

02 用户指南

文档版本 01
发布日期 2025-02-28



版权所有 © 华为云计算技术有限公司 2025。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址：贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址：<https://www.huaweicloud.com/>

目录

1 FunctionGraph 临时入库 AK/SK 替换指导.....	1
1.1 使用永久 AK/SK 替换.....	1
1.2 使用 STS Token 替换.....	2
2 使用前必读.....	4
2.1 FunctionGraph 使用流程.....	4
2.2 FunctionGraph 权限说明.....	7
2.2.1 创建用户并授权使用 FunctionGraph.....	7
2.2.2 FunctionGraph 自定义策略.....	9
2.2.3 FunctionGraph 资源.....	10
2.3 支持的编程语言.....	11
2.3.1 Node.js 语言.....	11
2.3.2 Python 语言.....	11
2.3.3 Java 语言.....	12
2.3.4 Go 语言.....	12
2.3.5 C#语言.....	12
2.3.6 PHP 语言.....	12
2.3.7 定制运行时语言.....	13
2.3.8 仓颉语言.....	19
3 构建函数.....	20
3.1 创建程序包.....	20
3.2 使用空白模板创建函数.....	26
3.2.1 创建事件函数.....	26
3.2.2 创建 HTTP 函数.....	29
3.3 使用示例模板创建函数.....	35
3.4 使用容器镜像部署函数.....	36
3.5 使用 Terraform 部署函数.....	40
3.5.1 概述.....	40
3.5.2 前置条件.....	40
3.5.2.1 获取访问密钥.....	41
3.5.2.2 准备 Terraform 环境.....	42
3.5.3 基础 Terraform 语法.....	43
3.5.4 编写函数资源脚本.....	43

3.5.5 使用 Terraform 命令创建函数.....	43
4 配置函数.....	45
4.1 配置初始化.....	45
4.2 配置常规信息.....	46
4.3 配置委托权限.....	48
4.4 配置网络.....	53
4.5 配置磁盘挂载.....	56
4.6 配置环境变量.....	66
4.7 配置函数异步.....	69
4.8 配置单实例多并发.....	73
4.9 版本管理.....	76
4.10 别名管理.....	78
4.11 配置动态内存.....	80
4.12 配置心跳函数.....	82
4.13 配置标签.....	83
4.14 配置快照式冷启动.....	84
4.15 配置日志组及日志流.....	88
4.16 有状态函数.....	89
4.16.1 简介.....	89
4.16.2 开发指导.....	90
4.16.3 注意事项.....	91
4.17 共享 VPC.....	91
5 在线调试.....	93
6 配置触发器.....	98
6.1 触发器管理.....	98
6.2 使用定时触发器.....	99
6.3 使用 APIG (专享版) 触发器.....	100
6.4 使用 Kafka 触发器.....	102
6.5 使用 DIS 触发器.....	104
6.6 使用 SMN 触发器.....	107
6.7 使用 LTS 触发器.....	108
6.8 使用 CTS 触发器.....	110
6.9 使用 DDS 触发器.....	112
6.10 使用 GeminiDB Mongo 触发器.....	113
6.11 使用 APIG 触发器.....	115
6.12 使用 APIC 触发器.....	117
6.13 使用 RabbitMQ 触发器.....	119
6.14 使用开源 Kafka 触发器.....	121
6.15 函数定时触发器 Cron 表达式规则.....	123
6.16 使用 EG 触发器.....	126
6.16.1 创建 EG 触发器 (RocketMQ 自定义事件源).....	126

6.16.2 创建 EG 触发器（ OBS 应用事件源 ）	128
6.16.3 创建 EG 触发器（ 云服务事件源 ）	129
7 调用函数.....	131
7.1 同步调用.....	131
7.2 异步调用.....	131
7.3 重试机制.....	133
8 监控.....	134
8.1 指标.....	134
8.1.1 监控信息说明.....	134
8.1.2 FunctionGraph 服务的监控指标参考.....	135
8.1.3 创建告警规则.....	138
8.2 日志.....	142
8.2.1 查看函数日志.....	142
8.2.2 管理函数日志.....	143
8.3 调用链管理.....	147
9 函数管理.....	151
10 依赖包管理.....	153
10.1 配置依赖包.....	153
10.2 引入依赖库.....	155
10.3 公共依赖.....	158
10.3.1 什么是公共依赖.....	158
10.4 公共依赖 Demo.....	158
10.4.1 使用 TensorFlow 进行线性回归.....	159
10.4.2 使用 pytorch 进行线性回归.....	160
10.4.3 sklearn.....	161
10.4.4 gym.....	161
11 预留实例管理（旧）	162
12 预留实例管理.....	166
13 函数流管理.....	175
13.1 函数流简介.....	175
13.2 创建函数流任务.....	194
13.3 函数流执行历史管理.....	204
13.4 创建函数流触发器.....	207
13.5 流式文件处理.....	212
14 扩大资源配额.....	215
15 GPU 函数管理.....	217
15.1 Serverless GPU 使用介绍.....	217
15.1.1 概述.....	217
15.1.2 应用场景.....	218

15.1.2.1 准实时推理场景.....	218
15.1.2.2 实时推理场景.....	219
15.1.2.3 离线异步任务场景.....	220
15.2 部署方式.....	220
15.2.1 自定义镜像方式部署.....	220
15.2.2 定制运行时方式部署.....	221
15.3 函数模式.....	221
16 应用中心.....	222
17 共享.....	227
18 可编程 CDN 函数.....	230
18.1 创建 CDN 函数.....	230
18.2 CDN 函数管理.....	232
19 CLI 命令参考.....	236
19.1 KooCLI 工具介绍.....	236
19.2 安装 KooCLI 命令行工具.....	236
19.3 函数调用.....	238
20 审计.....	241
20.1 云审计服务支持的 FunctionGraph 操作列表.....	241
20.2 在 CTS 事件列表查看云审计事件.....	241

1 FunctionGraph 临时入库 AK/SK 替换指导

1.1 使用永久 AK/SK 替换

1. 登录[函数 workflow 控制台](#)，右上角选择“我的凭证”。

图 1-1 我的凭证



2. 选择“访问密钥”，单击“新增访问密钥”并获取访问密钥，具体详情请参见[管理IAM用户访问密钥](#)。

图 1-2 新增访问密钥



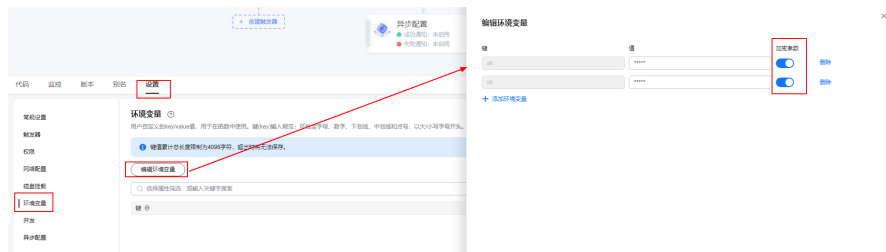
3. 返回函数 workflow 控制台，选择“函数 > 函数列表”，单击函数名称进入详情页。

图 1-3 进入函数详情页



4. 选择“设置 > 环境变量”，单击“编辑环境变量”，新增AK/SK环境变量，并开启“加密参数”，配置完成后，单击“确定”。

图 1-4 新增 AK/SK 环境变量



5. 选择“代码”页签，将原代码中“context.getAccessKey()”和“context.getSecretKey()”分别替换为“context.getUserData("ak")”和“context.getUserData("sk”)”。（当前以python语言为例，其他语言请根据实际情况改写）

图 1-5 修改 ak/sk 获取代码



6. 代码修改完成后，单击“部署”刷新代码。

1.2 使用 STS Token 替换

1. 登录函数 workflow 控制台，选择“函数 > 函数列表”，单击函数名称进入详情页。

图 1-6 进入函数详情页



- 2. 选择“代码”页签，将原代码中“context.getAccessKey()”和“context.getSecretKey()”替换为“context.getSecurityAccessKey()”、“context.getSecuritySecretKey()”和“context.getSecurityToken()”。（当前以python语言为例，其他语言请参见[更多语言使用参考](#)）

图 1-7 修改代码

```
107 def new_obs_client(context, obsServer):
108     ak = context.getAccessKey()
109     sk = context.getSecretKey()
110     return ObsClient(access_key_id=ak, secret_access_key=sk, server=obsServer)
111
```

↓

```
107 def new_obs_client(context, obsServer):
108     ak = context.getSecurityAccessKey()
109     sk = context.getSecuritySecretKey()
110     st = context.getSecurityToken()
111     return ObsClient(access_key_id=ak, secret_access_key=sk, security_token=st, server=obsServer)
112
```

说明

STS Token 必须将 securityAccessKey、securitySecretKey、securityToken 三个参数一起使用。

更多语言使用参考

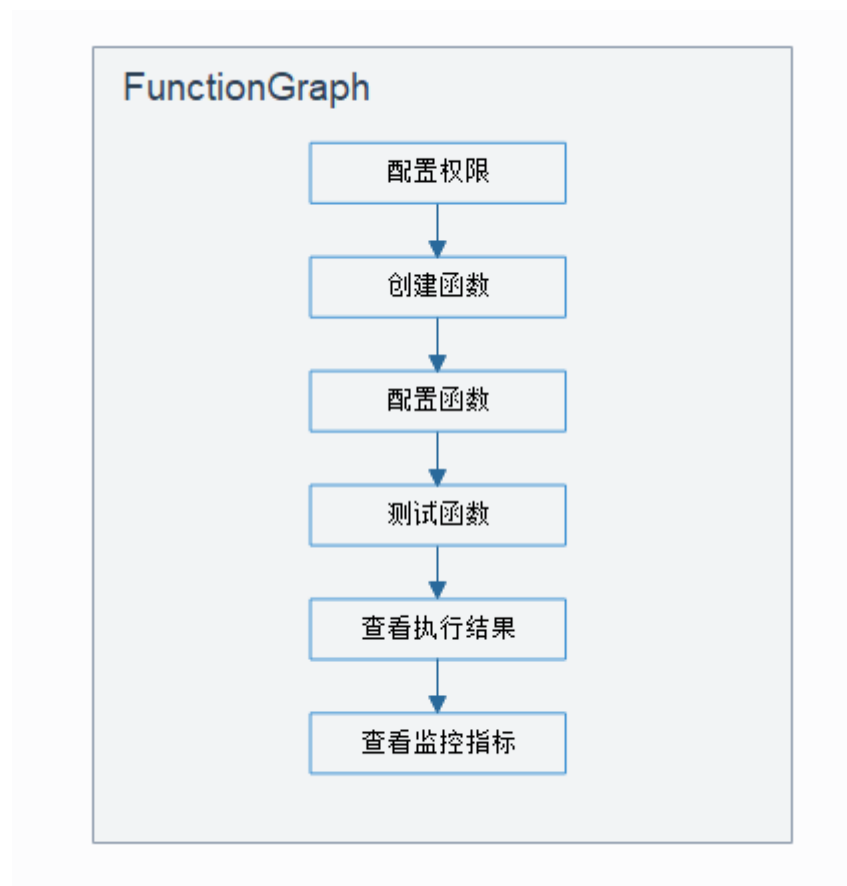
[Java](#)、[Python](#)、[Go](#)、[Nodejs](#)、[.Net](#)、[Php](#)。

2 使用前必读

2.1 FunctionGraph 使用流程

函数工作流FunctionGraph是一项基于事件驱动的功能托管计算服务。使用FunctionGraph函数，只需编写业务函数代码并设置运行的条件，无需配置和管理服务器等基础设施，函数以弹性、免运维、高可靠的方式运行。此外，按函数实际执行资源计费，不执行不产生费用。

使用FunctionGraph快速创建函数的流程如下：



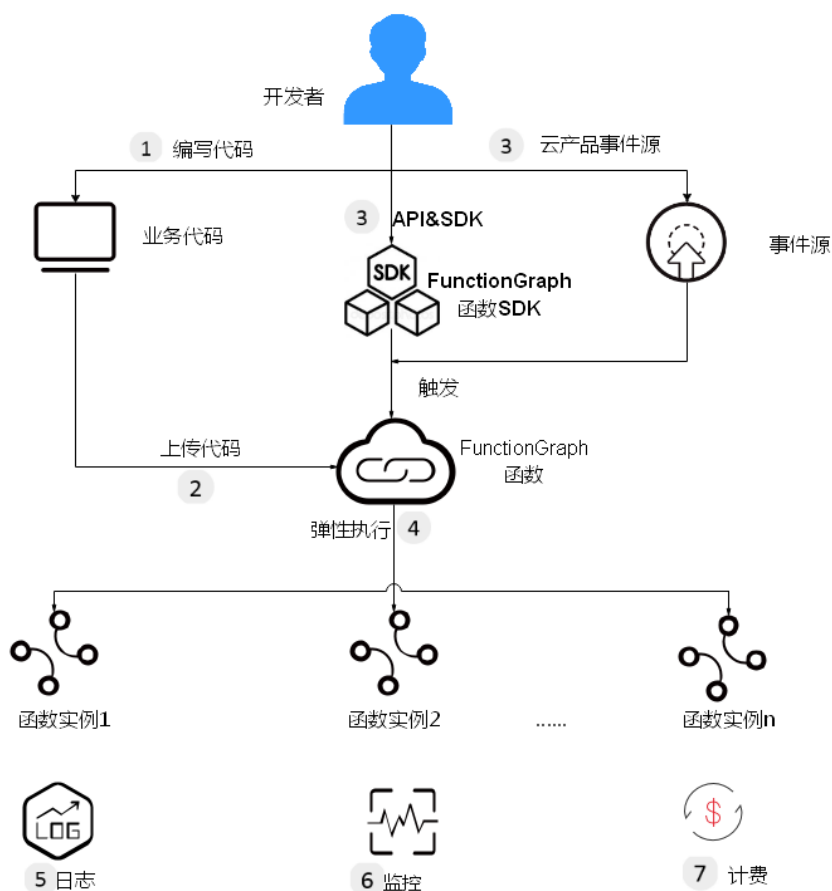
1. 配置权限：确保登录的用户已有“FunctionGraph Administrator”权限。
2. 创建函数：选择使用空白模板创建函数、示例代码创建函数、容器镜像部署函数。
3. 配置函数：配置代码源或修改其他参数配置。
4. 测试函数：创建测试事件来调试函数。
5. 查看执行结果：在函数详情页面，根据配置的测试事件，查看执行结果。
6. 查看监控指标：在函数详情页面的“监控”页签，查看函数监控指标。

函数使用流程

函数使用流程如图2-1所示。

1. 用户编写业务程序代码，打包上传至FunctionGraph函数，添加事件源（如SMN、APIG等），完成应用程序构建部署。
2. 通过RESTful API或者云产品事件源触发函数，生成函数实例，实现业务功能，函数在运行过程中的资源调度由FunctionGraph来管理。
3. 用户可以查看函数运行日志和监控信息，按照代码运行情况收费，代码未运行时不产生费用。

图 2-1 函数使用流程



说明如下：

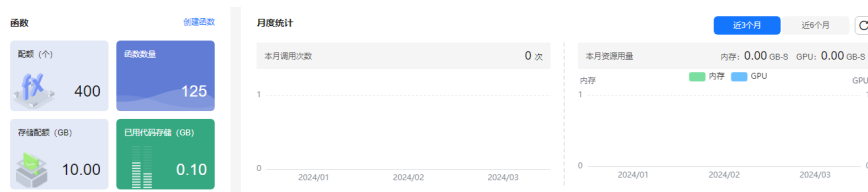
1. 编写代码
用户编写代码，目前支持Node.js、Python、Java、Go、C#、PHP等语言，详情请参见[开发指南](#)。
2. 上传代码
上传代码，目前支持在线编辑、上传ZIP或JAR包，从OBS引用ZIP包等，详情请参见[创建程序包](#)。
3. API和云产品事件源触发函数执行
通过API和云产品事件源触发函数执行，触发方法请参见[配置触发器](#)。
4. 弹性执行
函数在执行过程中，会根据请求量弹性扩容，支持请求峰值的执行，此过程用户无需配置，由FunctionGraph完成，并发数限制请参见[使用限制](#)。
5. 查看日志
FunctionGraph函数实现了与云日志服务的对接，您无需配置，即可查看函数运行日志信息，请参见[日志](#)。
6. 查看监控
FunctionGraph函数实现了与云监控服务的对接，您无需配置，即可查看图形化监控信息，请参见[指标](#)。
7. 计费方式
函数执行结束后，根据函数请求执行次数和执行时间计费，查看费用详情请参见[费用账单](#)。

总览页面介绍

登录FunctionGraph控制台，在左侧导航栏选择“总览”，进入“总览”页面。

- 可以查看函数数量/配额信息、代码存储/存储配额、函数月度调用次数/月度资源用量。

图 2-2 月度统计



- 可以查看租户层面的监控信息：调用次数、调用数TOP10、错误次数、错误数TOP10、运行时间、被拒绝次数。

运行监控指标说明如[表2-1](#)所示。

表 2-1 监控指标说明表

指标	单位	说明
调用次数	次	函数总的调用请求数，包含了错误和被拒绝的调用。异步调用在该请求实际被系统执行时才开始计数。

指标	单位	说明
调用数TOP10	-	展示指定时间范围内（最近1天/最近3天/自定义）的函数调用数TOP10。
运行时间	毫秒	最大运行时间为某统计粒度（周期）下，即某一时间段内所有函数单次执行最大的运行时间。 最小运行时间为某统计粒度（周期）下，即某一时间段内所有函数单次执行最小的运行时间。 平均运行时间为某统计粒度（周期）下，即某一时间段内所有函数单次执行平均的运行时间。
错误次数	次	指发生异常请求的函数不能正确执行完并且返回200，都计入错误次数。函数自身的语法错误或自身执行错误也会计入该指标。
错误数TOP10	-	展示指定时间范围内（最近1天/最近3天/自定义）的函数错误数TOP10。
被拒绝次数	次	由于并发请求太多，系统流控而被拒绝的请求次数。

- 可以查看函数流指标：调用次数、运行时间、错误次数、运行中

指标	单位	说明
调用次数	次	函数流总的调用请求数，包含了正确、错误和运行中的调用。异步函数流在请求被系统执行时才开始计数。
运行时间	毫秒	时间段内单次函数流执行平均的运行时间。
错误次数	次	指发生异常请求的函数流不能正确执行完，会计入错误次数。
运行中	个	正在运行中的函数流的数量。

2.2 FunctionGraph 权限说明

2.2.1 创建用户并授权使用 FunctionGraph

如果您需要对您所拥有的FunctionGraph进行精细的权限管理，您可以使用[统一身份认证服务](#)（Identity and Access Management，简称IAM），通过IAM，您可以：

- 根据企业的业务组织，在您的华为云账号中，给企业中不同职能部门的员工创建IAM用户，让员工拥有唯一安全凭证，并使用FunctionGraph资源。
- 根据企业用户的职能，设置不同的访问权限，以达到用户之间的权限隔离。
- 将FunctionGraph资源委托给更专业、高效的其他华为云账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

如果华为云账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用FunctionGraph服务的其它功能。

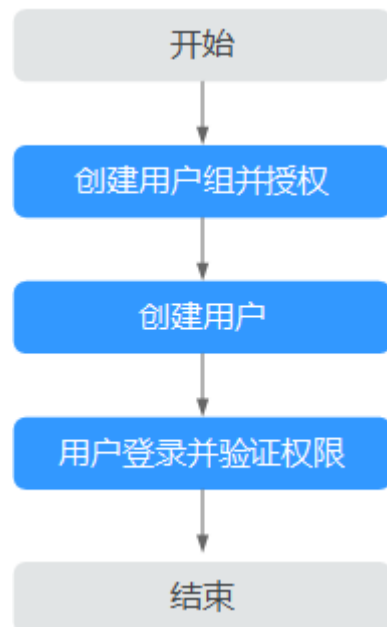
本章节为您介绍对用户授权的方法，操作流程如图2-3所示。

前提条件

给用户组授权之前，请您了解用户组可以添加的FunctionGraph权限，并结合实际需求进行选择，FunctionGraph支持的系统权限，请参见[FunctionGraph系统策略](#)。若您需要对除FunctionGraph之外的其它服务授权，IAM支持服务的所有权限请参见[权限策略](#)。

示例流程

图 2-3 给用户授权使用 FunctionGraph 权限的流程



1. 创建用户组并授权

在IAM控制台创建用户组，并授予FunctionGraph查询及调用权限“FunctionGraph Invoker”。

2. 创建用户并加入用户组

在IAM控制台创建用户，并将其加入1中创建的用户组。

3. 用户登录验证权限

新创建的用户登录管理控制台，验证FunctionGraph的函数查询权限。

- 在“服务列表”中选择“函数工作流 FunctionGraph”，进入“函数 > 函数列表”，单击“创建函数”进入到创建函数界面，发现无法创建函数，表示“FunctionGraph Invoker”已生效。
- 在“服务列表”中选择除FunctionGraph外的任一服务，若提示权限不足，表示“FunctionGraph Invoker”已生效。

2.2.2 FunctionGraph 自定义策略

如果系统预置的FunctionGraph权限，不满足您的授权要求，可以创建自定义策略。

目前华为云支持以下两种方式创建自定义策略：

- 可视化视图创建自定义策略：无需了解策略语法，按可视化视图导航栏选择云服务、操作、资源、条件等策略内容，可自动生成策略。
- JSON视图创建自定义策略：可以在选择策略模板后，根据具体需求编辑策略内容；也可以直接在编辑框内编写JSON格式的策略内容。

