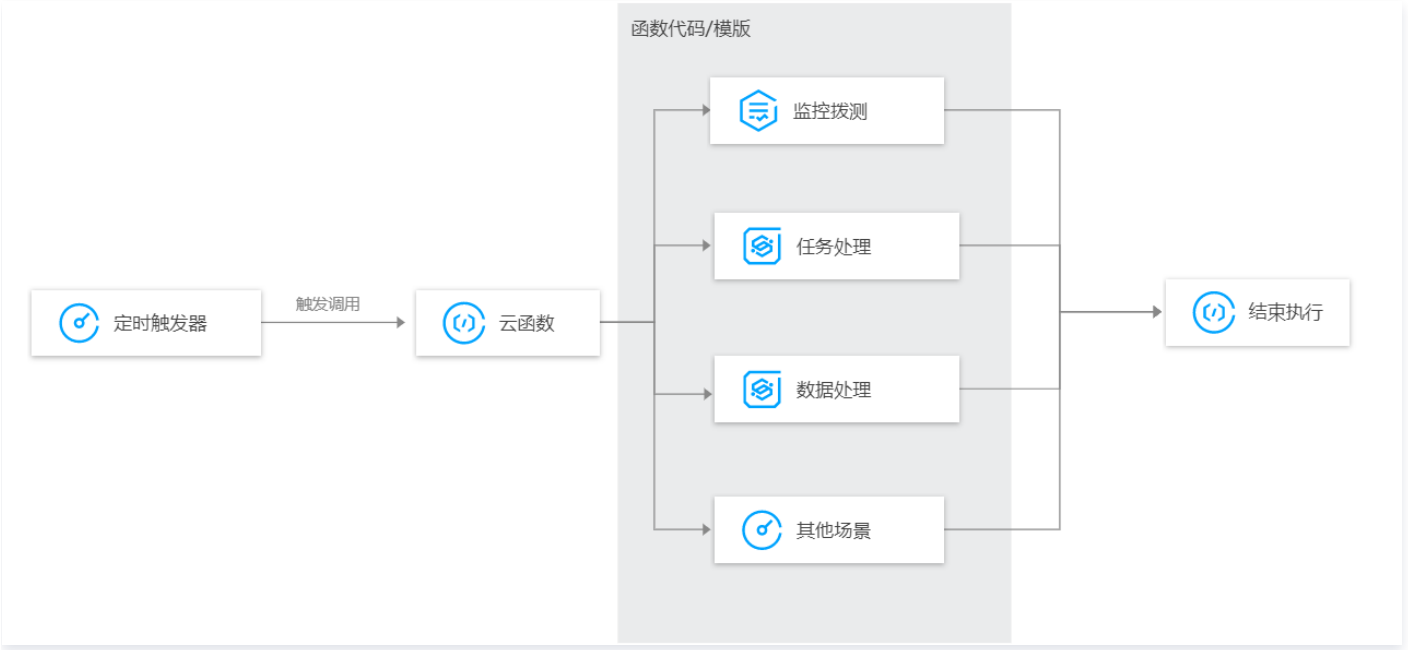
同义词配置

同义词配置需要在创建索引时指定，支持 Solr 和 WordNet 两种同义词格式，详情请参见 Solr synonyms 对格式的介绍。

定时任务
定时任务处理概述

最近更新时间：2022-04-18 10:52:30

通过云函数定时触发器，您可以快速创建定时任务，无需提前购买计算资源。云函数定时触发器依赖于 Serverless 强大的弹性扩缩容能力，可提供稳定快捷的定时任务处理能力。
与其他事件触发器不同，定时触发器 可以使用定时的时间驱动函数执行。设置 Cron 表达式即可快速使用，无需依赖外部调用。对自动监控拨测告警、自动任务执行、数据归档、数据清理、数据备份等典型定时任务场景有天然优势。相关流程如下图所示：



使用函数处理优势

- 提供定时任务全生命周期，帮助用户快速构建定时触发器场景。
- 全托管任务，按照标准 Cron 表达式周期进行触发执行，自动重试。
- 持续增加内置函数模板，降低主流需求下的定时任务开发成本。
- 基于云函数提供计算能力，拥有弹性伸缩、免运维、按需付费等特性。

多场景函数处理实践

已有典型场景模板及具体说明如下表所示：

函数处理场景	描述说明
高可用定时拨测	使用云函数实现高可用定时拨测能力。
定时任务执行	通过 puppeteer 实现定时对页面内容进行采集，数据存储等任务执行。
数据定时归档备份	通过云函数实现数据库定时备份至 COS 能力。

注意
使用定时任务云函数会提供免费额度，免费额度超限后会产生相应的计算费用。计费详情请参见 云函数 SCF 计费概述 。

SCF + 定时任务实现页面内容定时采集

最近更新时间：2023-10-09 11:24:01

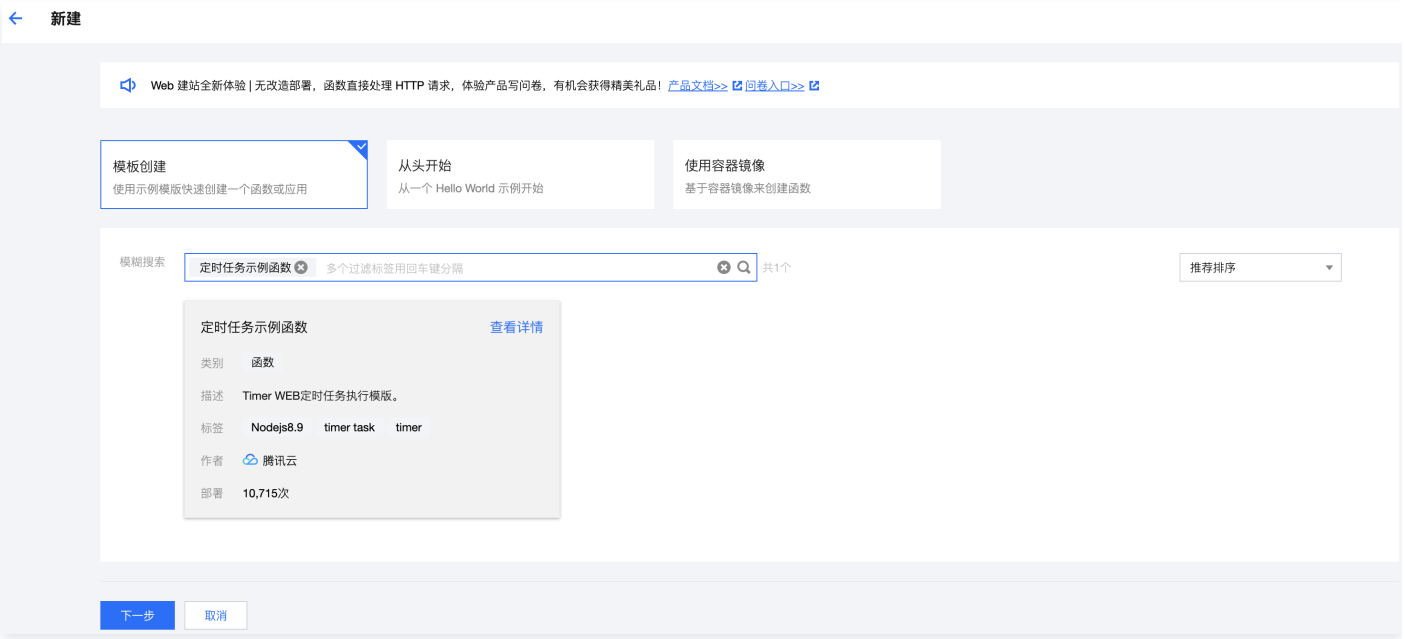
操作场景

本文使用了云函数 SCF，并在函数中通过 puppeteer 实现定时对页面内容进行采集、数据存储等任务。用户还可以通过函数执行数据爬取、定时签到、网页巡检等复杂的 Web 定时任务。

操作步骤

创建云函数

1. 登录 Serverless 控制台，单击左侧导航栏的函数服务。
2. 在**函数服务**页面上方选择**北京**地域，并单击**新建**进入新建函数页面，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**模板创建**。
   - **模糊搜索**：输入“定时任务示例函数”，并进行搜索。单击模板中的**查看详情**，即可在弹窗中查看相关信息，支持下载操作。
3. 单击**下一步**，函数名称默认填充，如需对函数代码进行修改，单击展开**函数代码**卡片并可参见 修改函数模板 进行修改。

4. 在**触发器配置**中，选择**自动创建**，则默认创建一个每1小时0分执行一次的定时触发器。如下图所示：

> **说明**
> - 如需根据需求自行调整触发器配置，请选择**自定义创建**。
> - 如需在测试成功后再创建定时触发器，请选择**暂不创建**。

5. 单击**完成**，完成函数的创建。

测试云函数

1. 在函数代码界面的下方，单击**测试**，查看函数的执行日志。
2. 测试成功后，可以根据实际情况，在**触发方式**页签中配置定时触发器，并验收相关 Base64。

相关操作

修改函数模板

当前模板函数引用 puppeteer 实现对网页内容截屏，并转换为 base64 打印到函数日志。您可以根据自己的定时任务需求对相关模板进行修改。
例如执行以下命令，获取页面 title：

```
// 获取页面title可供参考
const title = await page.title();
console.log(title);
```

增加以下代码，设置点击页面属性。

```
// 点击页面属性可供参考
await page.click('a');
```

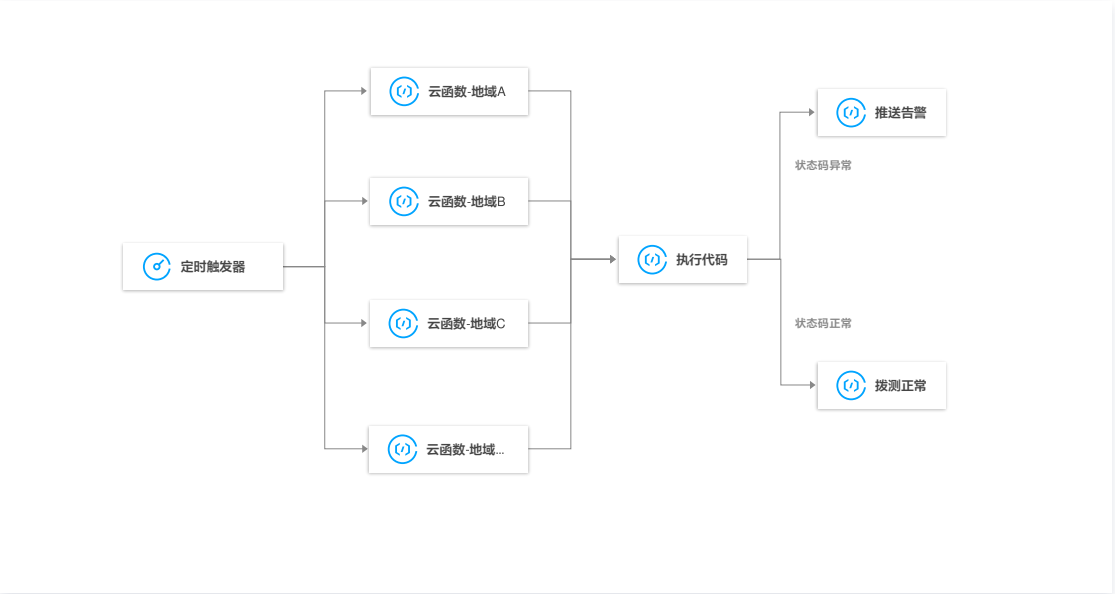
更多 puppeteer 使用指引可参见 puppeteer 文档 。配合该工具可以定时访问页面内容，并对页面进行任务操作，例如数据爬取、签到等。