具体创建步骤请参见：[创建自定义策略](#)。本章为您介绍常用的FunctionGraph自定义策略样例。

FunctionGraph 自定义策略样例

- 示例1：授权用户查询函数代码和配置

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "functiongraph:function:list",
        "functiongraph:function:getConfig",
        "functiongraph:function:getCode"
      ]
    }
  ]
}
```

- 示例2：拒绝用户删除函数

拒绝策略需要同时配合其他策略使用，否则没有实际作用。用户被授予的策略中，一个授权项的作用如果同时存在Allow和Deny，则遵循Deny优先。

如果您给用户授予FunctionGraph FullAccess的系统策略，但不希望用户拥有FunctionGraph FullAccess中定义的删除函数权限，您可以创建一条拒绝删除函数的自定义策略，然后同时将FunctionGraph FullAccess和拒绝策略授予用户，根据Deny优先原则，则用户可以对FunctionGraph执行除了删除函数外的所有操作。拒绝策略示例如下：

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "functiongraph:function:delete"
      ]
    }
  ]
}
```

- 示例3：特定资源权限配置

特定资源：授予IAM用户特定资源的相应权限。例如授予IAM用户所属应用Default下函数functionname的相应权限，需将函数functionname设置为指定资源路径，添加资源路径：FUNCTIONGRAPH:*:*:function:Default/functionname。

说明

指定函数资源：

【格式】FUNCTIONGRAPH:*:*:function:所属应用/函数名称

对于函数资源，IAM自动生成资源路径前缀“FUNCTIONGRAPH:*:*:function:”。通过所属应用和函数名称指定具体的资源路径，支持通配符*。例如：

FUNCTIONGRAPH:*:*:function:Default/*表示Default应用下的任意函数。

```

{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "functiongraph:function:list"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "functiongraph:function:listAlias",
        "functiongraph:function:listVersion",
        "functiongraph:function:getConfig",
        "functiongraph:function:getCode",
        "functiongraph:function:updateCode",
        "functiongraph:function:invoke",
        "functiongraph:function:updateConfig",
        "functiongraph:function:createVersion",
        "functiongraph:function:updateAlias",
        "functiongraph:function:createAlias"
      ],
      "Resource": [
        "FUNCTIONGRAPH:*:*:function:Default/*"
      ]
    }
  ]
}

```

2.2.3 FunctionGraph 资源

资源是服务中存在的对象。在FunctionGraph中，资源包括：函数和触发器，您可以在创建自定义策略时，通过指定资源路径来选择特定资源。

表 2-2 FunctionGraph 的指定资源与对应路径

资源类型	资源名称	资源路径
function	函数	<p>【格式】 FunctionGraph::::function:分组/函数名称</p> <p>【说明】 对于桶资源，IAM自动生成资源路径前缀 FunctionGraph:*:*:function: 通过函数名称指定具体的资源路径，支持通配符。例如： FunctionGraph:*:*:function:default/*表示default分组下任意函数。</p>

资源类型	资源名称	资源路径
trigger	触发器	<p>【格式】</p> <p>FunctionGraph:::trigger:触发器ID</p> <p>【说明】</p> <p>对于对象资源，IAM自动生成资源路径前缀 FunctionGraph:::trigger: 通过触发器ID指定具体的资源路径，支持通配符。例如： FunctionGraph:::trigger:*表示任意触发器。</p>

2.3 支持的编程语言

2.3.1 Node.js 语言

√表示支持，×表示不支持

语言版本	是否支持	开发指导
Node.js 6.10	√	接口定义、有关SDK接口说明和函数开发指导请参见 Node.js函数开发指南 。
Node.js 8.10	√	
Node.js 10.16	√	
Node.js 12.13	√	
Node.js 14.18	√	
Node.js 16.17	√	
Node.js 18.15	√	

2.3.2 Python 语言

√表示支持，×表示不支持

语言版本	是否支持	开发指导
Python 2.7	√	接口定义、有关SDK接口说明和函数开发指导请参见 Python函数开发指南 。
Python 3.6	√	
Python 3.9	√	
Python 3.10	√	

2.3.3 Java 语言

√表示支持，×表示不支持

语言版本	是否支持	开发指导
Java 8	√	接口定义、有关SDK接口说明和函数开发指导 请参见 Java函数开发指南 。
Java 11	√	

2.3.4 Go 语言

√表示支持，×表示不支持

语言版本	是否支持	开发指导
Go 1.x	√	接口定义、有关SDK接口说明和函数开发指导 请参见 Go函数开发指南 。

2.3.5 C#语言

√表示支持，×表示不支持

语言版本	是否支持	开发指导
C# (.NET Core 2.1)	√	接口定义、有关SDK接口说明和函数开发指导 请参见 C#函数开发指南 。
C# (.NET Core 3.1)	√	
C# (.NET Core 6.0)	√ (当前仅支持拉美-墨西哥城二)	

2.3.6 PHP 语言

√表示支持，×表示不支持

语言版本	是否支持	开发指导
PHP 7.3	√	接口定义、有关SDK接口说明和开发指导 请参见 PHP函数开发指南 。

2.3.7 定制运行时语言

场景说明

运行时负责运行函数的设置代码、从环境变量读取处理程序名称以及从FunctionGraph运行时API读取调用事件。运行时会将事件数据传递给函数处理程序，并将来自处理程序的响应返回给FunctionGraph。

FunctionGraph支持自定义编程语言运行时。您可以使用可执行文件（名称为bootstrap）的形式将运行时包含在函数的程序包中，当调用一个FunctionGraph函数时，它将运行函数的处理程序方法。

自定义的运行时在FunctionGraph执行环境中运行，它可以是Shell脚本，也可以是可在linux可执行的二进制文件。

📖 说明

在本地开发程序之后打包，必须是ZIP包（Java、Node.js、Python、Go）或者JAR包（Java），上传至FunctionGraph即可运行，无需其它的部署操作。制作ZIP包的时候，单函数入口文件必须在根目录，保证解压后，直接出现函数执行入口文件，才能正常运行。

对于Go runtime，必须在编译之后打zip包，编译后的动态库文件名称必须与函数执行入口的插件名称保持一致，例如：动态库名称为testplugin.so，则“函数执行入口”命名为testplugin.Handler。

编译说明

如果您通过Windows平台编译文件，然后在Linux平台里去执行，需要配置如下参数：
CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build main.go

运行时文件 bootstrap 说明

如果程序包中存在一个名为bootstrap的文件，FunctionGraph将执行该文件。如果引导文件未找到或不是可执行文件，函数在调用后将返回错误。

运行时代码负责完成一些初始化任务，它将在一个循环中处理调用事件，直到它被终止。

初始化任务将对函数的每个实例运行一次以准备用于处理调用的环境。

运行时接口说明

FunctionGraph提供了用于自定义运行时的HTTP API来接收来自函数的调用事件，并在FunctionGraph执行环境中发送回响应数据。

- 获取调用

方法 - Get

路径 - http://\$RUNTIME_API_ADDR/v1/runtime/invoke/request

该接口用来获取下一个事件，响应正文包含事件数据。响应标头包含信息如下。

表 2-3 响应标头信息说明

参数	说明
X-Cff-Request-Id	请求ID。

参数	说明
X-CFF-Access-Key	租户AccessKey，使用该特殊变量需要给函数配置委托。
X-CFF-Auth-Token	Token，使用该特殊变量需要给函数配置委托。
X-CFF-Invoke-Type	函数执行类型。
X-CFF-Secret-Key	租户SecretKey，使用该特殊变量需要给函数配置委托。
X-CFF-Security-Token	Security token，使用该特殊变量需要给函数配置委托。

- **调用响应**

方法 - POST

路径 - `http://$RUNTIME_API_ADDR/v1/runtime/invoke/response/$REQUEST_ID`

该接口将正确的调用响应发送到FunctionGraph。在运行时调用函数处理程序后，将来自函数的响应发布到调用响应路径。

- **错误上报**

方法 - POST

路径 - `http://$RUNTIME_API_ADDR/v1/runtime/invoke/error/$REQUEST_ID`

\$REQUEST_ID为获取事件的响应header中X-Cff-Request-Id变量值，说明请参见[表2-3](#)。

\$RUNTIME_API_ADDR为系统环境变量，说明请参见[表2-4](#)。

该接口将错误的调用响应发送到FunctionGraph。在运行时调用函数处理程序后，将来自函数的响应发布到调用响应路径。

运行时环境变量说明

下面是FunctionGraph执行环境中运行时相关的环境变量列表，除此之外，还有用户自定义的环境变量，都可以在函数代码中直接使用。

表 2-4 环境变量说明

键	值说明
RUNTIME_PROJECT_ID	projectID
RUNTIME_FUNC_NAME	函数名称
RUNTIME_FUNC_VERSION	函数的版本
RUNTIME_PACKAGE	函数组
RUNTIME_HANDLER	函数执行入口
RUNTIME_TIMEOUT	函数超时时间

键	值说明
RUNTIME_USERDATA	用户通过环境变量传入的值
RUNTIME_CPU	分配的CPU数
RUNTIME_MEMORY	分配的内存
RUNTIME_CODE_ROOT	包含函数代码的目录
RUNTIME_API_ADDR	自定义运行时API的主机和端口

用户定义的环境变量也同FunctionGraph环境变量一样，可通过环境变量获取方式直接获取用户定义环境变量。

示例说明

此示例包含1个文件（bootstrap文件），该文件都在Bash中实施。

运行时将从部署程序包加载函数脚本。它使用两个变量来查找脚本。

引导文件bootstrap内容如下：

```
#!/bin/sh
set -o pipefail
#Processing requests loop
while true
do
HEADERS="$(mktemp)"
# Get an event
EVENT_DATA=$(curl -sS -LD "$HEADERS" -X GET "http://$RUNTIME_API_ADDR/v1/runtime/invoication/request")
# Get request id from response header
REQUEST_ID=$(grep -Fi x-cff-request-id "$HEADERS" | tr -d '[:space:]' | cut -d: -f2)
if [ -z "$REQUEST_ID" ]; then
continue
fi
# Process request data
RESPONSE="Echoing request: hello world!"
# Put response
curl -X POST "http://$RUNTIME_API_ADDR/v1/runtime/invoication/response/$REQUEST_ID" -d "$RESPONSE"
done
```

加载脚本后，运行时将在一个循环中处理事件。它使用运行时API从FunctionGraph检索调用事件，将事件传递到处理程序，并将响应发布回给FunctionGraph。

为了获取请求ID，运行时会将来自API响应的标头保存到临时文件，并从该文件读取x-cff-request-id读取请求头的请求唯一标识。将获取到的事件数据做处理并响应发布返回FunctionGraph。

go源码示例，需要通过编译后才可执行。

```
package main

import (
"bytes"
"encoding/json"
"fmt"
"io"
"io/ioutil"
```

```

    "log"
    "net"
    "net/http"
    "os"
    "strings"
    "time"
)
var (
    getRequestUrl      = os.ExpandEnv("http://${RUNTIME_API_ADDR}/v1/runtime/invoication/
request")
    putResponseUrl     = os.ExpandEnv("http://${RUNTIME_API_ADDR}/v1/runtime/invoication/
response/{REQUEST_ID}")
    putErrorResponseUrl = os.ExpandEnv("http://${RUNTIME_API_ADDR}/v1/runtime/
invoication/error/{REQUEST_ID}")
    requestIdInvalidError = fmt.Errorf("request id invalid")
    noRequestAvailableError = fmt.Errorf("no request available")
    putResponseFailedError = fmt.Errorf("put response failed")
    functionPackage       = os.Getenv("RUNTIME_PACKAGE")
    functionName         = os.Getenv("RUNTIME_FUNC_NAME")
    functionVersion      = os.Getenv("RUNTIME_FUNC_VERSION")

    client = http.Client{
        Transport: &http.Transport{
            DialContext: (&net.Dialer{
                Timeout: 3 * time.Second,
            }).DialContext,
        },
    }
)

func main() {
    // main loop for processing requests.
    for {
        requestId, header, payload, err := getRequest()
        if err != nil {
            time.Sleep(50 * time.Millisecond)
            continue
        }

        result, err := processRequestEvent(requestId, header, payload)
        err = putResponse(requestId, result, err)
        if err != nil {
            log.Printf("put response failed, err: %s.", err.Error())
        }
    }
}

// event processing function
func processRequestEvent(requestId string, header http.Header, evtBytes []byte) ([]byte, error) {
    log.Printf("processing request '%s'.", requestId)
    result := fmt.Sprintf("function: %s:%s:%s, request id: %s, headers: %+v, payload: %s",
functionPackage, functionName,
    functionVersion, requestId, header, string(evtBytes))

    var event FunctionEvent
    err := json.Unmarshal(evtBytes, &event)
    if err != nil {
        return (&ErrorMessage{ErrorType: "invalid event", ErrorMessage: "invalid json formatted
event"}).toJsonBytes(), err
    }
}

```

```
    return (&APIGFormatResult{StatusCode: 200, Body: result}).toJsonBytes(), nil
}

func getRequest() (string, http.Header, []byte, error) {
    resp, err := client.Get(getRequestUrl)
    if err != nil {
        log.Printf("get request error, err: %s.", err.Error())
        return "", nil, nil, err
    }
    defer resp.Body.Close()

    // get request id from response header
    requestId := resp.Header.Get("X-CFF-Request-Id")
    if requestId == "" {
        log.Printf("request id not found.")
        return "", nil, nil, requestIdInvalidError
    }

    payload, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        log.Printf("read request body error, err: %s.", err.Error())
        return "", nil, nil, err
    }

    if resp.StatusCode != 200 {
        log.Printf("get request failed, status: %d, message: %s.", resp.StatusCode, string(payload))
        return "", nil, nil, noRequestAvailableError
    }

    log.Printf("get request ok.")
    return requestId, resp.Header, payload, nil
}

func putResponse(requestId string, payload []byte, err error) error {
    var body io.Reader
    if payload != nil && len(payload) > 0 {
        body = bytes.NewBuffer(payload)
    }

    url := ""
    if err == nil {
        url = strings.Replace(putResponseUrl, "{REQUEST_ID}", requestId, -1)
    } else {
        url = strings.Replace(putErrorResponseUrl, "{REQUEST_ID}", requestId, -1)
    }

    resp, err := client.Post(strings.Replace(url, "{REQUEST_ID}", requestId, -1), "", body)
    if err != nil {
        log.Printf("put response error, err: %s.", err.Error())
        return err
    }
    defer resp.Body.Close()

    responsePayload, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        log.Printf("read request body error, err: %s.", err.Error())
        return err
    }

    if resp.StatusCode != 200 {
        log.Printf("put response failed, status: %d, message: %s.", resp.StatusCode,
string(responsePayload))
    }
}
```

```

    return putResponseFailedError
}

return nil
}

type FunctionEvent struct {
    Type string `json:"type"`
    Name string `json:"name"`
}

type APIGFormatResult struct {
    StatusCode    int           `json:"statusCode"`
    IsBase64Encoded bool         `json:"isBase64Encoded"`
    Headers      map[string]string `json:"headers,omitempty"`
    Body         string        `json:"body,omitempty"`
}

func (result *APIGFormatResult) toJsonBytes() []byte {
    data, err := json.MarshalIndent(result, "", " ")
    if err != nil {
        return nil
    }

    return data
}

type ErrorMessage struct {
    ErrorType string `json:"errorType"`
    ErrorMessage string `json:"errorMessage"`
}

func (errMsg *ErrorMessage) toJsonBytes() []byte {
    data, err := json.MarshalIndent(errMsg, "", " ")
    if err != nil {
        return nil
    }

    return data
}

```

代码中的环境变量说明如下，请参见[表2-5](#)。

表 2-5 环境变量说明

环境变量	说明
RUNTIME_FUNC_NAME	函数名称
RUNTIME_FUNC_VERSION	函数版本
RUNTIME_PACKAGE	函数组

支持流式返回

自定义运行时支持在函数中返回超过6MB并且不超过200MB的大报文，以数据流的方式返回。

步骤1 登录函数 workflow 控制台，单击自定义运行时的函数名称进入详情页。

步骤2 选择“设置 > 高级配置”，开启“流式返回”。

----结束

2.3.8 仓颉语言

√表示支持，×表示不支持

语言版本	是否支持
Cangjie 1.0	√

3 构建函数

3.1 创建程序包

要创建FunctionGraph函数，首先需要创建函数部署程序包（包含代码和所有依赖项的文件）。用户可以自行创建部署程序包或直接在FunctionGraph函数控制台在线编辑代码，控制台将创建并上传部署程序包，从而实现FunctionGraph函数的创建。用户在编辑函数代码时支持类似工程方式的管理，可以创建文件、文件夹并对其进行编辑。如果用户代码是上传zip包的方式，则前端进行相应解压展示，提供编辑能力。

说明

- 用户在本地开发程序之后打包，必须是ZIP包（Java、Node.js、Python、Go）或者JAR包（Java），上传至FunctionGraph即可运行，无需其它的部署操作。
- 制作ZIP包的时候，单函数入口文件必须在根目录，保证解压后，直接出现函数执行入口文件，才能正常运行。
- 对于Go runtime，必须在编译之后打zip包，编译后的动态库文件名称必须与函数执行入口的插件名称保持一致，例如：动态库名称为testplugin.so，则“函数执行入口”命名为testplugin.Handler。
- 对于Java runtime，由于Java是编译型语言，所以不能在线编辑代码。如果函数没有引入其他第三方件，可以选择上传函数jar包。如果函数中引入其他三方件，则需要制作包含所有依赖三方件和函数jar的zip包，选择上传zip文件。

FunctionGraph函数支持的上传程序包的方式如表3-1。

表 3-1 代码上传方式说明

运行时	在线编辑	上传ZIP文件	上传JAR包	从OBS上传文件
Node.js	支持	支持	不支持	支持
Python	支持	支持	不支持	支持
Java	不支持	支持	支持	支持
GO	不支持	支持	不支持	支持
C#	不支持	支持	不支持	支持

运行时	在线编辑	上传ZIP文件	上传JAR包	从OBS上传文件
PHP	支持	支持	不支持	支持
定制运行时	支持	支持	不支持	支持

须知

上传代码时，如果代码中包含敏感信息（如账户密码等），请您自行加密，以防止信息泄露。

表 3-2 函数代码上传方式表

代码上传方式	操作
在线编辑	<p>用户在编辑函数代码时支持类似工程方式的管理，可以创建文件、文件夹并对其进行编辑。如果用户代码是上传zip包的方式，则前端进行相应解压展示，并支持用户在函数详情页的“代码”页签，进行在线编辑修改。</p> <ul style="list-style-type: none"> 文件：支持创建文件和文件夹功能。其中包括新建文件，新建文件夹、保存、关闭所有文件功能。 编辑：支持在编码框中，对代码进行撤销、恢复、剪切、复制、粘贴、查找、替换操作。 设置：支持设置编码框中代码字体大小、自动格式化和编码框主题颜色。
上传ZIP文件	<ol style="list-style-type: none"> 在函数详情页的“代码”页签，选择“上传自 > Zip文件”。 单击“添加文件”上传本地代码至平台。上传的zip文件大小限制为40M，如超过40M，请通过OBS上传。
从OBS上传文件	<ol style="list-style-type: none"> 在函数详情页的“代码”页签，选择“上传自 > OBS地址”。 单击“添加文件”上传本地代码至平台。

说明

当您部署的代码大于20M时，在线编辑器将不展示代码，但仍可以测试您的函数。

图 3-1 编辑器不展示代码



Node.js 程序包

在线编辑

FunctionGraph服务预装了适用于Node.js的开发工具包，如果自定义代码只需要软件开发工具包库，则可以使用FunctionGraph控制台的内联编辑器。使用控制台可以编辑代码并将代码上传到FunctionGraph，控制台会将代码及相关的配置信息压缩到FunctionGraph服务能够运行的部署程序包中。

上传程序包

如果编写的代码需要用到其他资源（如使用图形库进行图像处理），则需要先创建FunctionGraph函数部署程序包，然后再使用控制台上传部署程序包。Node.js编程语言支持以下两种方式上传程序包。

须知

- 制作zip包的时候，单函数入口文件必须在根目录，保证解压后，直接出现函数执行入口文件，才能正常运行。
- 解压后的源代码不能超过1.5G，超大代码请联系客服。

- **直接上传程序包**

在创建部署程序包后，可直接从本地上传ZIP程序包，ZIP程序包大小限制为40MB，如果超过该限制，请使用OBS存储桶。

更多函数资源的限制，请参见[使用限制](#)。

- **上传至OBS存储桶**

在创建部署程序包后，可先将.zip文件上传到要在其中创建FunctionGraph函数的区域中的OBS存储桶中，然后指定FunctionGraph函数中设置程序包的OBS存储地址，OBS中ZIP包大小限制为300MB。

更多函数资源的限制，请参见[使用限制](#)。

Python 程序包

在线编辑

FunctionGraph服务预装了适用于Python的开发工具包，如果自定义代码只需要软件开发工具包库，则可以使用FunctionGraph控制台的内联编辑器。使用控制台可以编辑代码并将代码上传到FunctionGraph，控制台会将代码及相关的配置信息压缩到FunctionGraph服务能够运行的部署程序包中。

上传程序包

如果编写的代码需要用到其他资源（如使用图形库进行图像处理），则需要先创建FunctionGraph函数部署程序包，然后再使用控制台上传部署程序包。Python编程语言支持以下两种方式上传程序包。

须知

- 制作zip包的时候，单函数入口文件必须在根目录，保证解压后，直接出现函数执行入口文件，才能正常运行。
- 解压后的源代码不能超过1.5G，超大代码请联系客服。
- 用python语言写代码时，自己创建的包名不能与python标准库同名，否则会提示module加载失败。例如“json”、“lib”，“os”等。

- **直接上传程序包**

在创建部署程序包后，可直接从本地上传ZIP程序包，ZIP程序包大小限制为40MB，如果超过该限制，请使用OBS存储桶。

更多函数资源的限制，请参见[使用限制](#)。

- **上传至OBS存储桶**

在创建部署程序包后，可先将.zip文件上传到要在其中创建FunctionGraph函数的区域中的OBS存储桶中，然后指定FunctionGraph函数中设置程序包的OBS存储地址，OBS中ZIP包大小限制为300MB。