SCF + 定时任务实现定时拨测并通过邮件发送告警

最近更新时间：2023-09-18 11:12:01

操作场景

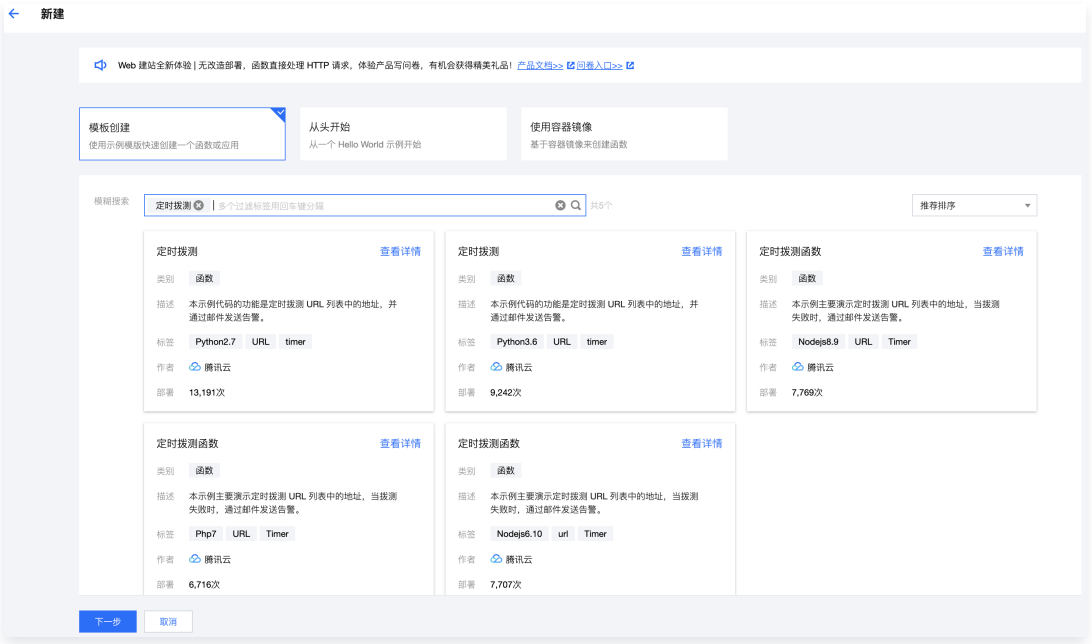
在本示例中，我们用到了云函数 SCF ，并在函数中实现特定 URL 列表的拨测，对测试失败的 URL 发送告警邮件。我们可以给该函数配置一个定时触发器，按照每小时或者每天的频度定时执行。
可以通过部署多地域函数拨测同一 URL 实现高可用的性能拨测任务，相关架构图如下：



操作步骤

创建云函数 PlayCheck

1. 登录 Serverless 控制台，单击左侧导航栏的函数服务。
2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面。
   设置以下参数信息，并单击**下一步**。如下图所示：
   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入“定时拨测”，并进行搜索，本文以运行环境 Python 2.7 为例。
     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。



3. 函数名称默认填充，可根据需要自行修改。
4. 在**函数代码**中，修改代码中的 “第三方 SMTP 服务” 。参数 mail_host、mail_user、mail_pass、mail_port 需要根据实际发送的邮箱或邮件服务器来配置，本示例我们以 QQ 邮箱为例进行说配置。

> **说明**
> 您可以从 QQ邮箱帮助中心 了解到如何开启 QQ 邮箱的 SMTP 功能。QQ 邮箱的 SMTP 功能开启后，相应的参数如下：
> - mail_host：SMTP 服务器地址，即 smtp.qq.com。
> - mail_user：登录用户名为您的邮箱地址，例如 123******@qq.com。
> - mail_pass：您在开启 SMTP 功能时设置的密码。
> - mail_port：服务器登录端口，由于 QQ 邮箱强制要求 SSL 登录，端口固定为465，同时代码中使用 smtplib.SMTP_SSL 创建 SSL 的 SMTP 连接。
> - 修改 “需要拨测的URL地址” 。
> - 修改 “告警邮件通知列表” 。

5. 在 “触发器配置” 中，选择 “自动创建” ，触发器参数保持默认。如需修改请选择 “自定义创建” ，主要参数信息如下：
   - **触发方式**：选择 “定时触发” 。
   - **触发周期**：默认为 “每1小时（每小时0分执行一次）” ，可根据实际情况调整。