更多函数资源的限制，请参见[使用限制](#)。

Java 程序包

由于Java是编译型语言，所以不能在线编辑代码，只能上传程序包，部署程序包可以是.zip文件或独立的.jar文件。

上传Jar包

- 如果函数没有引入其他依赖包，可以直接上传函数.jar包。
- 如果函数引入了其他依赖包，可先将依赖包上传至OBS桶，创建函数时设置依赖包，并上传函数.jar包。

上传zip

如果函数中引入其他三方件，也可以制作包含所有依赖三方件和函数.jar的.zip包，选择上传.zip文件。

Java编程语言支持以下两种方式上传程序包。

须知

- 制作zip包的时候，单函数入口文件必须在根目录，保证解压后，直接出现函数执行入口文件，才能正常运行。
- 解压后的源代码不能超过1.5G，超大代码请联系客服。

- **直接上传程序包**

在创建部署程序包后，可直接从本地上传ZIP程序包，ZIP程序包大小限制为40MB，如果超过该限制，请使用OBS存储桶。

更多函数资源的限制，请参见[使用限制](#)。

- **上传至OBS存储桶**

在创建部署程序包后，可先将.zip文件上传到要在其中创建FunctionGraph函数的区域中的OBS存储桶中，然后指定FunctionGraph函数中设置程序包的OBS存储地址，OBS中ZIP包大小限制为300MB。

更多函数资源的限制，请参见[使用限制](#)。

GO 语言程序包

上传程序包

只能上传程序包，部署程序包必须是.zip文件。Go编程语言支持以下两种方式上传程序包。

须知

- 制作zip包的时候，单函数入口文件必须在根目录，保证解压后，直接出现函数执行入口文件，才能正常运行。
- 解压后的源代码不能超过1.5G，超大代码请联系客服。

- **直接上传程序包**

在创建部署程序包后，可直接从本地上传ZIP程序包，ZIP程序包大小限制为40MB，如果超过该限制，请使用OBS存储桶。

更多函数资源的限制，请参见[使用限制](#)。

- **上传至OBS存储桶**

在创建部署程序包后，可先将.zip文件上传到要在其中创建FunctionGraph函数的区域中的OBS存储桶中，然后指定FunctionGraph函数中设置程序包的OBS存储地址，OBS中ZIP包大小限制为300MB。

更多函数资源的限制，请参见[使用限制](#)。

C#语言程序包

上传程序包

只能上传程序包，部署程序包必须是.zip文件。C#编程语言支持以下两种方式上传程序包。

须知

- 制作zip包的时候，单函数入口文件必须在根目录，保证解压后，直接出现函数执行入口文件，才能正常运行。
- 解压后的源代码不能超过1.5G，超大代码请联系客服。

- **直接上传程序包**

在创建部署程序包后，可直接从本地上传ZIP程序包，ZIP程序包大小限制为40MB，如果超过该限制，请使用OBS存储桶。

更多函数资源的限制，请参见[使用限制](#)。

- **上传至OBS存储桶**

在创建部署程序包后，可先将.zip文件上传到要在其中创建FunctionGraph函数的区域中的OBS存储桶中，然后指定FunctionGraph函数中设置程序包的OBS存储地址，OBS中ZIP包大小限制为300MB。

更多函数资源的限制，请参见[使用限制](#)。

PHP 语言程序包

在线编辑

FunctionGraph服务预装了适用于PHP的开发工具包，如果自定义代码只需要软件开发工具包库，则可以使用FunctionGraph控制台的内联编辑器。使用控制台可以编辑代码并将代码上传到FunctionGraph，控制台会将代码及相关的配置信息压缩到FunctionGraph服务能够运行的部署程序包中。

上传程序包

如果编写的代码需要用到其他资源（如使用图形库进行图像处理），则需要先创建FunctionGraph函数部署程序包，然后再使用控制台上传部署程序包。PHP编程语言支持以下两种方式上传程序包。

须知

- 制作zip包的时候，单函数入口文件必须在根目录，保证解压后，直接出现函数执行入口文件，才能正常运行。
- 解压后的源代码不能超过1.5G，超大代码请联系客服。

- **直接上传程序包**

在创建部署程序包后，可直接从本地上传ZIP程序包，ZIP程序包大小限制为40MB，如果超过该限制，请使用OBS存储桶。

更多函数资源的限制，请参见[使用限制](#)。

- **上传至OBS存储桶**

在创建部署程序包后，可先将.zip文件上传到要在其中创建FunctionGraph函数的区域中的OBS存储桶中，然后指定FunctionGraph函数中设置程序包的OBS存储地址，OBS中ZIP包大小限制为300MB。

更多函数资源的限制，请参见[使用限制](#)。

定制运行时程序包

在线编辑

使用控制台可以编辑代码并将代码上传到FunctionGraph，控制台会将代码及相关的配置信息压缩到FunctionGraph服务能够运行的部署程序包中。

上传程序包

如果编写的代码需要用到其他资源（如使用图形库进行图像处理），则需要先创建FunctionGraph函数部署程序包，然后再使用控制台上传部署程序包。定制运行时支持以下两种方式上传程序包。

须知

- 制作zip包的时候，单函数入口文件必须在根目录，保证解压后，直接出现函数执行入口文件，才能正常运行。
- 解压后的源代码不能超过1.5G，超大代码请联系客服。

- 直接上传程序包

在创建部署程序包后，可直接从本地上传ZIP程序包，ZIP程序包大小限制为40MB，如果超过该限制，请使用OBS存储桶。

更多函数资源的限制，请参见[使用限制](#)。

- 上传至OBS存储桶

在创建部署程序包后，可先将.zip文件上传到要在其中创建FunctionGraph函数的区域中的OBS存储桶中，然后指定FunctionGraph函数中设置程序包的OBS存储地址，OBS中ZIP包大小限制为300MB。

更多函数资源的限制，请参见[使用限制](#)。

3.2 使用空白模板创建函数

3.2.1 创建事件函数

概述

函数是处理事件的自定义代码，您可以使用空白模板函数创建函数，根据实际业务场景进行函数配置。

由于FunctionGraph承担计算资源的管理工作，在函数完成编码以后，需要为函数设置运算资源等信息，目前主要是在FunctionGraph函数控制台完成。

创建函数时可以使用空模板，也可以[使用示例模板创建函数](#)、[使用容器镜像部署函数](#)。

说明

使用空模板创建函数时，需要设置基础配置信息和代码信息，如[表3-3](#)所示，带*参数为必填项。每个FunctionGraph函数都运行在其自己的环境中，有其自己的资源和文件系统。

前提条件

1. 了解函数支持的编程语言，具体请参见[支持的编程语言](#)。
2. 创建程序包，具体请参见[创建程序包](#)。
3. 创建委托（可选，根据实际情况，确定是否需创建委托），具体请参见[配置委托权限](#)。

操作步骤

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 单击右上方的“创建函数”，进入“创建函数”页面。

3. 选择“创建空白函数”，参见表3-3填写函数信息，带*参数为必填项。

图 3-2 创建空白函数

The screenshot shows a configuration form for creating a function. The 'Function Type' is set to 'Event Function'. The 'Region' is set to 'cn-north-1'. The 'Project' is set to 'default'. The 'Function Name' is empty. The 'Enterprise Project' is set to 'default'. The 'Trust Name' is set to 'No trust'. The 'Runtime' is set to 'Node.js 16.17'. There are links for 'View Enterprise Project', 'Create Trust', and 'View Node.js Function Development Guide'.

表 3-3 函数基础配置信息表

参数	说明
*函数类型	<ul style="list-style-type: none"> 事件函数：通过触发器来触发函数执行。 HTTP函数：用户可以直接发送 HTTP 请求到 URL 触发函数执行。 <p>说明</p> <ul style="list-style-type: none"> HTTP函数当前不区分编程语言，函数执行入口必须在 bootstrap 文件中设置，用户直接写启动命令，端口统一开放成8000。 HTTP函数只允许创建APIG/APIC的触发器类型，其他触发器不支持。 HTTP函数的使用说明请参见创建HTTP函数。
*区域	选择要部署代码的区域。
*函数名称	函数名称，命名规则如下： <ul style="list-style-type: none"> 可包含字母、数字、下划线和中划线，长度不超过60个字符。 以大/小写字母开头，以字母或数字结尾。
委托名称	用户委托函数 workflow 服务去访问其他的云服务，则需要提供权限委托，创建委托，请参见 配置委托权限 。 如果用户函数不访问任何云服务，则不用提供委托名称。
*企业项目	选择已创建的企业项目，将函数添加至企业项目中，默认选择“default”。 <p>说明</p> 如果您没有开通企业管理服务，将无法看到企业项目选项。开通方法请参见 如何开通企业项目 。
运行时	选择用来编写函数的语言。 <p>须知</p> 控制台代码编辑器仅支持Node.js、Python和PHP。

- 高级设置：设置“日志标签”，最多可添加10个标签。函数执行时，可以按照自定义标签配置上报标签到云日志服务(LTS)，用户可以通过标签对日志进行过滤筛选。

表 3-4 日志标签参数说明

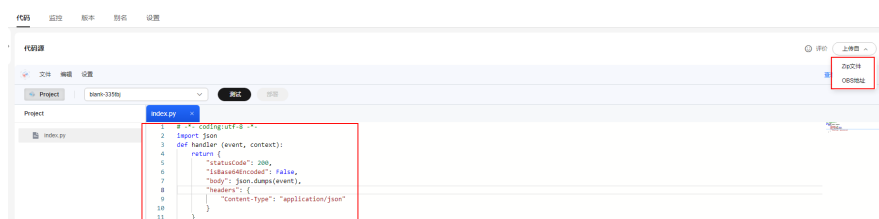
标签键	键/值只能包含数字、字母、下划线和中划线，最大长度64。
标签值	

- 填写完成后单击“创建函数”，页面跳转至代码配置页面，继续配置代码源。

配置代码源

- 您可以根据所选的运行时语言Runtime，参见[创建程序包](#)，选择适合的方式进行代码源部署，完成后单击“部署”。
以下图为例，运行时语言为“Node.js 10.16”，可以选择“在线编辑”、“Zip文件”、“OBS地址”三种方式进行代码源部署。

图 3-3 部署代码源



说明

当使用“OBS地址”方式进行代码源部署时，请确保创建函数选择的“区域”与创建OBS桶选择的“区域”一致。

- 代码若有修改，请修改完成后再次单击“部署”，重新部署代码。

查看代码信息

- 查看代码属性
代码属性展示最新部署代码的大小及上次修改时间。

图 3-4 查看代码属性



- 查看基本信息
函数创建完成后，各语言默认内存和执行超时时间如[图3-5](#)所示，请根据实际业务评估，若需修改“函数执行入口”、“内存（MB）”“执行超时时间（秒）”，可单击“编辑”，在常规设置中修改配置信息，具体请参见[配置常规信息](#)。

图 3-5 编辑基本信息



须知

函数一旦创建，便不能修改运行时语言。

表 3-5 各语言默认基本信息

Runtime	默认基本信息
JAVA	内存 (MB) : 512MB 函数执行入口: com.demo.TriggerTests.apigTest 执行超时时间 (秒) : 15s
Node.js	内存 (MB) : 128 MB 函数执行入口: index.handler 执行超时时间 (秒) : 3s
Custom	内存 (MB) : 128 MB 函数执行入口: bootstrap 执行超时时间 (秒) : 3s
PHP	内存 (MB) : 128 MB 函数执行入口: index.handler 执行超时时间 (秒) : 3s
Python	内存 (MB) : 128 MB 函数执行入口: index.handler 执行超时时间 (秒) : 3s
Go 1.x	内存 (MB) : 128 MB 函数执行入口: handler 执行超时时间 (秒) : 3s

3.2.2 创建 HTTP 函数

概述

HTTP函数专注于优化 Web 服务场景，用户可以直接发送 HTTP 请求到 URL 触发函数执行，从而使用自己的Web服务。HTTP函数只允许创建APIG/APIC的触发器类型，其他触发器不支持。

📖 说明

- HTTP函数当前不区分编程语言，函数执行入口必须在bootstrap文件中设置，用户直接写启动命令，端口统一开放成8000，绑定IP为127.0.0.1。
- bootstrap文件是HTTP函数的启动文件，HTTP函数仅支持读取bootstrap 作为启动文件名，其它名称将无法启动服务，bootstrap启动文件请参见[bootstrap文件示例](#)。
- HTTP函数支持多种开发语言。
- 用户函数需要返回一个合法的http响应报文。
- 该章节均以Nodejs为样例，若需要使用其他语言，则更换语言路径即可，代码包路径无需更换。其他各语言路径请参见[表3-6](#)。
- 关于Go语言构建FunctionGraph HTTP函数，请参考[使用Go构建FunctionGraph HTTP函数](#)。
- 函数发起HTTP请求时，如果是内网访问，则请求IP地址是动态；如果是公网访问，则请求IP地址是固定。如需了解更多详情请咨询技术支持工程师。

前提条件

⚠️ 注意

在调用API前，确保您的业务系统所在网络与API的访问域名或地址互通。

- 若业务系统与HTTP函数在相同VPC内时，可直接访问API。
- 若业务系统与HTTP函数在同一区域的不同VPC内时，可通过创建VPC对等连接，将两个VPC的网络打通，实现同一区域跨VPC访问API。具体步骤请参考[VPC对等连接说明](#)。
- 若业务系统与HTTP函数在不同区域的不同VPC内时，可通过创建云连接实例并加载需要互通的VPC，将两个VPC的网络打通，实现跨区域跨VPC访问API。具体步骤请参考[跨区域VPC互通](#)。
- 若业务系统与HTTP函数通过公网互通，请确保HTTP函数已绑定弹性IP。

1. 准备一个node脚本，代码示例如下：

```
const http = require('http'); // Import Node.js core module

var server = http.createServer(function (req, res) { //create web server
  res.writeHead(200, { 'Content-Type': 'text/html' });
  res.write('<html><body><h2>This is http function.</h2></body></html>');
  res.end();
});

server.listen(8000, '127.0.0.1'); //6 - listen for any incoming requests

console.log('Node.js web server at port 8000 is running..')
```

2. 准备一个bootstrap启动文件，作为HTTP函数的启动文件。

示例：

bootstrap文件内容如下

```
/opt/function/runtime/nodejs12.13/rtsp/nodejs/bin/node $RUNTIME_CODE_ROOT/index.js
```

3. 将上述两个文件打成zip包。

图 3-6 文件打成 zip 包



说明

如果执行HTTP类型是Python函数，则bootstrap文件中执行函数时，建议增加“-u”参数确保日志落盘。例如：

```
/opt/function/runtime/python3.6/rtsp/python/bin/python3 -u $RUNTIME_CODE_ROOT/index.py
```

若需要使用其他语言，则参见表3-6更换语言路径，代码包路径无需更换。

表 3-6 多语言路径说明

语言	路径
Java8	/opt/function/runtime/java8/rtsp/jre/bin/java
Java11	/opt/function/runtime/java11/rtsp/jre/bin/java
Node.js6	/opt/function/runtime/nodejs6.10/rtsp/nodejs/bin/node
Node.js8	/opt/function/runtime/nodejs8.10/rtsp/nodejs/bin/node
Node.js10	/opt/function/runtime/nodejs10.16/rtsp/nodejs/bin/node
Node.js12	/opt/function/runtime/nodejs12.13/rtsp/nodejs/bin/node
Node.js14	/opt/function/runtime/nodejs14.18/rtsp/nodejs/bin/node
Node.js16	/opt/function/runtime/nodejs16.17/rtsp/nodejs/bin/node
Node.js18	/opt/function/runtime/nodejs18.15/rtsp/nodejs/bin/node
Python2.7	/opt/function/runtime/python2.7/rtsp/python/bin/python
Python3.6	/opt/function/runtime/python3.6/rtsp/python/bin/python3
Python3.9	/opt/function/runtime/python3.9/rtsp/python/bin/python3

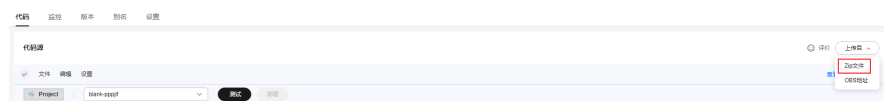
语言	路径
Python3.10	/opt/function/runtime/python3.10/rtsp/python/bin/python3
PHP7.3	/opt/function/runtime/php7.3/rtsp/php/bin/php
C# (.NET Core 2.1)	/opt/function/runtime/dotnet2.1/rtsp/dotnet/dotnet
C# (.NET Core 3.1)	/opt/function/runtime/dotnet3.1/rtsp/dotnet/dotnet

操作步骤

1. 创建函数

- a. 创建HTTP函数，详细配置信息请参见[创建函数](#)，如下参数需注意。
 - 函数类型：HTTP函数
 - 区域：选择要部署代码的区域
- b. 上传代码，此处以“从Zip文件”上传为例，上传准备好的zip包，完成后单击“部署”。

图 3-7 上传自 Zip 文件



说明

当如果使用“OBS地址”方式进行代码源部署时，请确保创建函数选择的“区域”与创建OBS桶选择的“区域”一致。

2. 创建触发器

说明

HTTP函数只允许创建APIG/APIIC的触发器类型，其他触发器不支持。

- a. 进入函数详情页面，选择“设置 > 触发器”页签，单击“创建触发器”。
- b. 配置触发器信息，此处以创建“API网关服务（APIG）”触发器为例，其他配置信息请参见[使用APIG触发器](#)。

图 3-8 创建触发器

创建触发器

触发器类型  API 网关服务 (APIG) 

可以通过HTTPS或者HTTP调用FunctionGraph函数，将各个API操作（如GET和PUT）映射到特定的FunctionGraph函数，当向该API发送HTTPS或者HTTP请求时，API 网关会调用相应的FunctionGraph函数。

当使用APIG触发器时，函数返回体必须为如下json格式：
{"statusCode": 200, "isBase64Encoded": false, "headers": {"Content-Type": "application/json;charset=UTF-8"}, "body": "hello world"}
(isBase64Encoded为true时，body必须为base64编码格式)

* API名称

支持汉字，英文，数字，下划线，且只能以英文和汉字开头，3-64字符

* 分组  [创建分组](#) 

一个分组仅允许被一个HTTP运行时函数关联

* 发布环境  [创建发布环境](#) 

* 安全认证 

IAM: IAM认证，只允许IAM用户能访问，安全级别中等；
App: 采用Appkey&Appsecret认证，安全级别高，推荐使用；
None: 无认证模式，所有用户均可访问，不推荐使用。
无认证模式，安全级别低，所有用户均可访问，不推荐使用

* 请求协议 

* 后端超时(毫秒)

后端超时设置1-60,000毫秒的范围内

 说明

示例中“安全认证”暂时选择“None”，用户在配置时应根据实际情况选择。

- App: 采用 Appkey&Appsecret 认证，安全级别高，推荐使用。
 - IAM: IAM 认证，只允许IAM用户能访问，安全级别中等。
 - None: 无认证模式，所有用户均可访问。
- c. 配置完成后，单击“确定”。API触发器创建完成后，会在API网关生成API“API_test_http”。
3. 发布API
- a. 单击“触发器”页签下的API名称，跳转至API的总览页面。

图 3-9 API 触发器



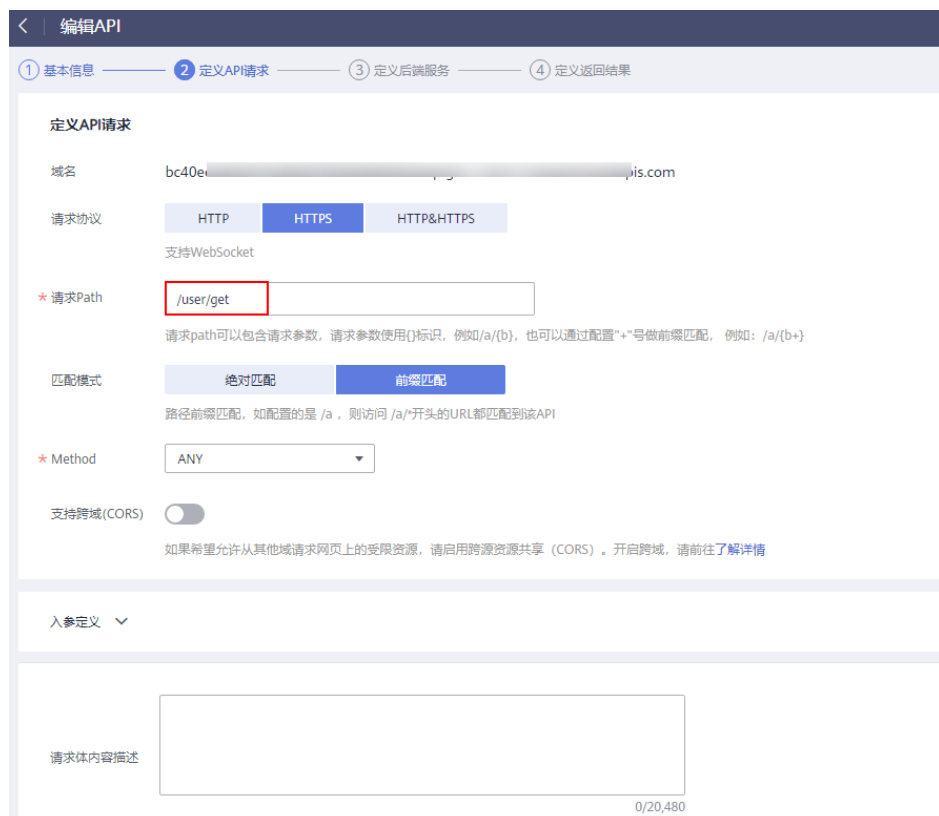
- b. 单击右上方的“编辑”，进入“基本信息”页面。

图 3-10 编辑 API



- c. 单击“下一步”，进入“定义api请求”页面，修改“请求Path”为“/user/get”并单击“立即完成”。

图 3-11 定义 API 请求



- d. 单击“发布API”，在发布页面继续单击“发布”。

4. 触发函数

- a. 返回函数 workflow 控制台，在左侧导航栏选择“函数 > 函数列表”，单击创建的 HTTP 函数进入函数详情页。

- b. 选择“设置 > 触发器”，复制“调用URL”，在浏览器访问。

图 3-12 复制 URL



- c. 查看请求结果。

图 3-13 查看请求结果

```
<html><body><h2>This is http function.</h2></body></html>
```

函数公共请求头

HTTP函数请求头默认携带如下字段。

表 3-7 默认请求头

字段	描述
X-CFF-Request-Id	当前请求ID
X-CFF-Memory	分配的内存
X-CFF-Timeout	函数超时时间
X-CFF-Func-Version	函数版本
X-CFF-Func-Name	函数名称
X-CFF-Project-Id	ProjectID
X-CFF-Package	函数组
X-CFF-Region	当前region

3.3 使用示例模板创建函数

概述

FunctionGraph平台提供了函数模板，在创建函数时选择模板，实现模板代码、运行环境自动填充，快速构建应用程序。

创建函数

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数模板”。
2. 在“函数模板”界面，“云服务”选择“函数 workflow”，模板选择 Python 2.7 的“context 使用指导”，单击“使用模板”。

说明

此处以 Python 2.7 的“context 使用指导”举例，请您根据实际需求选择模板。

3. 选择函数模板后，会加载模板内置的代码、配置信息，进入到“创建函数”界面。
4. 输入函数名称“context”，选择已创建的委托，其他设置保持不变，单击“创建函数”，进入配置详情页。

说明

若不配置委托，在触发函数时，执行结果会返回

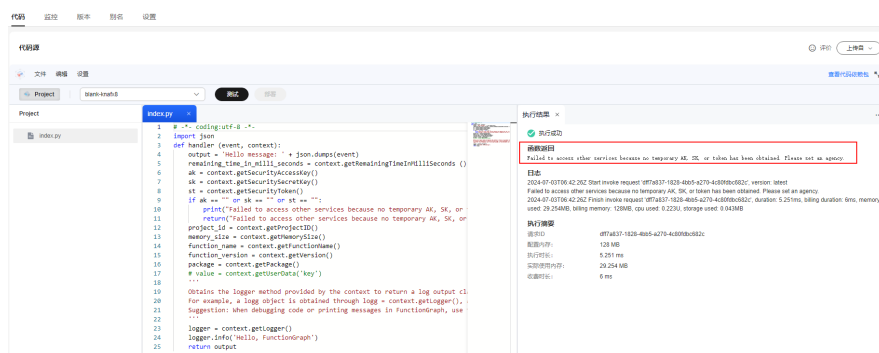
```
Failed to access other services because no temporary AK, SK, or token has been obtained. Please set an agency.
```

5. 请您根据实际业务进行参数配置。

触发函数

1. 在“context”函数的“代码”页签，单击“测试”。
2. 在弹出的“配置测试事件”对话框中，选择“空白模板”，再单击“创建”。
3. 继续单击“测试”，等待测试完成，查看测试结果。

图 3-14 执行成功结果



3.4 使用容器镜像部署函数

概述

用户在本地环境打包容器镜像，只要符合 OCI（Open Container Initiative）标准，都可以上传到 FunctionGraph，由平台加载并启动运行。与原来上传代码方式相比，用户可以使用自定义的代码包，不仅灵活也简化了用户的迁移成本。您可以选择“事件函数”类型创建自定义镜像函数，也可以选择“HTTP 函数”类型创建自定义镜像函数。

使用容器镜像部署函数，开发 HTTP 函数示例，请参见[开发 HTTP 函数](#)。

使用容器镜像部署函数，开发事件函数示例，请参见[开发事件函数](#)。

支持的功能：

- **下载用户镜像**
用户镜像储存在自己的SWR服务中，需要SWR Admin权限才能下载，FunctionGraph会在创建pod前使用swr api生成并设置好临时登录指令。
- **环境变量**
设置FunctionGraph函数的加密配置和环境变量，具体请参见[配置环境变量](#)。
- **挂载外部数据盘**
支持挂载外部数据盘，具体请参见[配置磁盘挂载](#)。
- **计费**
下载镜像、等待镜像Ready不计费。
- **预留实例**
支持预留实例，具体请参见[预留实例](#)。
- **流式返回**
支持在函数中返回超过6MB并且不超过200MB的大报文，以数据流的方式返回。
- **部署新镜像**
支持在函数中重新部署新镜像。

说明

用户容器会使用属主1003、属组1003启动，与其他类型的函数相同。

前提条件

请参见[配置委托权限](#)，创建一个包含“SWR Admin容器镜像服务（SWR）管理员”权限的委托，因为用户镜像储存在SWR服务中，只有拥有“SWR Admin”权限，才能调用与获取，拉取镜像。

操作步骤

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 单击右上方的“创建函数”，进入“创建函数”页面。
3. 选择“容器镜像”，参见[表3-8](#)。

图 3-15 创建容器镜像

函数类型 ?

事件函数

HTTP函数

处理事件请求的函数。您可以通过APIG、OBS、EG等云服务平台触发函数并执行。

区域 ?

▾

不同区域的资源之间内网不互通。请就近选择靠近您业务的区域，可以降低网络时延、提高访问速度。

项目 ?

▾

函数名称 ?

请输入函数名称

可包含字母、数字、下划线和中划线，以大小写字母开头，以字母或数字结尾，长度不超过60个字符。

企业项目 ?

▾
default

[查看企业项目](#)

企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。

权限 ?

使用现有委托

默认情况下，FunctionGraph将创建一个具有基础权限的执行权限，之后添加触发器时，您可以再对此默认角色进行自定义

现有委托 ?

▾
abcd

[创建委托](#)

用户镜像储存在SWR服务中，所选现有委托需要包含SWR Admin权限。

委托权限策略 ?

[查看策略](#)

展示您选择委托 abcd 中的权限策略。您可以在 IAM 控制台增加或者删除策略。

容器镜像 ?

请输入镜像URL

[选择镜像](#)

用于函数的容器镜像的位置。[查看镜像](#)

容器镜像覆盖 (可选)

CMD
Args
Working Dir
用户ID
用户组ID

表 3-8 配置信息

参数	说明
*函数类型	选择函数类型。 事件函数：通过触发器来触发函数执行。 HTTP函数：用户可以直接发送 HTTP 请求到 URL 触发函数执行。 说明 <ul style="list-style-type: none"> 自定义容器镜像需包含HTTP Server，监听端口为8000。 HTTP函数只允许创建APIG/APIC的触发器类型，其他触发器不支持。 事件函数需创建一个HTTP Server并实现Method为POST和Path为/invoke的函数执行入口，可实现Method为POST和Path为/init的函数初始化入口。 通过APIG服务调用函数服务时，isBase64Encoded的值默认为true，表示APIG传递给FunctionGraph的请求体body已经进行Base64编码，需要先对body内容Base64解码后再处理。 函数必须按以下结构返回字符串。 <pre style="margin-left: 20px; background-color: #f0f0f0; padding: 5px;"> { "isBase64Encoded": true false, "statusCode": httpStatusCode, "headers": {"headerName": "headerValue", ...}, "body": "..." } </pre>
*区域	选择要部署代码的区域。

参数	说明
*函数名称	函数名称，命名规则如下： <ul style="list-style-type: none"> 可包含字母、数字、下划线和中划线，长度不超过60个字符。 以大/小写字母开头，以字母或数字结尾。
*企业项目	选择已创建的企业项目，将函数添加至企业项目中，默认选择“default”。 <p>说明</p> 如果您没有开通企业管理服务，将无法看到企业项目选项。开通方法请参见 如何开通企业项目 。
容器镜像	输入镜像URL，即用于函数的容器镜像的位置。您可以单击“查看镜像”，查看自有镜像及共享镜像。如何制作镜像，请参见 制作镜像 。 swr中的镜像名，例如swr.region_id.myhuaweicloud.com/my_group/my_image:latest。
容器镜像覆盖	<ul style="list-style-type: none"> CMD：容器的启动命令，例如"/bin/sh"。该参数为可选参数，不填写，则默认使用镜像中的Entrypoint/CMD。字符串数组，以逗号分开。 Args：容器的启动参数，例如"-args,value1"。该参数为可选参数，不填写，则默认使用镜像中的CMD。字符串数组，以逗号分开。 Working Dir：容器的工作目录，当前不支持创建和修改文件路径，只能为“/”。该参数为可选参数，不填写，则默认使用"/"。 用户ID：镜像运行时的用户ID，若不填写，默认为1003。 用户组ID：镜像运行时的用户组ID，若不填写，默认为1003。
现有委托	选择包含SWR Admin权限的委托，如需创建委托，请参见 创建委托 。

说明

- Command、Args、Working dir三个参数之和不能超过5120。
- 初次执行时需要从SWR中拉取镜像，且冷启动时需要启动容器，所以自定义镜像冷启动比较慢。后续每次冷启动，如果节点上没有镜像，都需要从SWR中拉取。
- 镜像类型支持公开和私有，具体详情请参考[编辑镜像属性](#)。
- 自定义容器镜像开放端口限定为8000。
- 可支持的镜像包最大为10G，当镜像包过大时可以采取一些方式扩容，比如在线题库场景中，可以把原来加载在容器中的题库数据通过外部文件系统挂载盘方式挂载到容器中。
- FunctionGraph通过LTS日志采集容器输出到控制台的所有日志，可以通过标准输出或者开源日志框架重定向到控制台的方式打印业务信息。打印的内容建议包括系统时间、组件名称、代码行、关键数据等来方便定位。
- oom错误时，内存占用大小可以在函数执行结果中查看。
- 用户函数需要返回一个合法的http响应报文。

- 高级设置：设置“日志标签”，最多可添加10个标签。函数执行时，可以按照自定义标签配置上报标签到云日志服务(LTS)，用户可以通过标签对日志进行过滤筛选。

表 3-9 日志标签参数说明

标签键	键/值只能包含数字、字母、下划线和中划线，最大长度64。
标签值	

- 开启流式返回（可选）。
容器镜像函数创建完成后，详情页中选择“设置 > 高级设置”，开启“流式返回”。
- 部署新镜像（可选）。
在“代码”页签，右侧单击“部署新镜像”，在输入框中输入新镜像的URL，单击“确定”完成部署。镜像URL获取方式如下：
 - 登录容器镜像服务控制台，左侧导航栏选择“我的镜像”。
 - 选择“自有镜像”或“他人共享”页签，在镜像列表中单击镜像名称，进入详情页面。
 - 选择“镜像版本”页签，在镜像版本列表中复制下载指令，指令中去掉“docker pull”，即为镜像URL。

示例代码

以下示例使用NodeJS Express，在函数实例初始化时函数工作流会使用POST方法访问/init路径（可选），在每次调用时函数工作流会使用POST方法访问/invoke路径。函数通过req.headers获取context，req.body获取event，返回结果通过HTTP Response结构体输出。

```
const express = require('express');
const app = express();
const PORT = 8000;

app.post('/init', (req, res) => {
  res.send('Hello init\n');
});

app.post('/invoke', (req, res) => {
  res.send('Hello invoke\n');
});

app.listen(PORT, () => {
  console.log(`Listening on http://localhost:${PORT}`);
});
```

3.5 使用 Terraform 部署函数

3.5.1 概述

本手册指导开发者如何使用Terraform创建函数，方便开发者高效的创建函数资源。

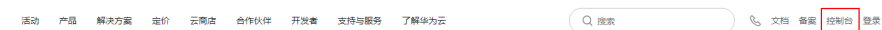
3.5.2 前置条件

3.5.2.1 获取访问密钥

访问密钥（AK/SK，Access Key ID/Secret Access Key）包含访问密钥ID（AK）和秘密访问密钥（SK）两部分，是您在华为云的长期身份凭证，您可以通过访问密钥访问华为云API。

登录华为云，单击右上角的“控制台”。

图 3-16 控制台



在“控制台”页面，鼠标移动至右上方的用户名，在下拉列表中选择“我的凭证”。

图 3-17 我的凭证



在“我的凭证”页面中，单击“访问密钥页签”。

图 3-18 访问密钥页签



单击“新增访问密钥”，输入“描述”信息。

图 3-19 新增访问密钥



单击“确定”，生成并下载访问密钥。

图 3-20 生成下载访问密钥



创建访问密钥成功后，您可以在访问密钥列表中查看访问密钥ID（AK），在下载的文件中查看访问密钥（SK）。

3.5.2.2 准备 Terraform 环境

安装 Terraform 执行环境

Terraform提供了多种环境的安装包，具体可以参考官网(<https://developer.hashicorp.com/terraform/downloads>)。

下面以Linux CentOS (系统需要有访问公网权限)为例指导安装Terraform。

使用root用户登录系统，新建目录/home/Terraform，cd到Terraform目录执行如下命令：

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo
sudo yum -y install terraform
```

基本的 Terraform 命令

执行Terraform后会显示Terraform命令详情。

```
Terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init      Prepare your working directory for other commands
  validate  Check whether the configuration is valid
  plan      Show changes required by the current configuration
  apply     Create or update infrastructure
  destroy   Destroy previously-created infrastructure

All other commands:
  console   Try Terraform expressions at an interactive command prompt
  fmt      Reformat your configuration in the standard style
  force-unlock Release a stuck lock on the current workspace
  get      Install or upgrade remote Terraform modules
  graph    Generate a Graphviz graph of the steps in an operation
  import   Associate existing infrastructure with a Terraform resource
  login    Obtain and save credentials for a remote host
  logout   Remove locally-stored credentials for a remote host
  metadata Metadata related commands
  output   Show output values from your root module
  providers Show the providers required for this configuration
  refresh  Update the state to match remote systems
  show     Show the current state or a saved plan
  state   Advanced state management
  taint   Mark a resource instance as not fully functional
```



```

untaint    Remove the 'tainted' state from a resource instance
version    Show the current Terraform version
workspace  Workspace management

Global options (use these before the subcommand, if any):
-chdir=DIR Switch to a different working directory before executing the
            given subcommand.
-help      Show this help output, or the help for a specified subcommand.
-version   An alias for the "version" subcommand.

```

查看更多命令详情请参考<https://developer.hashicorp.com/terraform/cli>。

3.5.3 基础 Terraform 语法

Terraform配置语言主要基于HCL语法，具有配置简单、可读性强等特点，并且兼容JSON语法。详情参见官网介绍<https://developer.hashicorp.com/terraform/language>，此文不做赘述。

3.5.4 编写函数资源脚本

华为云在Terraform已经注册了provider，函数作为资源挂在华为云的provider下。参考文档https://registry.terraform.io/providers/huaweicloud/huaweicloud/latest/docs/resources/fgs_function。

提供如下样例：

在服务器创建一个main.tf文件，将如下脚本拷贝到main.tf上并保存。

```

terraform {
  required_providers {
    huaweicloud = {
      source = "huaweicloud/huaweicloud"
      version = ">= 1.40.0"
    }
  }
}

provider "huaweicloud" {
  region     = "cn-east-3" #实际的区域
  access_key = "*****" #前面获取的key
  secret_key = "*****" #前面获取的key
}

resource "huaweicloud_fgs_function" "fgs_function" {
  name       = "test_func_rf"
  app        = "default"
  agency     = "function-admin"
  description = "function test"
  handler    = "index.handler"
  memory_size = 128
  timeout    = 3
  runtime    = "Python3.6"
  code_type  = "inline"
  func_code  =
"aW1wb3J0IGpzb24KZGVmIGhhbmRsZXIgaGVhZGV2ZW50LCBjb250ZXh0KT0KICAgIG91dHB1dCA9ICdlZWxsbyBtZ
XNzYWdlOiAnIENsganNvbi5kdW1wcyhldmVudCkKICAgIHJldHVybiBvdXRwdXQ="
}

```

access_key secret_key 需要替换为[获取访问密钥](#)生成的密钥。

3.5.5 使用 Terraform 命令创建函数

进入文件路径，执行terraform init命令初始化一个包含Terraform代码的工作目录。

```
[root@function-deploy ~]# terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of local-registry/huaweicloud/huaweicloud from the dependency lock file
- Using previously-installed local-registry/huaweicloud/huaweicloud v1.45.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