测试云函数

在函数代码界面的下方，单击**测试**，查看函数的执行日志，并前往邮箱查看是否收到了告警邮件。

SCF + 定时任务备份数据库到 COS 示例说明

最近更新时间：2022-05-16 17:59:14

实现场景

数据库备份通常是 DBA 每天要进行的工作。对数据库进行备份，可以在数据错误、数据库异常等有需要时及时进行数据回滚。最常用的方式是使用 crontab 定时任务，每日调用备份脚本进行数据库备份。而在备份脚本中，通常最方便使用的是 mysqldump 工具，导出表结构及表数据。
接下来，我们将利用云函数，实现数据库备份能力，然后通过配置定时触发器，确保备份函数可以按需每天、或按指定间隔时间运行。

函数实现概要

> **说明**
> 本实操教程假设以下情况：
> - 您需要对数据库定期的导出备份。
> - 您希望把备份结果放到 COS 以便于随时下载使用。

1. 创建 COS Bucket 用于接收备份的数据库。
2. 创建及部署云函数。
3. 根据需要配置定时触发器并进行测试。

步骤 1. 创建及部署云函数

最近更新时间：2023-07-31 15:00:33

前提条件

- 请参考 创建存储桶 来创建一个 Bucket，命名为 `mysql-backup`，并选择**北京**地域，权限选择**私有读写**。
- 请参考 初始化 MySQL 数据库，创建并初始化一个 MySQL 数据库。

> **说明**
> 本文使用 MySQL 数据库，您可根据实际需求使用其他数据库。

创建云函数

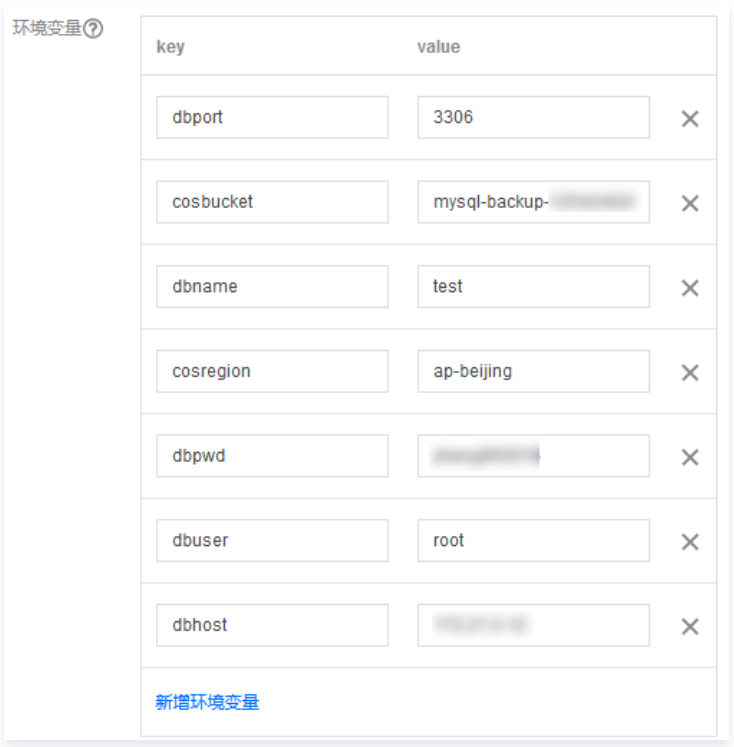
1. 登录 Serverless 控制台，单击左侧导航栏的函数服务。
2. 在“函数服务”页面上方选择**北京**地域，并单击**新建**进入新建函数页面，根据页面相关信息进行配置，如下图所示：

- **创建方式**：选择**模板创建**。
- **模糊搜索**：输入 “MySQL自动备份”，并进行搜索。
  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。

> **注意**
> 如果是 Windows 电脑，则下载到本地的代码会失去 mysqldump 工具的可执行权限。可以将项目放置到 Linux 或 MacOS 环境下，并在项目目录下执行 `chmod +x mysqldump` 命令，为 mysqldump 工具附加上可执行权限。

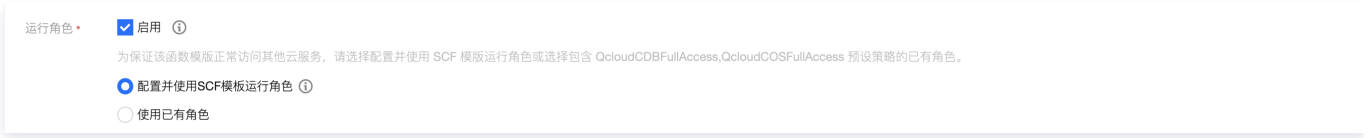
3. 单击**下一步**，函数名称默认填充，可根据需要自行修改。按照引导配置环境变量、运行角色和私有网络：

- **环境变量**：环境变量参考表格进行填写。如下图所示：



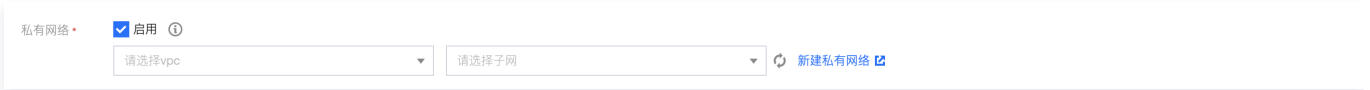
key	value
dbhost	请参考 访问 MySQL 数据库 获取。
dbport	
dbuser	新创建的 MySQL 数据库的用户名默认为 root。
dbname	需备份的数据库名称，本文以 test 为例。
dbpwd	已设置的 root 账号密码。
cosregion	Bucket 所在地域。
cosbucket	已创建的 Bucket 名称，本文中为 mysql-backup-您的 APPID，APPID 请前往 账号信息 获取。

- **运行角色**：勾选**启用**，本文以“配置并使用SCF模板运行角色”为例。详细说明如下：
- **配置并使用SCF模板运行角色**：选择该项将会自动创建并选择关联了 COS、CDB 全读写权限的 SCF 模板运行角色。
- **使用已有角色**：需在下拉列表中选择包含上述权限的已有角色。



> **说明**
> 云函数在运行时，会使用运行角色换取临时密钥，操作相关云产品资源。

- **私有网络**：若数据库使用的是内网地址，则需要启用私有网络，并选择和数据库相同的 VPC 和子网。如下图所示：



4. 在**触发器配置**中，选择**自动创建**，可自动创建一个每小时0分触发一次的定时触发器，实现定时备份数据库到 COS。如暂时无需创建触发器，请选择**暂不创建**。
5. 单击**完成**，完成函数和定时触发器创建。

步骤 2. 测试及启动云函数

最近更新时间：2023-08-21 09:41:22

前提条件

完成函数创建后，我们就可以测试函数的运行情况，并完成最终的定时触发配置。

> **注意**
> dump 目录只能设置为 `/tmp` ，因为运行环境内除 `/tmp` 目录下，其他均限制为只读。

测试及启动云函数

- 通过单击控制台下方的**测试**按键，可以直接触发函数运行。
- 通过函数的输出日志，可以查看代码的运行情况，检查 dump 文件是否生成正常，是否成功上传到 COS 存储桶中。
- 同时也可以到对应的备份存储桶中，查看生成的文件，检查是否数据正确、备份正常。

确认函数测试运行正常后，可在触发器中为函数新增一个定时触发器。您可以根据自身需要，配置为每天，或每12小时，或每月的指定时间运行。定时触发器可以按 crontab 格式编写触发时间，既可以按一定时间周期，也可以指定具体时间运行。

总结

在本实操示例教程中，仅为您提供了一种实现的参考思路。我们通过使用 mysqldump 工具，以及对象存储 COS 的 SDK，实现了数据库的按时备份能力。通过使用云函数，我们无需使用云服务器以及配置 crontab 脚本，就可以实现高可靠的定时运维能力。云函数，以及云函数的定时触发器，在运维过程中可以被广泛使用，实现例如备份、检查、告警、同步等各种能力。

视频处理
SCF + FFmpeg 实现批量自动视频剪辑

最近更新时间：2022-04-18 10:53:08

操作场景

常用的视频剪辑工具 Premiere、AE 等可以实现复杂的剪辑效果，在宣传视频制作、广告视频制作中被广泛应用。但在以下场景中，传统的视频剪辑工具或者模板化的视频处理软件无法满足**批量、自动化和可定制**的视频剪辑需求：

- 学校期望能在学生上完网课之后马上呈现所有学生学习过程中的精彩视频，搭配学校的 logo 和宣传语等，支持学生一键分享自己的成果。假设有 1 万个学生，则需要为每个学生制作唯一的视频，所以需要批量且自动化的完成 1 万个不同的视频剪辑。
- 某次营销活动中，需要为不同的用户生成不同的头像视频来吸引用户参与。每个用户的头像都不同，生成的视频也不同，用户可能成千上万，所以需要自动化完成。
- 网红运营公司期望能给所有主播生成统一的营业视频。可能有 100 个主播，专门找一个人剪辑 100 个视频好像勉强能接受，但如果每周都要剪一次不同的视频呢？所以需要自动化、批量和可定制化的剪辑。

这类视频剪辑场景还具有使用时段集中、计算量大的特点。单独购买高规格的服务器利用率很低，买低规格的服务器计算能力无法满足要求。Serverless 按量计费的特点，以及高性能的计算能力，完美匹配了这样的需求场景。既能达到 100% 的利用率，又能按量使用高性能计算能力。同时，Serverless 支持丰富的可编程环境，可支持不同开发习惯的开发者，灵活性更高。

前提条件

本文提到的所有视频剪辑的功能，均使用 FFmpeg 工具完成。

FFmpeg 使用方法

FFmpeg 是一个用来做视频处理的开源工具，支持视频剪辑、视频转码、视频编辑、音频处理、添加文字、视频拼接、拉流推流直播等功能。通过不同的 FFmpeg 命令可以编程完成不同的视频剪辑功能，组合编排起来，即可应对各种批量自动化的场景。
常见的视频剪辑场景主要包含以下几种：