执行terraform apply命令，在Enter a value: 处输入yes。

```
[root@function-deploy ~]# terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# huaweicloud_fgs_function.fgs_function will be created
+ resource "huaweicloud_fgs_function" "fgs_function" {
+ agency                = "function-admin"
+ app                   = "default"
+ app_agency            = (Known after apply)
+ code_filename         = (Known after apply)
+ code_type             = "inline"
+ depend_list           = (Known after apply)
+ description           = "function test"
+ enterprise_project_id = (Known after apply)
+ func_code              = "d78442e980ca8780ab5b0406f5e0ac10d59e08089"
+ functiongraph_version = (Known after apply)
+ handler               = "index.handler"
+ id                    = (Known after apply)
+ initializer_handler   = (Known after apply)
+ initializer_timeout   = (Known after apply)
+ memory_size           = 128
+ mount_user_group_id   = (Known after apply)
+ mount_user_id         = (Known after apply)
+ name                  = "zyl_test_func_01"
+ region                = (Known after apply)
+ runtime               = "python3.6"
+ timeout               = 5
+ urn                   = (Known after apply)
+ version               = (Known after apply)
}

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

huaweicloud_fgs_function.fgs_function: Creating...
huaweicloud_fgs_function.fgs_function: Creation complete after 1s [id=urn:fs:cn-east-3:00d901fba00f5e12f#ac00753b3532d:function:default:zyl_test_func_01:latest]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

执行成功代表函数创建完成。

4 配置函数

4.1 配置初始化

概述

初始化函数在函数实例启动成功后执行，执行成功后，实例才能开始调用请求处理函数处理请求。FunctionGraph保证一个函数实例在生命周期内，初始化函数成功执行且只能成功执行一次。初始化函数的执行时间也会被计量，用户需要为此付费，计费方式同请求处理函数。

应用场景

多个请求处理可以共享的业务逻辑适合放到初始化函数，以降低函数时延，例如深度学习场景下加载规格较大的模型、数据库场景下连接池构建。

前提条件

已创建函数。

初始化函数

步骤1 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 生命周期”，开启“初始化配置”。

图 4-1 开启初始化配置



表 4-1 初始化配置参数说明

参数	说明
配置初始化函数	如需初始化，请开启此参数。
初始化超时时间（秒）	函数初始化的超时时间，如开启函数初始化功能则设置，不开启则不设置。 函数初始化超时时间设置范围为1-300秒。
函数初始化入口	在函数配置页面中，可以选择开启函数初始化功能。各runtime的函数初始化入口命名规范与原有函数执行入口保持一致。如Node.js和Python函数，命名规则：[文件名].[初始化函数名]。 说明 <ul style="list-style-type: none"> 如不开启函数初始化功能则无需配置函数初始化入口。 函数初始化入口需要和函数执行入口在同一文件下。

📖 说明

- 开启函数初始化功能后，各runtime的函数初始化入口命名规范与原有函数执行入口保持一致。如Node.js和Python函数，命名规则：[文件名].[初始化函数名]。
- 函数代码配置信息请参见[创建程序包](#)。

---结束

4.2 配置常规信息

概述

在函数创建完成后，“内存”、“函数执行入口”、“执行超时时间”会根据您所选的“运行时语言”默认填写。如果您需要贴合实际业务场景修改配置，请参见本章节进行配置。

前提条件

已创建函数。

操作步骤

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 选择“设置 > 常规设置”，参见[表1 基本信息配置说明](#)填写函数信息，带*参数为必填项。

表 4-2 基本信息配置说明

参数	说明
所属应用	<p>当前创建的新函数所属应用均为“default”应用，且无法更改。</p> <p>须知 “应用”实际作用就是文件夹功能。新版本里会逐步弱化并下线老界面的“应用”概念，未来会通过标签分组的方式来管理函数的分类等。</p>
*函数执行入口	<ul style="list-style-type: none"> Node.js、Python和PHP函数执行入口的命名规则： [文件名].[执行函数名]，必须包含“.”。 例如：myfunction.handler。 Java函数执行入口的命名规则：[包名].[类名].[执行函数名]。 例如：com.xxxxx.exp.Myfunction.myHandler。 Go函数执行入口的命名规则：与用户上传的代码包中的可执行文件名保持一致。 例如：用户编译的可执行文件名为handler，则填handler。 C#函数执行入口的命名规则：格式为[程序集名]::[命名空间].[类名]::[执行函数名] 例如：HelloCsharp::Example.Hello::Handler。
*企业项目	<p>选择已创建的企业项目，将函数添加至企业项目中，默认选择“default”。</p> <p>说明 如果您没有开通企业管理服务，将无法看到企业项目选项。开通方法请参见如何开通企业项目。</p>
*执行超时时间（秒）	<p>在“配置”页签，函数运行的超时时间，超时的函数将被强行停止。如果执行时间超过90秒，请采用异步调用的方式。</p> <p>函数超时时间设置范围为3~259200秒。</p> <p>说明 前台等待后台返回结果的最大时长为90s，如果设置的超时时间超过90秒，且函数实际运行时间也超过90秒，前台会提醒超时，但是后台仍在运行，可以在日志中查看返回结果。</p>
内存（MB）	<p>函数实例内存规格，取值范围：128、256、512、768、1024、1280、1536、1792、2048、2560、3072、3584、4096、8192、10240。</p>
描述	<p>填写对函数的描述，不超过512个字符。</p>

4. 填写完成后单击“保存”。

4.3 配置委托权限

概述

大部分场景下，FunctionGraph都需要与其他云服务协同工作，通过创建云服务委托，让FunctionGraph以您的身份使用其他云服务，代替您进行一些资源运维工作。

应用场景

若您在FunctionGraph服务中使用如下场景，请先[创建委托](#)。创建委托时授予的权限类型请您根据实际业务需要进行调整，比如前期您在开发阶段授予Admin权限，后期在**生产环境中建议您调整为细粒度最小使用权限**，保证业务所需权限的同时，也降低权限过大的风险。具体对应授权项参见[表1 常见授权项选择](#)进行选择。

表 4-3 常见授权项选择

场景	Admin权限	细粒度最小使用权限	说明
使用自定义镜像	SWR Administrator	暂不支持	SWR Admin：容器镜像服务（SWR）管理员，拥有该服务下的所有权限。 如何创建自定义镜像，请参见 使用容器镜像部署函数 。
挂载sfs turbo文件系统	SFS Turbo ReadOnlyAccess	sfsturbo:shares:getShare（查询单个文件系统详细信息） sfsturbo:shares:showFsDir（查询目录是否存在）	SFS Turbo ReadOnlyAccess：弹性文件服务SFS Turbo的只读权限。 sfsturbo:shares:getShare：您拥有SFS服务下查询单个文件系统信息的权限。 sfsturbo:shares:showFsDir：您拥有SFS服务下查询目录是否存在的权限。 如何挂载挂载sfs turbo文件系统，请参见 添加sfs turbo文件系统 。
挂载ECS共享目录	ECS ReadOnlyAccess	ecs:cloudServers:get（查询云服务器详情）	ECS ReadOnlyAccess：弹性云服务器器的只读访问权限。 ecs:cloudServers:get：您拥有ECS服务下查询云服务器信息的权限。 如何挂载ECS共享目录，请参见 添加ECS共享目录 。

场景	Admin权限	细粒度最小使用权限	说明
配置预留实例策略	AOM ReadOnlyAccess	aom:metric:get (查询指标) aom:metric:list (查询指标列表)	AOM ReadOnlyAccess: 应用运维管理服务只读权限。 aom:metric:get: 您拥有AOM服务下查询某个指标信息的权限。 aom:metric:list: 您拥有AOM服务下查询指标列表详情的权限。
使用DIS触发器	DIS Administrator	暂不支持	数据接入服务 (DIS) 管理员, 拥有该服务下的所有权限。 如何创建DIS触发器, 请参见 使用DIS触发器 。
使用DMS触发器	DMS ReadOnlyAccess	dms:instance:get (查看实例详情信息)	DMS ReadOnlyAccess: 分布式消息服务只读权限。 dms:instance:get: 您拥有DMS服务下查看实例详情信息的权限。
配置跨域VPC访问	VPC Administrator	vpc:ports:get (查询端口) vpc:ports:create (创建端口) vpc:vpcs:get (查询VPC) vpc:subnets:get (查询子网) vpc:vips:delete (虚IP解绑VM) vpc:securityGroups:get (查询安全组列表或详情)	拥有VPC Administrator权限的用户可以对VPC内所有资源执行任意操作。在函数配置跨VPC访问时, 则函数必须配置具备VPC管理权限的委托。 VPC细粒度最小使用权限: 您拥有VPC服务下虚IP解绑VM、查询端口、创建端口、查询VPC、查询子网、查询安全组列表或详情的权限。 如何配置跨域VPC访问, 请参见 配置网络 。
DNS域名解析	DNS ReadOnlyAccess	dns:recordset:get (查询租户Record Set资源) dns:zone:get (查询租户zone) dns:recordset:list (查询Record Set列表) dns:zone:list (查询zone列表)	DNS ReadOnlyAccess: 云解析服务只读权限, 拥有该权限的用户仅能查看云解析服务资源。在函数配置调用DNS服务的接口解析内网域名时, 则函数必须具备读取DNS资源权限的委托。 DNS细粒度最小使用权限: 您拥有DNS服务下查询租户Record Set资源列表和查询租户zone列表的权限。 如何调用DNS服务的接口解析内网域名, 请参见 解析DNS内网域名 。

场景	Admin权限	细粒度最小使用权限	说明
开启异步通知	目标服务为OBS时: OBS Administrator	obs:bucket:HeadBucket (获取桶元数据) obs:bucket:CreateBucket (创建桶) obs:object:PutObject (PUT上传、POST上传、复制对象、追加写对象、初始化上传段任务、上传段、合并段)	OBS Administrator: 对象存储服务管理员, 拥有该服务下的所有权限。 OBS细粒度最小使用权限: 您拥有OBS服务下获取桶元数据、创建桶、PUT上传、POST上传、复制对象、追加写对象、初始化上传段任务、上传段、合并段的权限。 如何配置异步通知请参见 配置函数异步 。
	目标服务为SMN时: SMN Administrator	smn:topic:publish (发布消息) smn:topic:list (查询主题列表)	SMN Administrator: 消息通知服务管理员, 拥有该服务下的所有权限。 SMN细粒度最小使用权限: 您拥有SMN服务下发布消息、查询主题列表的权限。 如何配置异步通知请参见 配置函数异步 。
	目标服务为DIS时: DIS Administrator	暂不支持	DIS Administrator: 数据接入服务管理员, 拥有该服务下的所有权限。 如何配置异步通知请参见 配置函数异步 。

创建委托

说明

如下示例表示: 为FunctionGraph赋予VPC Administrator权限, 仅在授权区域生效。

按照如下参数设置委托, 创建委托的具体步骤请参见[如何创建委托](#)。

1. 登录统一身份认证服务 (IAM) 控制台。
2. 在统一身份认证服务 (IAM) 的左侧导航窗格中, 选择“委托”页签, 单击右上方的“+创建委托”。

图 4-2 创建委托



3. 开始配置委托。

图 4-3 填写基本信息

* 委托名称

* 委托类型 普通账号
将账号内资源的操作权限委托给其他华为云账号。

云服务
将账号内资源的操作权限委托给华为云服务。

* 云服务

* 持续时间

描述

0/255 ↕

- 委托名称：serverless-trust。
 - 委托类型：选择“云服务”。
 - 云服务：选择“函数工作流 FunctionGraph”。
 - 持续时间：选择“永久”。
 - 描述：填写描述信息。
4. 单击“下一步”，进入委托选择页面，在右方搜索框中搜索需要添加的权限并勾选。此处以添加VPC Administrator权限为例。

图 4-4 选择策略

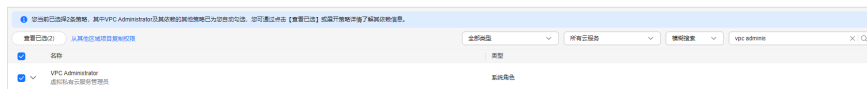
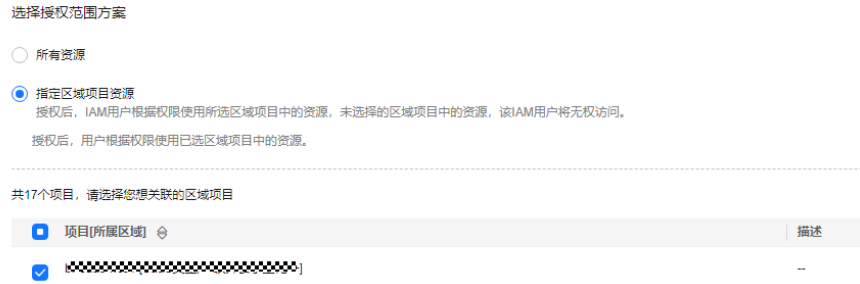


表 4-4 委托权限示例

权限名称	使用描述/场景
VPC Administrator	虚拟私有云服务管理员。

5. 单击“下一步”，选择权限的作用范围。此处以勾选“区域级项目”为例，您需根据业务需要选择对应的权限作用范围。

图 4-5 根据业务需要选择对应的权限



说明

如果系统策略不满足授权要求，管理员可以通过“可视化视图”和“JSON视图”两种方式创建自定义策略，并通过给用户组授予自定义策略来进行精细的访问控制，具体详情请参见[创建自定义策略](#)。

配置函数委托

1. 在服务控制台左侧导航栏，选择“计算 > 函数工作流”。进入函数工作流控制台后在左侧导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 选择“设置 > 权限”，单击“创建委托”，参见2~5，根据实际业务场景，配置函数委托。

表 4-5 配置函数委托参数说明

参数	说明
函数配置委托	选择已创建的函数委托。
函数执行委托	勾选“为函数执行单独设置委托”，需配置此参数。

说明

- 在创建函数过程中选择委托时，勾选“为函数执行单独设置委托”时，弹出“函数执行委托”，函数执行委托与函数配置委托可独立设置，这将减小不必要的性能损耗；不勾选时，函数执行委托和函数配置委托将使用同一委托，即使用同一个选择的委托或不使用任何委托。如图4-6所示。

图 4-6 设置委托



- 函数配置委托，比如函数需要创建DIS触发器，则需要配置具有DIS访问权限的委托。当没有使用任何函数配置委托或者函数配置委托不存在时，不能创建DIS触发器。
- 函数执行委托配置后用户可以通过函数执行入口方法中的context参数获取具有委托中权限的token、AK、SK，用于访问其他云服务。

- 配置完成后单击“保存”。

修改委托

修改委托：如果需要修改委托的权限、持续时间、描述等，可以在IAM控制台修改委托。

⚠ 注意

- FunctionGraph相关委托修改后，约10分钟生效（如context.getToken更新）。
- 通过context获取的委托相关信息有效期24h，需要注意在失效前及时刷新。

4.4 配置网络

访问公网

函数创建成功后，默认具有公网访问权限，即函数可直接访问公网上的服务。函数访问公网上的服务需要固定公网出口IP的场景（例如被访问服务需要白名单验证），可以通过[开启VPC](#)，在VPC内配置公网NAT网关绑定EIP的方式实现，具体请参见[配置固定公网IP](#)。

访问 VPC

函数支持用户创建虚拟私有云（VPC）并访问自己VPC内的资源。VPC开启后，函数不再具有默认的公网访问权限，如果需要访问公网，可通过在VPC内配置公网NAT网关绑定EIP的方式实现，具体请参见[配置固定公网IP](#)。

函数VPC配置支持开启IPv6。（当前仅华东二支持）

相关权限

配置委托权限请参见[配置委托权限](#)。

- 使用VPC功能：需要为函数配置“VPC Administrator”委托权限，或参见[表1 最小授权项配置](#)授予访问VPC需要配置的最小权限，让函数拥有操作相关云服务的权限。

表 4-6 最小授权项配置

权限	授权项
删除端口	vpc:ports:delete
查询端口	vpc:ports:get
创建端口	vpc:ports:create
查询VPC	vpc:vpcs:get
查询子网	vpc:subnets:get

- 解析内网域名：需要为函数配置“DNS ReadOnlyAccess”委托权限。

操作步骤

1. 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 选择“设置 > 网络设置”，开启“函数访问VPC内资源”，配置VPC和子网。

开启“函数访问VPC内资源”时，需在默认“default”安全组中配置如下要求的入方向规则和出方向规则，添加规则详情请参见。

- 入方向规则：配置“策略”为“允许”、“协议端口”为“ICMP”、“源地址”设置的最小范围为函数所选VPC网段的规则。例如，函数的VPC网段为192.168.x.x/24，安全组的入方向需增加一个规则：“策略”选择“允许”、“协议端口”选择“ICMP”、“源地址”设置最小范围的网段为192.168.x.x/24。

图 4-7 入方向规则



- 出方向规则：配置“策略”为“允许”的相关业务端口规则。

图 4-8 配置 VPC



说明

1. 创建虚拟私有云VPC和子网请参见[创建虚拟私有云基本信息及默认子网](#)。
2. 使用跨VPC访问能力时必须配置具备VPC管理权限的委托，创建委托请参考[配置委托权限](#)。
3. 单个租户在一个项目下所有的函数最多能绑定4个不同的子网。（不区分VPC，此处的项目指在创建账号的时候分配的一个32位的唯一值project_id，且主账户和子账户的project_id相同。）

4. 配置域名（可选）。

如果函数需要通过内网域名访问VPC内的服务，可配置和VPC绑定的域名，域名可以配置多个，如图4-8所示。

说明

1. 创建内网域名请参见[创建内网域名](#)。
 2. 当前函数仅支持对A记录集类型的域名解析，记录集添加请参见[记录集类型及配置规则](#)。
5. 配置VPC对等连接网段（可选）。

图 4-9 VPC 对等连接网段

函数访问VPC内资源

VPC [创建虚拟私有云](#)

子网 [创建子网](#)

域名 [创建域名](#)

VPC对等连接网段 [创建域名](#)

您可以声明代码中使用到的VPC网段，用以检测是否与服务器使用VPC网段冲突。网段间使用分号分隔且不能超过5个。

说明

- 您可以声明代码中使用到的VPC网段，用以检测是否与服务器使用VPC网段冲突。
 - 当前中国-香港、亚太-曼谷、亚太-新加坡、亚太-雅加达、拉美-墨西哥城一、拉美-墨西哥城二、拉美-圣保罗一、拉美-圣地亚哥暂不支持。
6. 开启IPv6（可选，当前仅华东二支持）。
- a. 创建虚拟私有云，确保默认子网配置中支持开启IPv6，具体详情请参见[创建虚拟私有云和子网](#)。
 - b. 在步骤3的“VPC”和“子网”中选择刚才创建的已开启IPv6的VPC和子网。

图 4-10 开启 IPv6

函数访问VPC内资源

VPC [创建虚拟私有云](#)

子网 [创建子网](#)

域名 [创建域名](#)

VPC对等连接网段 [创建域名](#)

开启IPv6 [创建域名](#)

开启IPv6功能后，将不能关闭

说明

开启IPv6功能后，将自动为子网分配IPv6网段，暂不支持自定义设置IPv6网段。该功能一旦开启，将不能关闭，更多介绍请参见[IPv6网络](#)。

7. 配置完成后单击“保存”。

配置固定公网 IP

函数需要在VPC内访问公网或者需要固定公网IP的场景，可以选择给VPC添加公网NAT网关并绑定EIP的方式。

前提条件

1. 已创建虚拟私有云和子网，请参见[创建虚拟私有云基本信息及默认子网](#)。
2. 已申请弹性云公网IP，请参见[申请弹性公网IP](#)。

创建公网NAT网关步骤如下

1. 在服务控制台左侧导航栏，继续选择“网络> NAT网关”进入NAT网关控制台，单击“购买公网NAT网关”。
2. 在公网NAT网关购买页面，输入相关信息，选择已创建的虚拟私有云及子网（此处以vpc-01为例），在确认规格信息后提交，完成购买。具体操作步骤请参见[购买公网NAT网关](#)。
3. 购买完成后，单击公网NAT网关名称进入详情页面，选择“添加SNAT规则”，单击“确定”完成配置。

网络限制

根据对网络的不同设置，函数有以下网络访问能力，您可按需设置。

网络配置	说明
允许函数访问公网	当前函数默认的公网NAT访问带宽在多个租户间共享，带宽小，仅适合小量调用的测试业务场景使用；如果对带宽、性能、可靠性有高要求的生产业务场景，需开启函数访问VPC，在VPC内添加公网NAT网关并绑定EIP，分配独占的外网访问带宽。
允许函数访问VPC内资源	开启“允许函数访问VPC内资源”时，函数将禁用默认网卡并使用VPC绑定的网卡，是否允许公网访问由配置的VPC决定，开关“允许函数访问公网”将不生效。
仅允许指定的VPC调用函数	开启“仅允许指定的VPC调用函数”时，将仅允许通过指定的VPC调用函数，并禁止通过公网调用函数。

4.5 配置磁盘挂载

概述

FuncitonGraph提供了文件系统挂载功能，多个函数可以通过共用一个文件系统，实现文件共享。相比于对单个函数实例分配的临时磁盘空间限制，可以极大扩展函数的执行和存储空间。

场景介绍

须知

使用磁盘挂载功能需要开放如下端口：

1. 111、445、2049、2051、2052、20048。
2. 对于Ubuntu系统还需再开放3个端口，获取方式请在任意目录下执行如下命令。

```
rpcinfo -plgrep mountd|grep tcp
```

具体请参见[弹性文件服务会占用用户的哪些资源](#)。

目前FunctionGraph函数支持以下文件系统配置。

- SFS文件系统
弹性文件服务（Scalable File Service, SFS）提供按需扩展的高性能文件存储（NAS），可为云上多个弹性云服务器（Elastic Cloud Server, ECS），容器（CCE&CCI），裸金属服务器（BMS）提供共享访问。SFS为用户提供一个完全托管的共享文件存储，能够弹性伸缩至PB规模，具备高可用性和持久性，为海量数据、高带宽型应用提供有力支持。适用于多种应用场景，包括HPC、媒体处理、文件共享、内容管理和Web服务等。
- SFS Turbo文件系统
SFS Turbo分为SFS Turbo标准型(500GB~32TB)、SFS Turbo标准型-增强版(10TB~320TB)、SFS Turbo性能型(500GB~32TB)和SFS Turbo性能型-增强版(10TB~320TB)。SFS Turbo为用户提供一个完全托管的共享文件存储，能够弹性伸缩至320TB规模，具备高可用性和持久性，为海量的小文件、低延迟高IOPS型应用提供有力支持。适用于多种应用场景，包括高性能网站、日志存储、压缩解压、DevOps、企业办公、容器应用等。详情请参见[SFS产品介绍](#)。
- ECS共享目录
ECS共享目录是通过nfs服务，把ECS上的指定目录设置为共享文件系统（详情请参见[添加ECS共享目录](#)），函数（和ECS相同的VPC配置）可以挂载对应目录进行读写等操作，实现计算资源的动态扩展。此类型适合业务不太频繁的场景。

使用文件系统挂载功能具有以下优势：

- 函数执行空间相比于/tmp，可以极大扩展存储空间。
- 多个函数之间可以共享访问已经配置好的文件系统。
- ECS计算资源动态扩展，利用ECS已有的存储能力实现更大的计算能力。

📖 说明

您可以在/tmp路径下写临时文件，最大不能超过10,240MB。

创建委托

为函数添加文件系统配置需要先给函数设置相关服务的委托。

创建委托时，委托类型选择云服务，云服务选择FunctionGraph，因为委托数目有限，而且目前界面上不支持修改，建议可以创建一个权限较大的委托（Tenant Administrator），可以支持在函数中操作当前区域内的所有资源，请参见[配置委托权限](#)。

创建文件系统

进入弹性文件服务，创建SFS容量型或者SFS Turbo文件系统，请参见[创建文件系统](#)。

添加 SFS 容量型文件系统

设置委托

进入需要进行挂载配置的函数详情页，在配置页选择已有的委托（委托需要拥有当前Region的sfs administrator权限）。

如果没有委托，需要创建新委托，创建完后回到此界面选择。

添加挂载配置

在函数详情页单击“设置 > 磁盘挂载”，单击“添加挂载”，如果当前没有添加过文件系统配置，那么需要设置一下用户ID和组ID。

用户ID和用户组ID分别对应Linux系统中的uid和gid，作为函数在运行中访问文件系统的身份。

📖 说明

用户ID：支持输入-1或1~65534的整数且不包含1000和1002，默认值-1，表示FunctionGraph后台自动分配身份标识。

用户组ID：支持输入-1或1~65534的整数且不包含1000和1002，默认值-1，表示FunctionGraph后台自动分配身份标识。

如果已经在云上的服务器挂载过sfs，有一个目录的属主是test-user，那么就可以用id test-user命令查询对应的uid和gid。

接下来要选择需要挂载的sfs文件系统，然后设置在函数中访问的目录。

须知

其中函数访问路径最多设置为两级，/mnt和/home都是已存在的路径，FunctionGraph V1版本推荐挂载在以/mnt或/home开头的路径下。FunctionGraph V2版本增加了校验，建议您挂载在/mnt或/home开头的路径下，如果直接挂载/mnt或/home一级目录，会报“failed to mount exist system path”。

添加 sfs turbo 文件系统

设置委托

挂载sfs turbo文件系统需要给函数设置委托（至少拥有sfs administrator以及VPC administrator权限）。如果没有对应权限的委托，需要新创建。

设置VPC

sfs turbo涉及VPC内部网络访问，添加sfs turbo文件系统前需要给函数配置sfs turbo对应的VPC。

1. 在弹性文件服务中，获取需要挂载的文件系统的VPC和子网信息，具体操作请参见[管理文件系统](#)。
2. 参见[配置网络](#)开启VPC访问，输入1中获取的VPC和子网。

添加挂载-SFS Turbo

添加sfs turbo和添加sfs过程相似，只要选好需要挂载的文件系统，设置好函数访问路径即可。

添加 ECS 共享目录

添加委托

挂载ECS共享目录需要给函数设置委托（至少拥有tenant guest以及VPC administrator权限），如果没有对应权限的委托，需要新创建，详情请参见[创建委托](#)。

配置VPC

添加ECS共享目录前，也需要给函数配置ECS对应的VPC，可以到ECS详情页的“基本信息”页签中查看“虚拟私有云”。单击虚拟私有云名称，进入虚拟私有云的详情页，查看子网。

获取到这两个信息后，可以在函数配置中配置对应的VPC。

添加挂载-ECS

需要在界面上输入ECS上的共享目录路径信息和函数访问路径。

图 4-11 填写路径信息

* 共享目录路径 请输入实际存在的路径，以免挂载出现问题。

* 函数访问路径 函数访问此文件系统的路径。不能使用系统已经存在的目录，建议用/mnt或者/home开头的两级目录,最多只能为两级目录。

后续操作

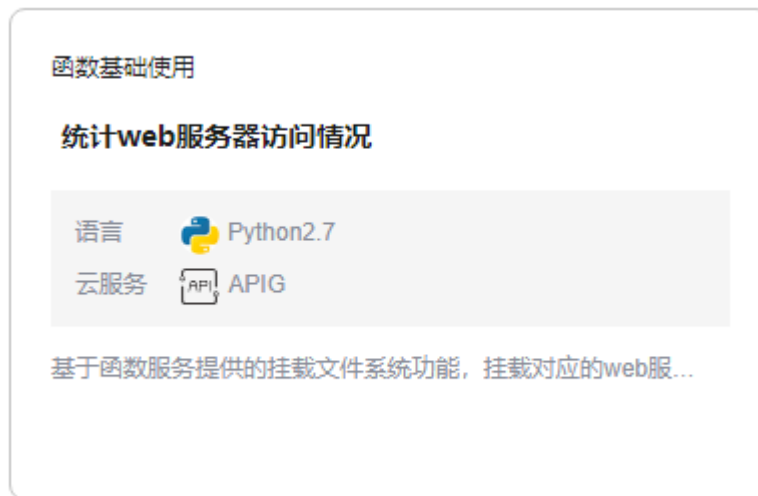
当函数挂载了文件系统配置后，对函数访问路径的读写就相当于对相关文件系统的读写。

如果把日志路径配置为函数访问路径的子目录，就可以轻松实现函数日志的持久化。

如下以函数挂载SFS Turbo为例，同时使用“统计web服务器访问情况”的函数模板，对运行在云上的服务器进行日志分析。

- 步骤1** 登录[函数 workflow 控制台](#)，在左侧导航栏选择“函数模板”。
- 步骤2** 在“函数模板”界面右上角搜索框中，输入“统计web服务器访问情况”进行搜索。
- 步骤3** 在搜索结果中，单击“使用模板”进入配置界面，如[图4-12](#)所示，参数配置如下：

图 4-12 函数模板



- 区域：选择与创建的VPC、文件系统一致的区域，创建VPC和文件系统的详情请参见[配置网络](#)、[创建文件系统](#)。
- 项目：默认。
- 函数名称：自定义。
- 委托名称：选择拥有文件系统、VPC和APIG权限的委托，创建委托详情请参见[配置委托权限](#)。
- 企业项目：根据实际业务选择。
- 环境变量：参数“access_log_path”是日志文件的地址，此处配置为“/home/test/access_log.log”。

📖 说明

文件系统是绝对路径，使用文件系统时需在前面添加“/”；如不挂载文件系统，则无需添加“/”，则参数配置为“code/access_log.log”。

- 触发器类型：默认为“API 网关服务（APIG）”，APIG配置详情请参见[使用APIG触发器](#)。
- API名称：自定义。
- 分组：根据实际业务选择。
- 发布环境：选择“RELEASE”。
- 安全认证：选择“None”。
- 请求协议、后端超时（毫秒）：默认。

步骤4 参数配置完成后，单击“创建函数”，完成“统计web服务器访问情况”的函数创建。

步骤5 进入创建好的函数详情页中，选择“代码”页签，在“index.py”文件中添加如下代码，然后单击“部署”。