1. 视频转码
2. 视频裁剪
3. 视频加文字
4. 视频加图片
5. 视频拼接
6. 视频加音频
7. 视频转场
8. 视频特效
9. 视频加速慢速播放

FFmpeg 命令

下文介绍了一些具体的 FFmpeg 命令，您可以本地 安装 FFmpeg 后进行测试。

```
// 将 MOV 视频转成 mp4 视频
ffmpeg -i input.mov output.mp4
```

```
// 将原视频的帧率修改为 24
ffmpeg -i input.mp4 -r 24 -an output.mp4
```

```
// 将 mp4 视频转为可用于直播的视频流
ffmpeg -i input.mp4 -codec: copy -bsf:v h264_mp4toannexb -start_number 0 -hls_time 10 -hls_list_size 0 -f
hls output.m3u8
```

```
// 将视频分别变为 480x360，并把码率改 400
```

```
ffmpeg -i input.mp4 -vf scale=480:360,pad=480:360:240:240:black -c:v libx264 -x264-params nal-
hrd=cbr:force-cfr=1 -b:v 400000 -bufsize 400000 -minrate 400000 -maxrate 400000 output.mp4
```

```
// 给视频添加文字，例如字幕、标题等。
// `fontfile`是要使用的字体的路径，`text`是您要添加的文字，
// `fontcolor`是文字的颜色，`fontsize`是文字大小，`box`是给文字添加底框。
// `box=1`表示 enable，`0`表示 disable，`boxcolor`是底框的颜色，black@0.5 表示黑色透明度是 50%，`boxborderw`是底框
距文字的宽度
// `x`和`y`是文字的位置，`x`和`y`不只支持数字，还支持各种表达式，具体可以去官网查看
ffmpeg -i input.mp4 -vf "drawtext=fontfile=/path/to/font.ttf:text='您的文
字':fontcolor=white:fontsize=24:box=1:boxcolor=black@0.5:boxborderw=5:x=(w-text_w)/2:y=(h-text_h)/2" -
codec:a copy output.mp4
```

```
// 给视频添加图片，例如添加 logo、头像、表情等。filter_complex 表示复合的滤镜，overlay 表示表示图片的 x 和 y，enable 表
示图片出现的时间段，从 0-20 秒
ffmpeg -i input.mp4 -i avatar.JPG -filter_complex "[0:v][1:v] overlay=25:25:enable='between(t,0,20)'" -
pix_fmt yuv420p -c:a copy output.mp4
```

```
// 视频拼接，list.txt 里面按顺序放所有要拼接的视频的文件路径，如下。
// 注意，如果视频的分辨率不一致会导致拼接失败。
ffmpeg -f concat -safe 0 -i list.txt -c copy -movflags +faststart output.mp4
// list.txt 的格式如下
file 'xx.mp4'
file 'yy.mp4'
```

```
// 视频加音频，stream_loop 表示是否循环音频内容，-1 表示无限循环，0 表示不循环。shortest 表示最短的 MP3 输入流结束时完成
编码。
ffmpeg -y -i input.mp4 -stream_loop -1 -i audio.mp3 -map 0:v -map 1:a -c:v copy -shortest output.mp4
```

说明

FFmpeg 也支持单独对音频进行编辑，具体使用方法可参考 FFmpeg 官网。

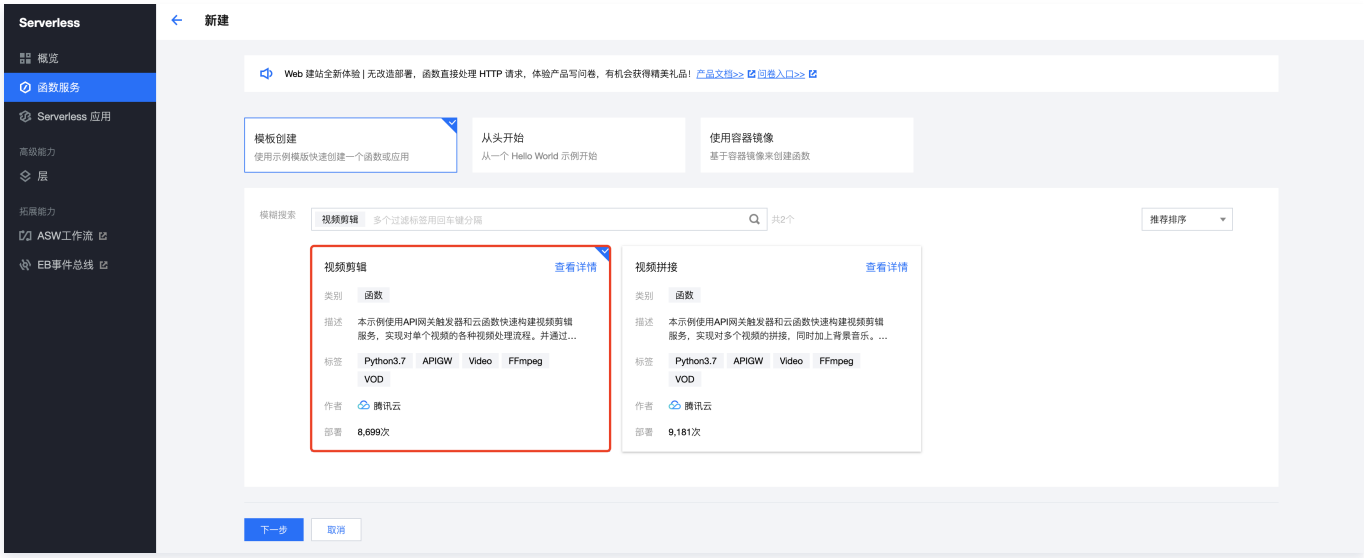
执行 FFmpeg 命令

本文以 Python 为例，可以参考以下代码示例执行 FFmpeg 命令。

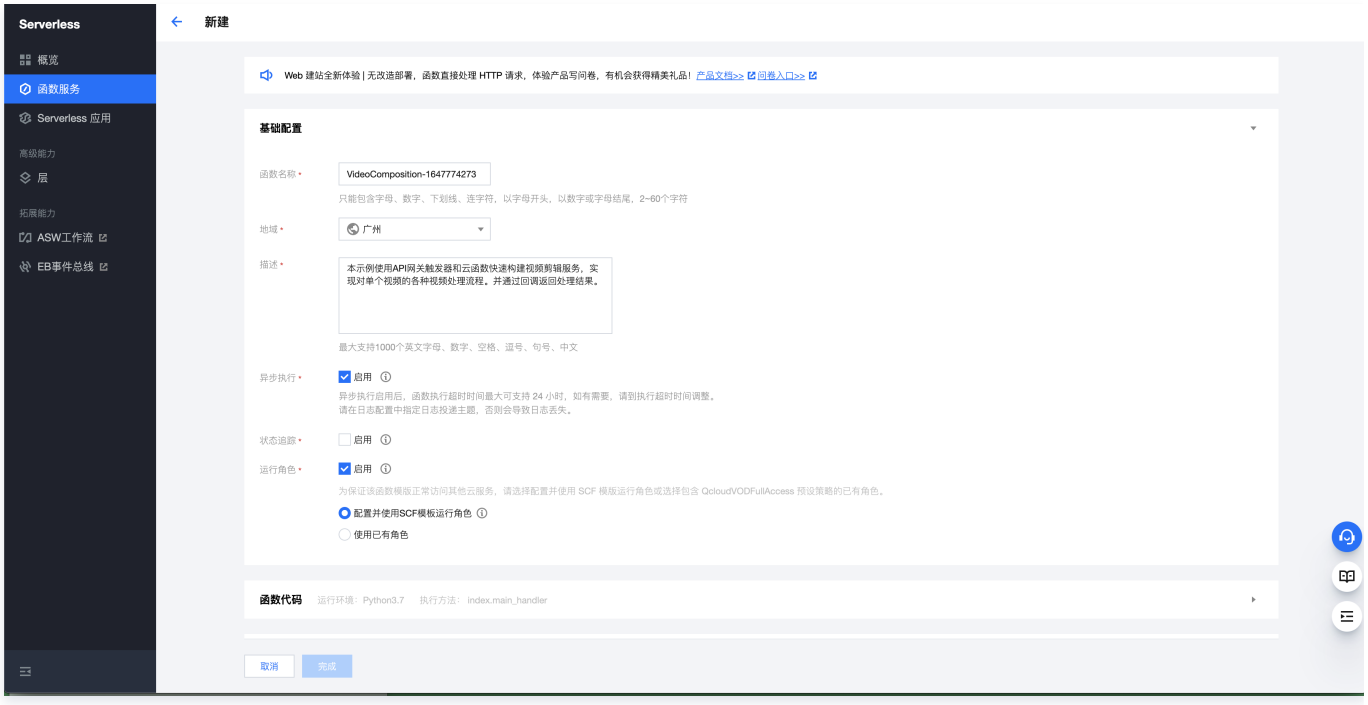
```
child = subprocess.run('./ffmpeg -i input.mov output.mp4',
                       stdout=subprocess.PIPE,
                       stderr=subprocess.PIPE, close_fds=True, shell=True)
if child.returncode == 0:
    print("success:", child)
else:
    print("error:", child)
        raise KeyError("处理视频失败，错误：", child)
```

在 SCF 上使用 FFmpeg

1. 登录 Serverless 控制台，单击左侧导航栏的函数服务。
2. 在“函数服务”页面上方选择地域，并单击**新建**进入新建函数页面。
3. 设置以下参数信息，并单击**下一步**，如下图所示：
   - **创建方式**：选择**模板创建**。
   - **模糊搜索**：输入“视频剪辑”，并进行搜索。
     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。单击 github 地址可以访问模板源码，模板源码中可查看此 API 的调用参数和使用方式。



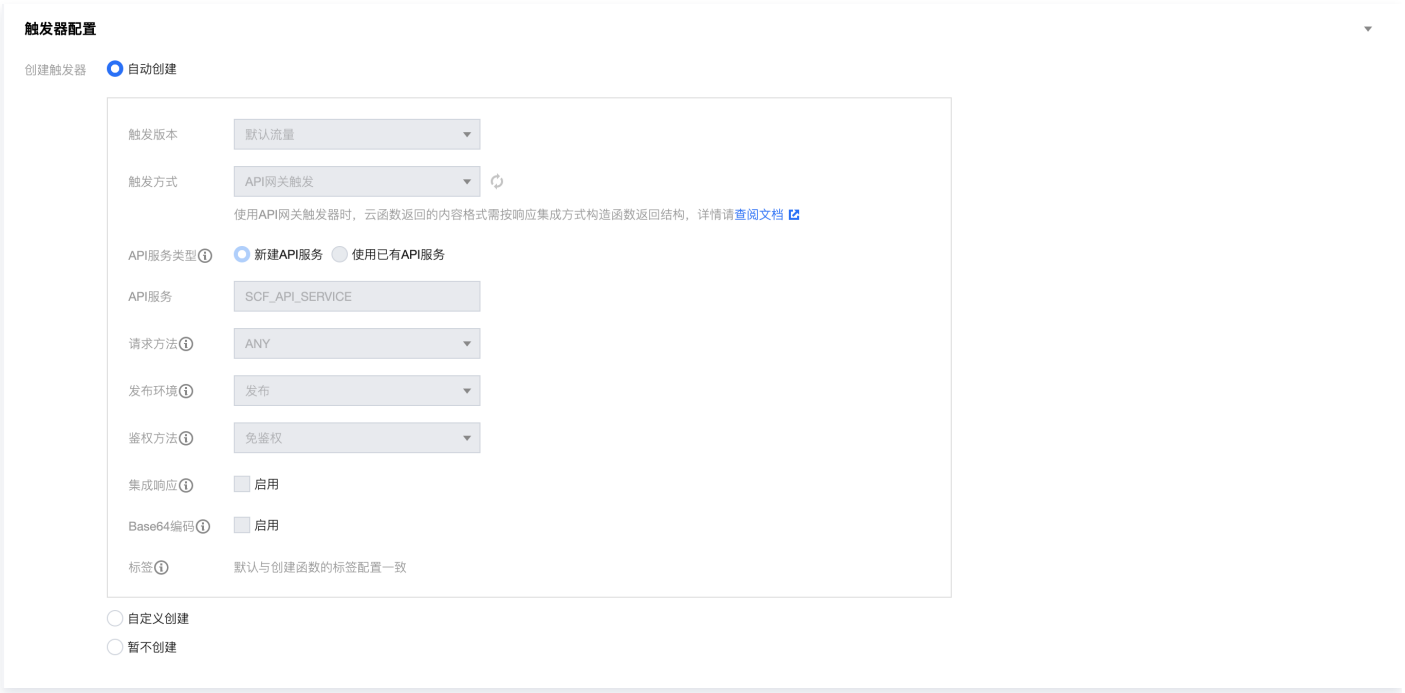
4. 在**基础配置**中，默认生成**函数名称**，可根据使用需求自行修改。按照引导配置异步执行和运行角色：
   - 异步执行：勾选“启用”
   - 运行角色：勾选“启用”，选择“配置并使用 SCF 模板运行角色”，将会自动创建并选择关联了 VOD 全读写权限的 SCF 模板运行角色，或选择“使用已有角色”，在下拉列表中选择包含上述权限的已有角色。本文以“配置并使用 SCF 模板运行角色”为例。如下图所示：