```
import shutil
shutil.copyfile('/opt/function/code/access_log.log', '/home/test/access_log.log')
```

图 4-13 添加代码



同时，添加公共依赖包“Jinja2-2.10”，添加依赖包详情请参见[如何为函数添加依赖包](#)。

说明

如果不挂载文件系统，则无需增加以上代码。

步骤6 在函数详情页中，选择“设置 > 网络配置”，开启“函数访问VPC内资源”。“VPC”和“子网”选择已创建好的VPC和子网，单击“保存”。

图 4-14 配置函数访问 VPC 内资源



步骤7 选择“磁盘挂载”，单击“添加挂载”，然后选择“SFS Turbo”。

- 文件系统名称：选择已经创建好的SFS Turbo文件系统。
- 函数访问路径：配置为“/home/test”。
- 共享目录路径：文件系统共享的路径，默认不填写表示函数能访问到文件系统所有目录路径。如果配置具体目录路径，则表示函数只能访问该目录路径。

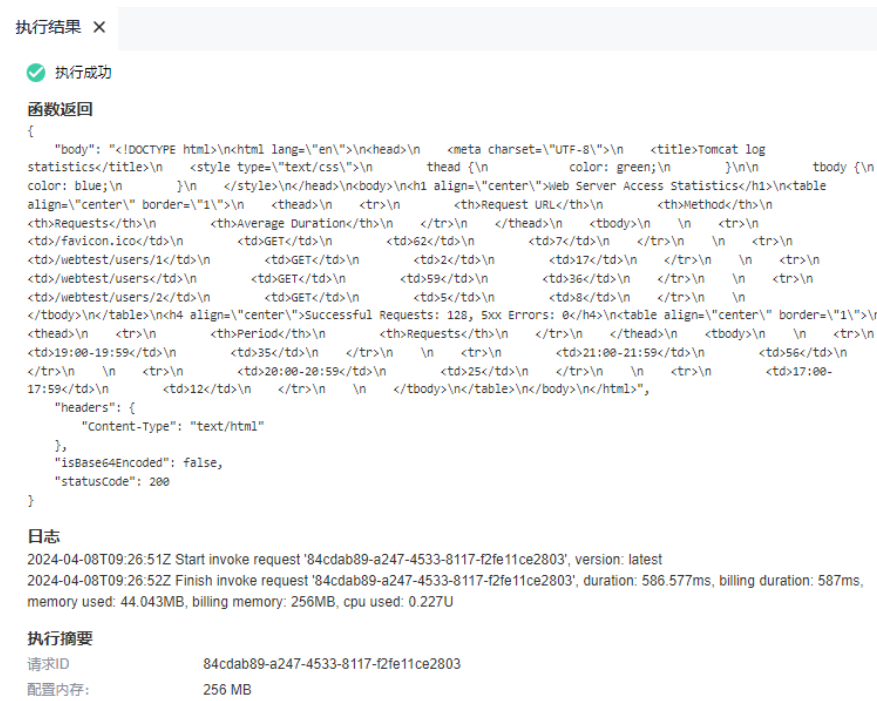
步骤8 选择“代码”，选择“配置测试事件”，然后创建空白模板，最后单击“创建”。

图 4-15 配置测试事件



步骤9 选择刚才创建好的测试事件，单击“测试”。

图 4-16 测试结果



步骤10 选择“设置 > 触发器”，复制APIG的URL，然后用浏览器打开该URL。

图 4-17 复制 URL



图 4-18 打开 URL 结果

Web Server Access Statistics

Request URL	Method	Requests	Average Duration
/favicon.ico	GET	62	7
/webtest/users/1	GET	2	17
/webtest/users	GET	59	36
/webtest/users/2	GET	5	8

Successful Requests: 128, 5xx Errors: 0

Period	Requests
19:00-19:59	35
21:00-21:59	56
20:00-20:59	25
17:00-17:59	12

----结束

ECS 创建 nfs 共享目录

1. Linux系统

- CentOS、SUSE、Euler OS、Fedora或OpenSUSE等系统

i. 配置yum源

①在/etc/yum.repos.d目录下创建文件euleros.repo（文件名可随意取，但是必须以“.repo”结尾）。

②使用如下命令进入euleros.repo编辑配置信息。

```
vi /etc/yum.repos.d/euleros.repo
```

Euler 2.0SP3 yum配置信息如下：

```
[base]
name=EulerOS-2.0SP3 base
baseurl=http://repo.huaweicloud.com/euler/2.3/os/x86_64/
enabled=1
gpgcheck=1
gpgkey=http://repo.huaweicloud.com/euler/2.3/os/RPM-GPG-KEY-EulerOS
```

Euler 2.0SP5 yum配置信息如下：

```
[base]
name=EulerOS-2.0SP5 base
baseurl=http://repo.huaweicloud.com/euler/2.5/os/x86_64/
enabled=1
gpgcheck=1
gpgkey=http://repo.huaweicloud.com/euler/2.5/os/RPM-GPG-KEY-EulerOS
```

说明

参数说明

name: 仓库的名称。

baseurl: 仓库的地址。

- 使用http协议的网络地址: `http://path/to/repo`
- 使用本地仓库地址: `file:///path/to/local/repo`

gpgcheck: 表示是否进行gpg (GNU Private Guard) 校验, 以确定RPM包来源的有效性和安全性。gpgcheck设置为1表示进行gpg校验, 0表示不进行gpg校验。如果没有这一项, 默认是检查的。

③保存配置的repo文件。

④执行如下命令清理缓存。

```
yum clean all
```

ii. 使用如下命令安装nfs-utils

```
yum install nfs-utils
```

iii. 设置共享文件夹

打开/etc/exports, 比如要把/sharedata目录设置为共享目录, 可以填入如下内容:

```
/sharedata 192.168.0.0/24(rw,sync,no_root_squash)
```

说明

上述内容的含义是: 把/sharedata这个目录共享给192.168.0.0/24这个子网段的其他服务器。

命令输入完成后, 可以执行命令`exportfs -v`显示共享的目录, 从而判断是否设置成功。

iv. 使用如下命令启动nfs服务

```
systemctl start rpcbind
service nfs start
```

v. 修改共享目录

比如需要新增/home/myself/download到共享目录, 可以在/etc/exports中新增如下内容。

```
/home/myself/download 192.168.0.0/24(rw,sync,no_root_squash)
```

然后重启nfs服务。

```
service nfs restart
```

或者用如下命令, 无需重启nfs服务。

```
exportfs -rv
```

vi. 设置rpcbind开机启动 (可选)

如果需要设置rpcbind服务开机启动, 可执行如下命令。

```
systemctl enable rpcbind
```

- Ubuntu系统

i. 使用如下命令安装nfs-kernel-server

```
sudo apt-get update
sudo apt install nfs-kernel-server
```

ii. 设置共享文件夹

```
vim /etc/exports
```

打开/etc/exports, 比如要把/sharedata目录设置为共享目录, 可以填入如下内容。

```
/sharedata 192.168.0.0/24(rw,sync,no_root_squash)
```

说明

上述内容的含义是：把/sharedata这个目录共享给192.168.0.0/24这个子网段的其他服务器。

iii. 启动nfs服务

```
service nfs-kernel-server restart
```

说明

命令输入完成后，可以执行命令`exportfs -v`显示共享的目录，从而判断是否设置成功。

iv. 修改共享目录

比如需要新增/home/myself/download到共享目录，可以在/etc/exports中新增如下内容：

```
/home/myself/download 192.168.0.0/24(rw,sync,no_root_squash)
```

然后重启nfs服务

```
service nfs restart
```

或者用如下命令，无需重启nfs服务：

```
exportfs -rv
```

2. Windows系统

1. 安装nfs server软件

目前可用的收费软件有：hanewin nfs server，可到对应[官网](#)下载。

免费的有：FreeNFS、winffsd等，可到[sourceforge](#)上下载。

2. 打开nfs功能

- 如果是winffsd，可参见：<https://github.com/winffsd/winffsd>。
 - i. 下载winffsd并解压，同时在解压后目录中创建一个文件夹“nfs”。
 - ii. 设置“nfs”文件的共享和读写权限。
 - 1) 在“nfs”文件单击右键，然后选择“属性”。
 - 2) 选择“共享”页签，然后单击“共享(S)...”。
 - 3) 添加“Everyone”用户，然后单击“共享(H)”。

图 4-19 添加 Everyone 用户



- 4) 选择“安全”页签，在“组或用户名(G)”列表中选择“Everyone”，然后单击“编辑(E)...”。
 - 5) 在弹出的“安全”设置窗口中，在“组或用户名(G)”列表中选择“Everyone”，在“Everyone的权限(P)”列表中的“允许”勾选框下勾选“读取”和“写入”，最后单击“确定”。
- iii. 关闭所有的防火墙，包括“域网络”、“专用网络”和“公用网络”，等使用完后，再打开防火墙。
 - iv. 进入路由器的虚拟服务器进行端口映射，打开外部网络的端口111、2049及1058。（注意：需要有个外网IP）
 - v. 最后执行如下命令即可，更多的命令解释请参考<https://github.com/winffsd/winffsd>。

```
WinNFSd.exe -addr {你自己的本机IP地址192.168.xxx.xxx} F:\nfs /nfs
```

- 如果是hanewin nfs server，可以参见如下步骤。
 - i. 以Windows系统管理员身份运行下载好的hanewin nfs server的“.exe”文件。
 - ii. 安装完成后，打开“NFS Server”文件，选择“Edit > Preferences”。
 - iii. “NFS”、“Server”和“PortMapper”三个页签下配置保持默认，选择“Exports”页签，单击“Edit exports file”，配置共享目录，配置完成后单击“Save”。

说明

配置共享目录格式可参考“D:\share -public -name:nfs”：将“share”文件夹权限设置为“public”，同时定义了一个别名“nfs”。

- iv. 最后单击“确定”。
- v. 关闭所有的防火墙，包括“域网络”、“专用网络”和“公用网络”，等使用完后，再打开防火墙。

说明

在Linux系统中执行如下命令挂载目录，验证文件共享是否成功。

```
mount -t nfs -o nolock 192.168.xxx.xxx:/nfs /mnt
```

- “192.168.xxx.xxx”是windows系统的IP地址。
- “nfs”是配置共享目录时创建的别名。
- “/mnt”是指定将远程目录挂载到本地的“/mnt”目录下。

4.6 配置环境变量

概述

环境变量可以在不修改代码的情况下，将动态参数传递到函数，调整函数的执行行为。

应用场景

- 区分多环境：相同的函数逻辑，可根据部署环境的不同，配置不同的环境变量以区分。例如，通过环境变量给测试和开发环境配置不同的数据库。
- 配置加密：函数中访问其他服务的认证信息，例如账号和密码，ak/sk，可通过配置加密环境变量，在代码中动态获取，保证敏感数据的安全。
- 动态配置：函数逻辑中需要动态调整的配置，例如查询周期、超时时间，可提取为环境变量避免业务每次变化都需要修改代码。

操作步骤

设置FunctionGraph函数的加密配置和环境变量，无需对代码进行任何更改，可以将设置动态参数传递到函数代码和库。

图 4-20 添加环境变量



例如Node.js语言加密配置和环境变量的值(value)可以通过Context类中的getUserData(string key)获取，详细请参见[Node.js函数开发指南](#)。

警告

- 设置加密配置、环境变量时，用户自定义的键(key)/值(value)，键(key)输入规范：可包含字母、数字、下划线_，以大/小写字母开头。
- 设置“键”和“值”的总长度不超过4096个字符。（当前支持局点：利雅得、汽车一、约翰内斯堡、伊斯坦布尔、乌兰察布一、圣保罗一、香港、新加坡、上海二、圣地亚哥、雅加达、贵阳二零一）
- 设置环境变量时，FunctionGraph会明文展示所有输入信息，请不要输入敏感信息（如账户密码等），以防止信息泄露。
- 打开加密开关之后，界面上会对键值进行加密，参数传输过程中键值也处于加密状态。

预设值

环境变量存在如下预设值，您无法配置和预设值同名的环境变量。

表 4-7 预设值及说明

环境变量名	含义	获取方式和默认值
RUNTIME_PROJECT_ID	函数的项目ID	Context类提供接口或通过系统环境变量获取
RUNTIME_FUNC_NAME	函数名称	Context类提供接口或通过系统环境变量获取
RUNTIME_FUNC_VERSION	函数版本	Context类提供接口或通过系统环境变量获取
RUNTIME_HANDLER	函数执行入口	通过系统环境变量获取
RUNTIME_TIMEOUT	函数执行的超时时间	通过系统环境变量获取
RUNTIME_USERDATA	用户通过环境变量传入的值	Context类提供接口或通过系统环境变量获取
RUNTIME_CPU	函数占用的CPU资源，取值与MemorySize成比例	Context类提供接口或通过系统环境变量获取
RUNTIME_MEMORY	函数配置的内存大小	Context类提供接口或通过系统环境变量获取 单位MB
RUNTIME_MAX_RESP_BODY_SIZE	最大返回值限制	通过系统环境变量获取 系统默认为6291456 Byte
RUNTIME_INITIALIZER_HANDLER	函数初始化入口	通过系统环境变量获取

环境变量名	含义	获取方式和默认值
RUNTIME_INITIALIZER_TIMEOUT	函数初始化超时时间	通过系统环境变量获取
RUNTIME_ROOT	Runtime包的路径	通过系统环境变量获取 系统默认路径为/home/snuser/runtime
RUNTIME_CODE_ROOT	代码在容器中的存放目录	通过系统环境变量获取 系统默认路径为/opt/function/code
RUNTIME_LOG_DIR	系统日志在容器中存放的目录	通过系统环境变量获取 系统默认路径为/home/snuser/log

示例

使用环境变量设置以下信息：安装文件的目录、存储输出的位置、存储连接和日志记录设置等。这些设置与应用程序逻辑解耦，在需要变更设置时，无需更新函数代码。

python语言示例

在如下编写函数代码片段中，参数“obs_output_bucket”为图片处理后存储地址。

```
def handler(event, context):
    srcBucket, srcObjName = getObsObjInfo4OBSTrigger(event)
    obs_address = context.getUserData('obs_address')
    outputBucket = context.getUserData('obs_output_bucket')
    if obs_address is None:
        obs_address = '{obs_address_ip}'
    if outputBucket is None:
        outputBucket = 'casebucket-out'

    ak = context.getAccessKey()
    sk = context.getSecretKey()

    # download file uploaded by user from obs
    GetObject(obs_address, srcBucket, srcObjName, ak, sk)

    outFile = watermark_image(srcObjName)

    # 将转换后的文件上传到新的obs桶中
    PostObject(obs_address, outputBucket, outFile, ak, sk)

    return 'OK'
```

nodejs语言示例

```
exports.handler = async (event, context) => {
    let bucket = context.getUserData('obs_output_bucket');
    console.log(bucket);
    const output =
    {
        'statusCode': 200,
        'headers':
        {
```

```

    'Content-Type': 'application/json'
  },
  'isBase64Encoded': false,
  'body': JSON.stringify(event),
}
return output;
}

```

说明

- 非HTTP函数使用context.getUserData('xxx')方式获取环境变量。
- HTTP函数使用系统方法获取环境变量，比如python函数使用os.Environ['xx']、nodejs函数使用process.env.xx。

通过设置环境变量obs_output_bucket，可以灵活设置存储输出图片的OBS桶。

图 4-21 环境变量



4.7 配置函数异步

概述

函数可以被同步或异步调用，异步调用场景下，FunctionGraph持久化请求后立即返回，不等待请求最终处理完成，用户无法实时感知请求处理结果。如果您希望异步请求处理失败后重试或者希望获取异步处理结果通知，可通过函数异步配置项进行设置。

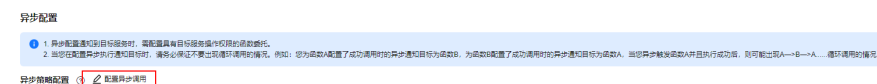
应用场景

- 失败重试：用户代码异常造成的失败，FunctionGraph默认不重试。如果函数中有需要重试的场景，例如调用三方服务经常失败，可配置重试提升成功率。
- 结果通知：FunctionGraph可自动通知异步函数的执行结果给下游服务做进一步处理。例如执行失败信息保存到OBS，后续分析失败原因；执行成功信息推送到DIS或再次触发函数做处理等。

操作步骤

- 步骤1** 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2** 选择待配置的函数，单击进入函数详情页。
- 步骤3** 选择“设置 > 异步配置”，在“异步配置”页签，单击“配置异步调用”。

图 4-22 配置异步调用



- 步骤4** 填写配置参数，参见如下[表4-8](#)，例如设置目标服务为函数工作流（FunctionGraph）。

图 4-23 填写配置参数

异步策略配置

最大重试次数 取值范围: 0-3
异步调用失败后的最大重试次数, 单位: 次

消息最大有效期(s) 取值范围: 1-86,400
消息最大存活时长, 单位: 秒

成功时通知

执行成功时发送通知到以下目标

目标服务

函数名称 [创建函数](#)

版本/别名

失败时通知

执行失败时发送通知到以下目标

目标服务

函数名称 [创建函数](#)

版本/别名

表 4-8 参数说明

参数	说明
异步策略配置	<ul style="list-style-type: none"> 最大重试次数: 异步调用失败后最大重试次数, 默认为1次, 取值范围: 0-3。 消息最大有效期 (s): 消息最大存活时长, 取值范围: 1-86400。
成功时通知	目标服务: 执行成功时发送通知到以下目标服务 <ol style="list-style-type: none"> 函数工作流 (FunctionGraph) 对象存储服务 (OBS) 数据接入服务 (DIS) 消息通知服务 (SMN)
失败时通知	目标服务: 执行失败时发送通知到以下目标服务 <ol style="list-style-type: none"> 函数工作流 (FunctionGraph) 对象存储服务 (OBS) 数据接入服务 (DIS) 消息通知服务 (SMN)

步骤5 填写完成后单击“确定”。

说明

1. 若配置函数异步时，报“用户权限不足”，请添加“FunctionGraph Administrator”权限。具体操作方法请参见[创建用户并授权使用FunctionGraph](#)。
2. 异步配置通知到目标服务时，需配置具有目标服务操作权限的函数委托。
3. 当您在配置异步执行通知目标时，请务必保证不要出现循环调用的情况。例如：您为函数A配置了成功调用时的异步通知目标为函数B，为函数B配置了成功调用时的异步通知目标为函数A，当您异步触发函数A并且执行成功后，则可能出现A→B→A.....循环调用的情况。

---结束

配置说明

异步调用目标的配置说明参见[表4-9](#)，异步调用目标的事件内容参见如下示例：