> **注意**
> 示例需要授权 SCF 操作 VOD 的权限，已默认勾选“运行角色”并自动完成函数运行角色的创建和所需 VOD 操作权限策略 QcloudVODFullAccess 的关联，如需调整，请选择“使用已有策略”或取消“运行角色”勾选。

5. 在**触发器配置**中，选择“自动创建”，如下图所示：

> **注意**
> 如需使用已有 API 服务创建 API 网关触发器或修改触发器配置，请选择“自定义创建”。



6. 单击**完成**，即可完成函数和触发器创建并获得该函数的 HTTP 触发域名。

落地案例

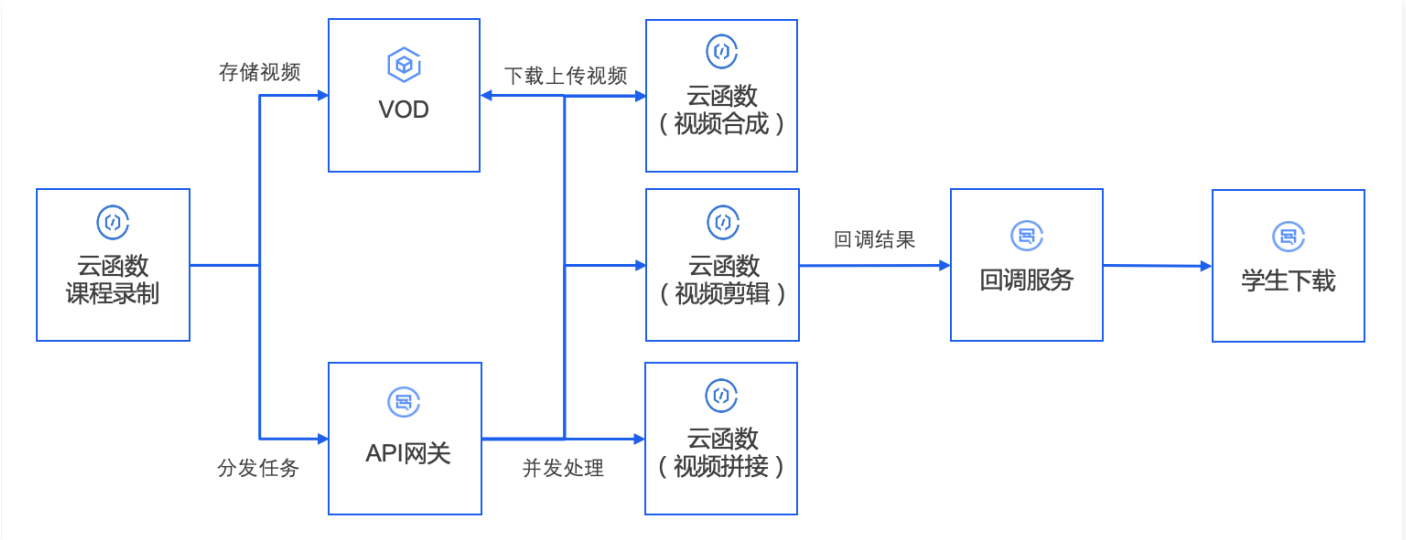
某在线教育企业，需要在每次学生上完网课之后把网课录像制作成一段 30 秒的视频，作为学生的学习成果。此案例有几个关键的信息：

1. 通常一堂课有 200 个学生，需要同时制作 200 个视频。
2. 需要把 1 小时的上课视频剪辑成 30 秒。
3. 由于每个学生的上课屏幕有所不同，因此录制的视频都是不同的。
4. 最终的成果视频还需要加上学生的名字和头像。
5. 学生结束上课的时间很集中，因此制作视频时会有短时高并发。
6. 每次上完课的时候才会需要制作视频，时段比较固定且集中。

综合上述特点，用 Serverless 云函数来做这样的视频剪辑具有多个优势：

1. 解决了 200 个并发的问题，不需要自行搭建过多服务器。
2. 解决了只在发生时段使用的问题，其他时段都没有成本产生。
3. 解决了需要较强计算能力快速制作视频的问题。

案例的参考架构图如下所示：



总结

通过编排、组合、复用上面列举的各种音视频剪辑的场景，即可满足多种场景诉求。将视频剪辑中用来控制各种效果的参数，转成调用服务时传入的参数，即可实现各种效果的定制化。