```
{
  "timestamp": "2020-08-20T12:00:00.000Z+08:00",
  "request_context": {
    "request_id": "1167bf8c-87b0-43ab-8f5f-26b16c64f252",
    "function_urn": "urn:fss:xx-xxxx-x:xxxxxxx:function:xxxx:xxxx:latest",
    "condition": "",
    "approximate_invoke_count": 0
  },
  "request_payload": "",
  "response_context": {
    "status_code": 200,
    "function_error": ""
  },
  "response_payload": "hello world!"
}
```

表 4-9 配置参数说明

参数	说明
timestamp	调用时间戳。
request_context	请求上下文。
request_context.request_id	异步调用的请求ID。
request_context.function_urn	异步执行的函数URN。
request_context.condition	调用错误类别。
request_context.approximate_invoke_count	异步调用的执行次数。当该值大于1时，说明函数计算对您的函数进行了重试。
request_payload	请求函数的原始负载。
response_context	返回上下文。
response_context.statusCode	调用函数的返回码（系统）。当该返回码不为200时，说明出现了系统错误。
response_context.function_error	调用错误信息。
response_payload	执行函数返回的原始负载。

触发函数

须知

配置异步调用的函数的返回值的字节数不能大于256KB，否则会返回空值。

步骤1 按照[操作步骤](#)创建3个函数，分别是test1、test2和test3，test1的异步策略配置如下：

- 成功时通知
 - 目标服务：选择“函数工作流（FunctionGraph）”。
 - 函数名称：选择“test2”。
 - 版本/别名：默认选择“latest”。
- 失效时通知
 - 目标服务：选择“函数工作流（FunctionGraph）”。
 - 函数名称：选择“test3”。
 - 版本/别名：默认选择“latest”。

步骤2 选择“代码”页签，配置测试事件。

选择“空白模板”，将[配置说明](#)中的异步调用目标的示例代码拷贝到代码区，单击“创建”。

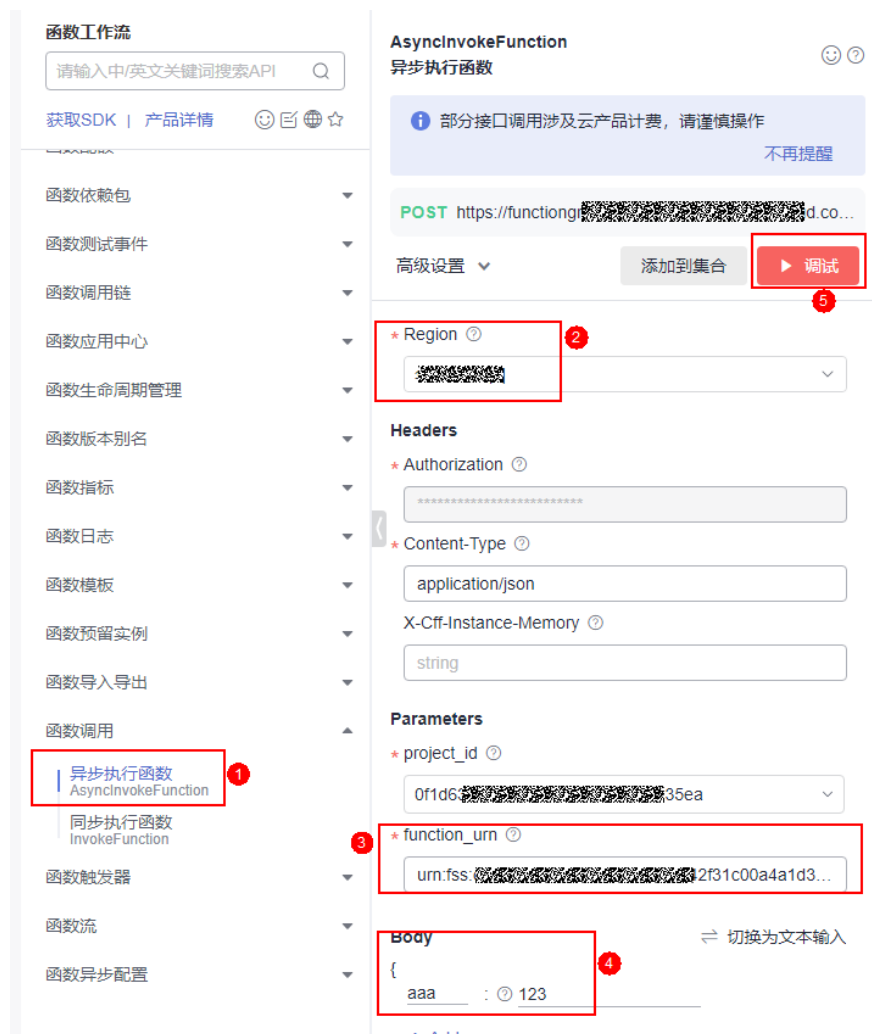
步骤3 函数工作流控制台界面右上角选择“工具 > API Explorer”，进入API Explorer页面，然后在左上角选择“函数工作流”。

步骤4 在左侧API接口列表中选择“函数调用 > 异步执行函数”，参数配置如下：

- Region：请保持与[步骤1](#)中函数创建时选择的区域一致。
- function_urn：函数的URN，此处填写test1的URN地址。
- Body：自定义配置，格式为“string:JSON字符串”，举例“aaa:123”。

其他参数保持默认，单击“调试”。

图 4-24 异步执行函数参数配置



----结束

查看函数异步执行结果

- 步骤1** 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2** 如果在**触发函数**中调试结果执行成功，单击test2函数名称进入详情页；如果调试结果执行失败，单击test3函数名称进入详情页。
- 步骤3** 选择“监控 > 日志”，查询函数test2或test3运行日志。

----结束

4.8 配置单实例多并发

📖 说明

该特性仅FunctionGraph v2版本支持。

前提条件

1. 已配置“单实例并发数”的存量函数，仍支持函数修改该配置，但当“单实例并发数”修改为1时，该参数后续不再支持修改。
2. 通过“容器镜像”和“HTTP函数”方式创建的函数，“单实例并发数”默认值是1，但支持设置多并发数，取值范围1-1000。
3. 如需其他场景使用单实例多并发功能，请联系华为云函数 workflow 服务工程师申请白名单。

概述

默认情况下，每个函数实例同一时刻只处理一个请求，多并发时，例如并发三个请求，FunctionGraph会启动三个函数实例处理请求。FunctionGraph推出的单实例多并发能力，可以让您在一个实例上并发处理多个请求。

应用场景

单实例多并发适合函数处理逻辑中有较长时间等待下游服务响应的场景，也适合函数逻辑中初始化时间较长的场景，具备以下优势：

- 降低冷启动概率，优化函数处理时延：例如并发三个请求，不配置单实例多并发，FunctionGraph默认启动三个实例处理请求，会有三次冷启动。若配置了单实例支持三并发，三个并发请求，FunctionGraph只启动一个实例处理请求，减少了两次冷启动。
- 减少总请求处理时长，节省费用：单实例单并发下，多个请求的总处理时长为每个请求的处理时长相加。单实例多并发下，同一个实例对并发的多个请求的计费时间为，从第一个请求开始处理计时，到最后一个并发的请求处理结束计一次时长费用。

单实例单并发与单实例多并发的对比

当一个函数执行需要花费5秒，若配置为单实例单并发，三次函数调用请求分别在三个函数实例执行，总执行时长为15秒。

若配置为单实例多并发，设置单实例并发数为5，即单个实例最多支持5个并发请求，如果有三次函数调用请求，将在一个实例内并发处理，总执行时间为5秒。

说明

单实例并发数大于1，在您设置的“单函数最大实例数”范围内，超过单实例并发处理能力时会自动扩容新实例。

表 4-10 单并发与多并发对比

对比项	单实例单并发	单实例多并发
日志打印	-	Node.js Runtime使用console.info()函数，Python Runtime使用print()函数，Java Runtime使用System.out.println()函数打印日志，该方式会把当前请求的Request ID包含在日志内容中。当多请求在同一个实例并发处理时，当前请求可能有很多个，继续使用这些函数打印日志会导致Request ID错乱。此时应该使用context.getLogger()，获取一个日志输出对象，通过这个日志输出对象打印日志，例如Python Runtime： log = context.getLogger() log.info("test")
共享变量	不涉及	单实例多并发处理时，修改共享变量会导致错误。这要求您在编写函数时，对于非线程安全的变量修改要进行互斥保护。
监控指标	按实际情况进行监控。	相同负载下，函数的实例数明显减少。
流控错误	不涉及	太多请求时，body中的errorcode为“FSS.0429”，响应头中的status为429，错误信息提示：Your request has been controlled by overload sdk, please retry later。

配置单实例多并发

1. 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 选择“设置 > 并发”，开始配置。
参见[表4-11](#)进行配置，完成后单击“保存”。

图 4-25 并发基础配置

基础配置

单实例并发数 ?

函数支持一个实例同时并发执行多个请求，这个值用来配置单个函数实例可以同时处理多少个请求。

单函数最大实例数

单个函数最多运行在多少个按量实例中，最大值1000；-1表示不限制实例数

表 4-11 参数说明

参数	说明
单实例并发数	<p>单个实例支持的请求并发数，默认值1，取值范围为1-1000。</p> <p>说明 通过“容器镜像”和“HTTP函数”方式创建的函数需要配置该参数，其它方式创建的函数无需配置。</p>
单函数最大实例数	<p>单个函数最多运行在多少个按量实例中，默认值400，最大值为1000；-1表示不限制实例数。</p> <p>说明 超过实例数限制处理能力的请求会被直接丢弃，而不是重试。 当前超过实例数限制导致的请求错误不会直接显示在函数日志中，您可以通过配置函数异步来获取错误详细信息。</p>

配置约束

- 对于Python函数，由于Python GIL锁导致实例上的线程被绑定到一个核上，造成多并发无法使用多核，即使配置更大资源规格也无法提升函数处理性能。
- 对于Node.js函数，由于V8引擎的单进程单线程，造成多并发无法使用多核，即使配置更大资源规格也无法提升函数处理性能。

4.9 版本管理

概述

在函数从开发、测试、生产过程中，可以发布一个或多个版本，实现对函数代码的管理。对于发布的每个版本的函数、环境变量会另存为相应版本的快照，函数代码发布后，您可以根据实际需要修改版本配置信息。

函数创建以后，默认版本为latest版本，每个函数都有一个latest版本。函数代码发布后，您可以根据实际需要修改版本配置信息。

说明

版本相当于函数服务的快照，可对应代码里的tag，函数版本会对应函数的配置、代码等，新版本默认不绑定触发器。当用户新建版本后，对应版本的配置（如环境变量等）、代码等都无法更新，从而保证版本的稳定性、可追溯性等。

发布版本

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 在“版本”页签下，单击“发布新版本”。

图 4-26 发布新版本参数配置

版本号

非必填项，若不指定则默认按时间生成版本号，如v20180116-202722
可包含字母、数字、中划线、下划线和点，长度不超过42个字符，以字母/数字开头和结尾

描述

0/512

- 版本号：您自定义的版本号，用于区分不同的版本。当您未设置时，系统以时间生成版本号，例如：v20220510-190658。
 - 描述：对于版本的描述信息，可以不填。
4. 单击“确定”，系统自动完成版本发布，当前函数版本也会切换至新创建的版本。

📖 说明

- 单个函数最多可以发布20个版本。
- latest版本设置了预留实例，能修改函数配置。新发布的非latest版本默认不带预留实例。
- 基于latest创建的新版本默认不会挂载磁盘，如果不绑定触发器就无法单独设置环境变量。

删除版本

1. 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 在“latest版本”的“版本”页签下，选择需要删除的函数版本。

图 4-27 删除版本



📖 说明

- latest版本不能删除。
 - 如果函数版本关联了别名，则删除版本时会把关联的别名删除。
4. 单击弹框中的“确认”，删除函数版本。

⚠️ 警告

删除版本将永久删除关联的代码、配置、别名及事件源映射，但不会删除日志。删除操作无法恢复，请谨慎操作。

4.10 别名管理

概述

别名指向函数的特定版本，推荐您创建别名并把别名暴露给客户端（例如绑定触发器到别名上而不是某个版本上）。这样，通过修改在别名上配置的版本，可以实现版本的更新和回滚，客户端无感知。一个别名支持配置最多两个版本，在不同的版本上可以分配不同的权重，实现灰度发布。

约束与限制

单个函数最多可以创建10个别名。

创建别名

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 在“别名”页签下，单击“创建别名”。

图 4-28 创建别名

The screenshot shows the 'Create Alias' form with the following fields and options:

- * 别名名称**: Text input field with placeholder '请输入别名名称'. Below it, a note states: '单个函数最多允许创建10个别名。若超过，请删除别名或者重新发布版本后重试。可包含字母、数字、下划线和中划线，长度不超过63个字符。以大/小写字母开头，以字母或数字结尾'.
- * 对应版本**: Dropdown menu with '-请选择-' and a downward arrow.
- 开启灰度版本**: Toggle switch, currently turned on.
- * 灰度版本**: Dropdown menu with '-请选择-' and a downward arrow.
- 灰度类型**: Two buttons: '按百分比随机灰度' (selected) and '按指定规则灰度'.
- 权重**: Range selector with '-' and '+' buttons, currently set to '0~100 %'. Below it, a note states: '您可以按照流量比例随机灰度，您也可以根据请求中的 Header 进行灰度。切流的比例。假设值为 5%，函数工作流会将 5% 的流量打到灰度版本，95% 的流量打到主版本。'
- 描述**: Text area with placeholder '请输入描述' and a character count '0/512'.

- 别名名称：您自定义的别名名称，用于区分不同的别名。
- 对应版本：选择需要关联的版本。
- 开启灰度版本：选择是否开启灰度版本，开启灰度版本后，一个别名可以同时关联两个版本，根据设置的权重比例，函数切换部分主版本的请求到灰度版本运行。

- 灰度版本：选择需要关联的灰度版本，latest版本不能作为灰度版本。
- 灰度类型：有“按百分比随机灰度”和“按指定规则灰度”两种类型。

表 4-12 灰度类型介绍

灰度类型	说明
按百分比随机灰度	选择该类型后需为灰度版本设置“权重”。它为切流的比例，假设值为 5%，则函数工作流会将 5% 的流量打到灰度版本，95% 的流量打到主版本，该值支持输入 0-100 的整数。
按指定规则灰度	该功能目前仅支持 HTTP 函数或 APIG 触发器。选择该类型后需要设置如下参数： <ul style="list-style-type: none"> ● 规则模式：有“满足下列所有规则”和“满足下列任意规则”两种，设置后可以根据“Header”进行灰度。 ● 规则列表：设置“Header”规则条件，详情请参见表 4-13。

表 4-13 规则列表介绍

参数类型	参数	条件	值
Header，默认唯一值。	Header 名称，不区分大小写。	有“=”和“in”两种条件类型。	Header 值，类型是字符串。当条件为“in”时，可以设置多个值，且用“，”隔开，表示满足其中一个值即可触发。

举例：“别名”为“alias1”，“对应版本”为“version1”，“灰度版本”为“version2”，“规则模式”为“满足下列所有规则”，“Header”设置为“aaa”，“条件”设置为“=”，“值”设置为“123”，则表示函数在执行请求时，如果请求的函数别名是“alias1”且请求的“Header”中携带有“aaa”等于“123”这样的参数，那么本次请求会使用该函数的“version2”执行函数，如果请求的“Header”不满足设置的规则条件，则使用该函数的“version1”执行函数。

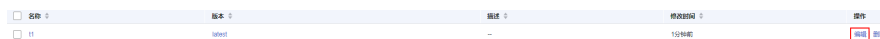
- 描述：对于别名的描述信息。

4. 单击“确定”，完成别名的创建。

修改别名

1. 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 在“latest版本”的“别名”页签下，选择需要修改的函数别名。

图 4-29 修改别名



4. 单击“确定”，完成函数别名修改。

删除别名

1. 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 在“latest版本”的“别名”页签下，选择需要删除的函数别名。

图 4-30 删除别名



4. 单击弹框中的“确认”，删除函数版本。

4.11 配置动态内存

概述

默认情况下，一个函数唯一绑定了一个资源规格。开启动态内存可以让您在处理指定请求时，设置本次处理函数实例使用的资源规格，如果您不指定，函数将使用默认配置的资源规格。

应用场景

以使用函数做视频转码为例：视频文件大小从MB到GB，不同编码格式和分辨率对转码需要的计算资源要求差别很大。为了保证转码性能，通常需要配置一个很大的资源规格，但是在处理低分辨率（例如短视频）视频时，会造成资源浪费。您可以把转码业务实现为元数据获取和转码两个函数，根据元数据信息指定转码函数的资源规格，最小化资源占用，达到更低的成本开销。

前提条件

已创建函数，若未创建，请参见[使用空白模板创建函数](#)。

操作步骤

- 步骤1** 登录FunctionGraph控制台，在左侧导航栏选择“函数 > 函数列表”，单击已创建的函数名称。

图 4-31 选择已创建的函数

函数名称	软件包类型	运行时
diaozhong	Zip	Java 8

4.12 配置心跳函数

概述

心跳函数用于检测用户函数运行时的异常，例如以下场景：

- 用户函数死锁，无法正常运行。
- 用户函数内存溢出，无法正常运行。
- 用户函数网络异常，无法正常运行。

在配置了自定义心跳函数后，当用户函数运行时，FunctionGraph每隔5s向函数实例发送一次心跳请求，触发心跳函数。如果心跳请求返回异常，FunctionGraph会认为函数实例异常，终止此函数实例。

FunctionGraph心跳请求的超时时间是3s，如果连续6次心跳请求未响应，函数实例将被终止。

约束条件

1. 当前心跳函数只支持Java语言。
2. 心跳函数入口需要与函数执行入口在同一文件下。

Java心跳函数格式为：

```
public boolean heartbeat() {
    // 自定义检测逻辑
    return true
}
```

3. 心跳函数目前无输入参数，返回值为bool类型。

操作步骤

步骤1 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 高级设置”，开始配置。

步骤4 开启“配置心跳函数”开关，并填写心跳函数入口。

图 4-34 配置心跳函数



表 4-14 心跳函数配置说明

参数	说明
配置心跳函数	开启心跳函数，FunctionGraph将检测用户函数运行时的异常场景。

参数	说明
心跳函数入口	心跳函数入口需要与函数执行入口在同一文件下。 格式为[包名].[类名].[执行函数名]，不超过128个字符。

步骤5 配置完成后单击“保存”。

----结束

4.13 配置标签

概述

标签用于标识资源，当您拥有相同类型的许多云资源时，可以使用标签按各种维度（例如用途、所有者或环境）对云资源进行分类。

您可以在函数创建完成后，在配置详情页添加标签，最多可以给同一个函数资源添加20个标签。

应用场景

为函数添加标签，可以方便您快速识别和管理拥有的函数资源。例如，您可以为账户中的函数资源定义一组标签，以跟踪每个函数资源的所有者和用途，使函数管理变得更加轻松。

前提条件

确保已开通标签管理服务（TMS），未开通TMS服务时无法使用TMS预定义标签能力。具体请参见[权限管理](#)。

添加标签

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 选择“设置 > 标签”，单击“添加标签”。
4. 参考如下命名规则，添加标签键和标签值。
 - 每个标签由一对键值对（Key-Value）组成，且每个标签键（Key）都必须是唯一的，每个标签键（Key）只能有一个值（Value）。
 - 每个函数最多可以添加20个标签。

说明

如果您的组织已经设定函数 workflow 服务的相关标签策略，则需按照标签策略规则为函数添加标签。标签不符合标签策略的规则，则可能会导致函数创建失败，请联系组织管理员了解标签策略详情。

表 4-15 标签命名规则

参数	规则
标签键	<ul style="list-style-type: none"> 不能为空 不能以_sys_或空格开头，不能以空格结尾 可用UTF-8格式表示的字母（包含中文）、数字和空格，以及以下字符：_ . : = + - @ 128个字符以内且不与其他标签键重复
标签值	<ul style="list-style-type: none"> 可以为空字符串 可用UTF-8格式表示的字母（包含中文）、数字、空格，以及以下字符：_ . : / = + - @ 长度0~255个字符（中文也可以输入255个字符）

- 添加完成后单击“保存”。
保存后的标签键无法修改，标签值可以执行修改操作。

使用标签检索函数

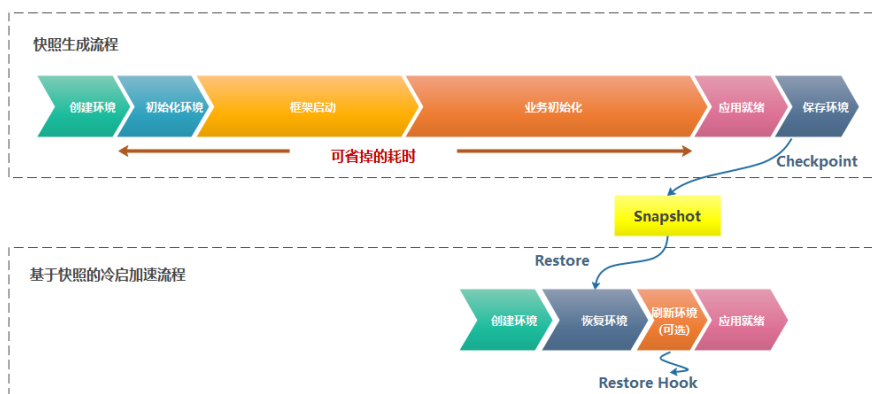
- 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
- 在搜索框中先选择筛选条件“标签”，再勾选标签键值对，一次可添加多个标签作为筛选条件。
- （可选）您可以继续添加筛选条件，例如：运行时、软件包类型等。
- 在函数列表查看检索结果。

4.14 配置快照式冷启动

概述

华为云发布的基于进程级快照的冷启动加速方案，是一种性能优化服务，用户无需额外付费，只需进行简单的配置、少量的代码修改，即可享受到该创新方案带来的冷启动性能提升。

当用户 Java 函数打开冷启动加速的配置开关后，华为云 FunctionGraph 会预先执行函数对应的初始化代码，获取其初始化执行上下文环境的快照，并进行加密缓存。后续调用该函数并触发冷启动扩容时，会**直接从提前初始化后的应用快照来恢复执行环境**，而非重新走一遍初始化流程，以此达到极大提升启动性能的效果。



约束与限制

- 仅支持Java函数。
- 如果应用函数强依赖于有状态，需要考虑使用Restore Hook进行状态刷新。
- 对于强依赖CPU指令集特性的函数，请先提前与客服确认。
- 依赖硬编码的host环境（例如hostname，或者hostip）的函数在迁移到其他主机上的时候，可能会有问题。建议避免依赖这些变量，请先提前与客服确认。
- 当前仅支持基于x86机器开发的应用。

前提条件

- 已创建Java函数。

操作步骤

1. 登录 FunctionGraph 控制台，配置 Java 函数，并打开“快照式冷启动”开关。

图 4-35 开启快照式冷启动



2. （可选）配置 Restore Hook，并在函数代码中实现对应的 Hook 逻辑。

图 4-36 开启 Restore Hook

Restore Hook

* Hook超时时间 (s)

* Hook处理函数

函数代码中Restore Hook示例如下：

```
public class App
{
    public void restore(Context context) {
        System.setProperty("myKey", "restoreValue");
    }

    public void init(Context context) {
        System.setProperty("myKey", "initValue");
    }

    2 usages
    public void print(APIGTriggerEvent event, Context context) {
        System.out.println("value: " + System.getProperty("myKey"));
    }
}
```

- 3. 函数发布新版本后，触发快照的自动化制作。

图 4-37 发布新版本

发布新版本

1 新版本将以LATEST版本内容发布，版本发布后代码和配置将无法编辑。单个函数下您最多可以发布20个版本（含latest版本）。 [了解更多](#)

版本号

非必填项，若不指定则默认按时间生成版本号，如v20180116-202722
可包含字母、数字、中划线、下划线和点，长度不超过42个字符，以字母/数字开头和结尾

快照式冷启动

描述

0/512

- 4. 请耐心等待快照制作完成（5min 超时时间）。

图 4-38 快照制作中

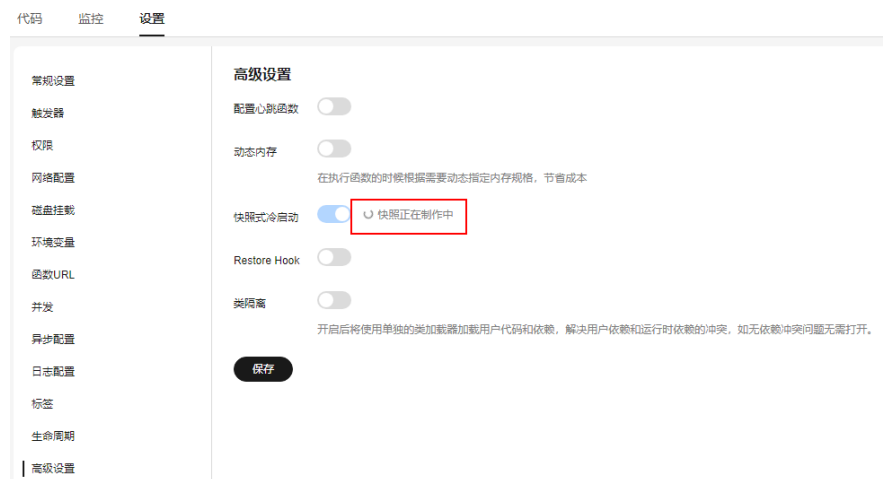
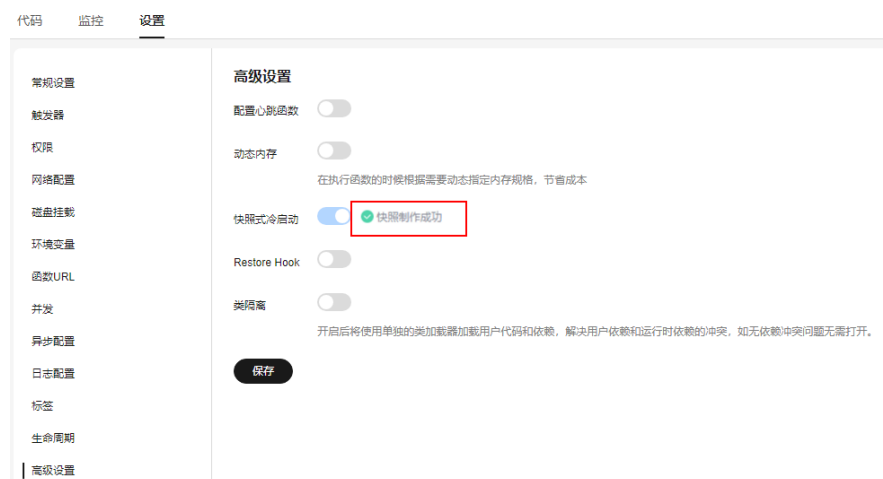
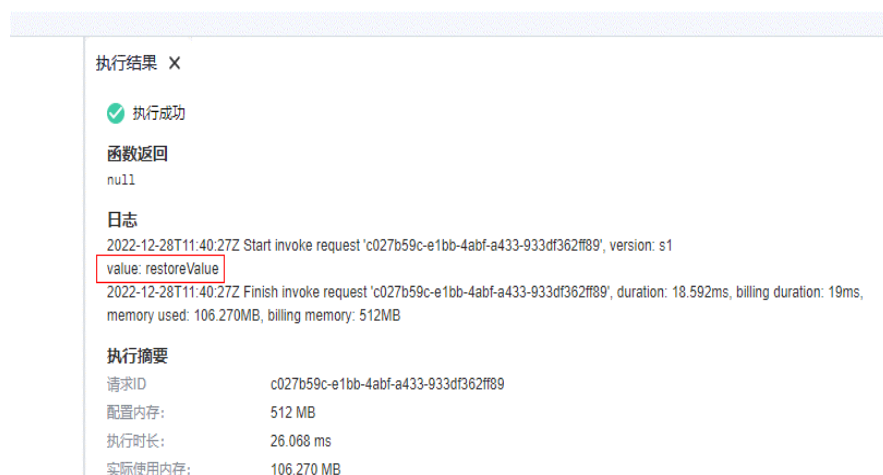


图 4-39 快照制作成功



5. 调用 Java 函数，体验快照优化后的性能提升。



4.15 配置日志组及日志流

说明

此特性当前仅FunctionGraph V2版本支持。

概述

用户可以针对某个函数自行关联日志组和日志流，管理函数日志，即进行函数调用后，调用日志会保存到指定的日志组和日志流下。若未指定，函数调用日志会保持原逻辑自动生成在系统默认创建的日志流下（即创建一个函数默认创建一个对应的日志流），具体请参见[云日志服务（LTS）管理函数日志](#)。

前提条件

已在LTS控制台自行创建日志组及日志流。

操作步骤

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 选择“设置 > 日志配置”，参见[表4-16](#)进行配置。

表 4-16 日志配置参数说明

参数	说明
启动日志记录	V2版本默认为开启状态，V1版本当前不支持此特性。
日志组	为当前函数指定日志组。禁选functiongraph默认创建的日志组functiongraph.log.group.xxx。 默认展示系统自动生成的日志组（以functiongraph开头）。
日志流	指定日志组下的日志流。 默认展示为创建函数时自动生成的日志流（以函数名称开头）。

4. 配置完成后单击“保存”。
5. 调用函数后，您可以参考[查看函数日志](#)，在指定的日志组及日志流下查看生成的函数日志。

说明

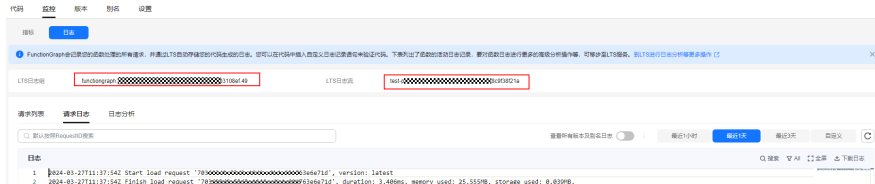
函数的日志流可以在更新时随时修改。

查看函数日志

进行函数调用后，您可以在指定的日志组及日志流下查看生成的函数日志。

1. 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 选择“监控 > 日志”，查看函数日志。
 - 如图4-40，表示当前函数日志指定生成在该日志组及日志流下。
 - 若未指定，此处的日志组显示为系统默认创建的日志组及日志流，日志组名称以“functiongraph”开头。

图 4-40 查看函数日志



- 图中日志列表中可以区分是否请求经过冷启动，下边一条请求显示的冷启动成功，其中耗时13.100ms，表示当前请求是经过冷启动的调用总耗时，日志中上边一条日志显示调用成功，其中耗时1.671ms，表示当前请求不经过冷启，调用耗时。

图 4-41 日志



4.16 有状态函数

4.16.1 简介

状态

由于计算过程最关注的往往是被循环使用的数据，所以狭义的状态可以定义为在计算过程中被循环使用的数据。例如，机器学习中模型训练的大部分算法的设计思想是在构建合适的损失函数后，利用链式法则进行求导以计算出“梯度”，随后使用递归方式循环更新已有的模型参数。这些模型参数是训练过程中需要被循环使用的数据，即一种状态。

在FunctionGraph内核中，状态用来表示在业务处理中跨函数调用的过程数据，其中跨函数调用是指通过API提供的invoke接口进行同一应用内的函数调用。

有状态函数

根据FaaS服务中是否存储状态，可以将函数分为无状态函数和有状态函数（stateful function）两类。有状态函数就是状态存储在FaaS服务中、定义了初始化状态（initstate）方法/属性的函数。

FunctionInstance

有状态函数实例是一个逻辑实例。区别于普通函数实例，有状态函数实例可以持久化运行过程中产生的特定的某个状态数据，并在下一次调用时会将该状态数据进行恢复。

有状态函数实例以InstanceID为标识。InstanceID作为有状态函数实例的逻辑ID，是系统分配的唯一的一个32位ID。

objRef

在FunctionGraph内核中，objRef是一个对象，用来获取函数调用的结果。由于某些函数调用请求尚未结束，需要一个对象来获取这个未处理完成的结果。objRef将其值与其计算（函数调用的执行）分离，从而能够更灵活地进行计算，尤其是更容易表达并行化计算。

4.16.2 开发指导

有状态函数当前支持Java、Node.js、Python运行时。

有状态函数的创建、编辑、触发流程与普通无状态函数相同。主要区别有：

1. 需要在函数配置界面打开有状态函数开关，选择“设置 > 高级设置”，开启“有状态函数”。

图 4-42 有状态函数开关



2. 需要在函数中编写状态初始化代码，具体可参考样例模板。

状态管理

- **函数实例的生成**

创建函数实例调用句柄时，系统会自动生成新的状态实例，并加载到函数中执行。

创建函数实例调用句柄方式，各语言可参照demo模板。以下以java为例：

- a. 通过 `f = new Function (context, functionName)` 不指定函数实例逻辑名称创建。
- b. 通过 `f = new Function (context, functionName, instanceName)` 指定一个全新的函数实例逻辑名称 `instanceName` 值调用有状态函数。

- **函数实例调用句柄的恢复**

通过指定的函数实例逻辑名称 `instanceName` 恢复获取函数实例调用句柄。

```
f = new Function (context)
f.getInstance (functionName, instanceName)
```

- **函数实例的访问**

开发者可以通过 `context.state` 访问当前函数实例绑定的状态数据值。

- **状态的操作**

- 运行函数前，系统会依据函数实例调用句柄所绑定的函数实例ID自动加载状态数据到 `context.state` 中。

- 开发者可以通过`context.state`访问当前函数实例绑定的状态数据值。
- 通过`f.saveState()`接口保存状态数据的修改。
- 当函数执行结束后，如果调用了`f.saveState()`接口，系统会自动持久化状态数据。如果未调用`f.saveState()`接口，则调用结束之后，状态数据不会被保存。

objRef 操作

如何创建objRef?

- 通过`f.invoke()`返回objRef。

如何使用objRef?

- 通过`objRef.get`等待objRef完成并获取objRef对应的值。

4.16.3 注意事项

限制

1. 有状态函数当前仅“亚太-新加坡”支持。
2. 目前有状态函数递归调用次数不能超过1层，即不支持同一个有状态函数的递归调用。
3. 使用Java语言所编写的函数的返回值，只支持JsonObject类型。
4. 单个状态大小限制不允许超过256K，每个租户最多支持创建100个有状态函数实例。
5. 状态数据在执行和持久化过程中不会进行加密，如涉及敏感数据需用户自行加密处理。
6. 有状态函数不支持预留实例调用。
7. `instanceName`支持大小写字母和数字，最短1位，最长128位。
8. 如果函数需要使用VPC功能，则主、被调函数需要绑定同一VPC子网。

Node.js 的特殊 init 方法

Node.js函数有单独的 `init()` 方法，用于初始化生成`instanceId`,

调用有状态函数，必须添加如下操作：

```
const f = new Function(context, functionName, instanceName);
await f.init();
```

4.17 共享 VPC

共享VPC是基于资源访问管理（Resource Access Manager，简称RAM）服务的机制，VPC的所有者可以将VPC内的子网共享给其他账号使用，实现网络资源跨租户共享。在函数中可以配置其他用户共享给您的子网，从而可以在函数中访问该子网下的资源。

如果需要在函数中访问其他用户共享给您的子网，请先确保该子网的拥有者已经正常为您配置了子网共享（如何配置请参考[VPC子网共享给其他账号](#)），然后在函数的网络配置中选择共享的子网即可，具体操作请参考[访问VPC](#)。如果后续VPC子网拥有者取消了共享，则您将无法在函数中访问该子网。

有关VPC子网共享的更多信息，请参见《虚拟私有云用户指南》的“[共享VPC](#)”相关内容。

5 在线调试

注意事项

事件数据作为event参数传入入口函数，配置后保存可以持久化，以便下次测试使用。每个函数最多可配置10个测试事件。

创建测试事件

- 步骤1** 登录FunctionGraph控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数页面。
- 步骤2** 单击函数名称，进入函数详情界面。
- 步骤3** 在函数详情页，选择函数版本，单击“测试”，弹出“配置测试事件”页。
- 步骤4** 在“配置测试事件”界面填写测试信息，如表5-1所示，带*参数为必填项。

表 5-1 测试信息

参数	说明
配置测试事件	可创建新的测试事件也可编辑已有的测试事件。 默认值为：“创建新的测试事件”。
事件模板	使用空白模板需要编辑测试事件。 使用已有模板会自动加载相对应的测试事件，事件模板说明如表5-2所示。
*事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如even-123test。
测试事件	输入测试事件。

表 5-2 事件模板说明

模板名称	模板说明
API 网关服务 (APIG)	模拟APIG事件, 触发函数。
API 网关服务 (APIG专享版)	模拟APIG (专享版) 事件, 触发函数。
云审计服务 (CTS)	模拟CTS事件, 触发函数。
文档数据库服务 (DDS)	模拟DDS事件, 触发函数。
云数据库 GeminiDB Mongo	模拟GeminiDB Mongo事件, 触发函数。
数据接入服务 (DIS)	模拟DIS事件, 触发函数。
云日志服务 (LTS)	模拟LTS事件, 触发函数。
对象存储服务 (OBS)	模拟OBS事件, 触发函数。
消息通知服务 (SMN)	模拟SMN事件, 触发函数。
定时触发器 (TIMER)	模拟TIMER事件, 触发函数。
分布式消息服务 Kafka版 (KAFKA)	模拟Kafka事件, 触发函数。
开源Kafka (OPENSOURCEKAFKA)	模拟开源Kafka事件, 触发函数。
分布式消息服务 RabbitMQ版 (RABBITMQ)	模拟RabbitMQ事件, 触发函数。
空白模板	模板事件为: {"key": "value"}, 可以根据需要修改。
登录安全实时分析	可以作为“登录安全实时分析”函数模板的输入。
图片分类	可以作为“实时图片分类 (按图片内容)”等函数模板的输入。
图片鉴黄	可以作为“图片鉴黄”函数模板的输入。
语音识别	可以作为“语音识别”函数模板的输入。

步骤5 单击“保存”，完成测试事件创建。

----结束

测试函数

函数创建以后, 可以在线测试函数能否正常运行, 验证能否实现预期功能。

步骤1 返回函数 workflow 控制台, 在左侧的导航栏选择“函数 > 函数列表”。

步骤2 单击函数名称, 进入函数详情界面。

步骤3 在函数详情页，选择函数版本，选择测试事件，单击“测试”。

图 5-1 选择测试事件



步骤4 单击“测试”，可以得到函数运行结果。

📖 说明

“日志”页签最多显示2K日志，如需查看完整日志，请参见[管理函数日志](#)的操作。

----结束

修改测试事件

步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 单击函数名称，进入函数详情界面。

步骤3 在函数详情页，选择函数版本，单击“配置测试事件”，弹出“配置测试事件”页。

步骤4 在“配置测试事件”界面修改测试信息，如[表5-3](#)所示。

表 5-3 测试信息

参数	说明
创建新的测试事件	重新创建新的测试事件。
编辑已有测试事件	修改已有的测试事件。
测试事件	修改测试事件代码。

步骤5 单击“保存”，完成配置修改。

----结束

删除测试事件

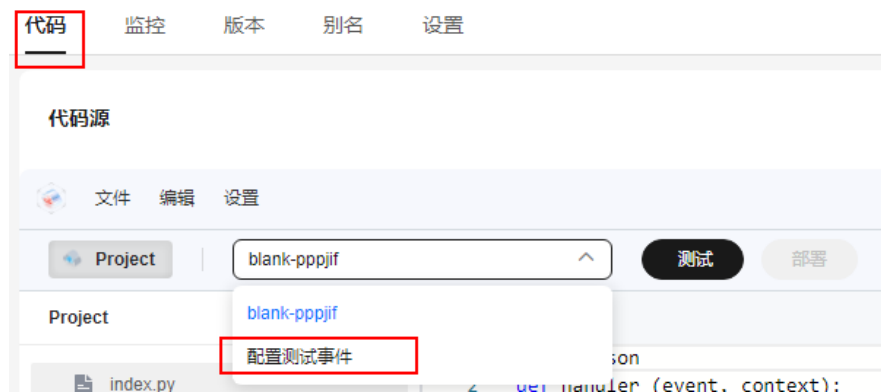
- 步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2 单击函数名称，进入函数详情界面。
- 步骤3 在函数详情页，选择函数版本，如图5-2所示。

图 5-2 选择函数版本



- 步骤4 在“代码”页签中选择“配置测试事件”进入编辑界面，如图5-3所示。

图 5-3 选择配置测试事件



- 步骤5 在“配置测试事件”的编辑界面中，选择“编辑已有测试事件”，然后在左侧“已保存测试事件”列表中选中待删除事件名称，单击“删除”。

图 5-4 删除测试事件

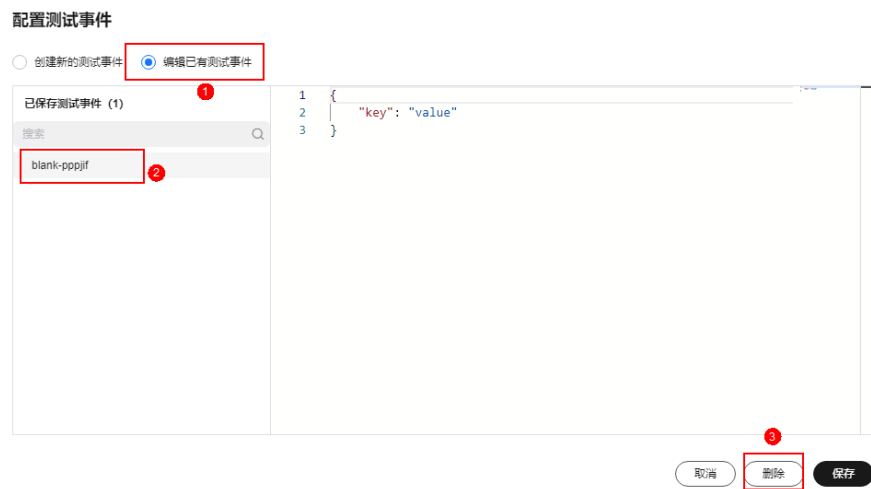


表 5-4 配置测试事件信息

参数	说明
创建新的测试事件	选择已提供的测试事件。
编辑已有测试事件	选择需要删除的测试事件。

----结束

6 配置触发器

6.1 触发器管理

停用/启用触发器

已经创建的触发器，通过设置停用/启用，控制触发器的状态。**SMN触发器、APIG触发器创建以后，不能停用，只能删除。**

步骤1 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 单击函数名称，进入函数详情界面。

步骤3 选择“设置 > 触发器”，进入“触发器”页签，在需要停用/启用的触发器所在行，单击“停用”/“启用”，停用/启用触发器。

----结束

删除触发器

已经创建的触发器，如果不再使用，可以执行删除操作。

步骤1 返回函数工作流控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 单击函数名称，进入函数详情界面。

步骤3 单击“触发器”，进入“触发器”页签，在需要删除的触发器所在行，单击“删除”，删除触发器。

----结束

📖 说明

在触发器列表页面中，触发器类型页签会优先展示用户当前使用的触发器。

图 6-1 触发器展示



6.2 使用定时触发器

本节介绍创建定时触发器，按照设置的频率，定期触发函数运行，供用户了解定时触发器的使用方法。

关于定时触发器事件源具体介绍请参见[支持的事件源](#)。

前提条件

已经创建函数，创建过程请参见[创建函数](#)。

创建定时触发器

步骤1 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-2 创建触发器



步骤4 设置以下信息。

- 触发器类型：选择“定时触发器 (TIMER)”。
- 定时器名称：您自定义的定时器名称，例如：Timer。
- 触发规则：固定频率和Cron表达式。
 - 固定频率：固定时间间隔触发函数，该类型下支持配置单位为分、时、天，每种类型仅支持整数配置，其中分钟支持范围(0, 60]，小时支持范围(0, 24]，天支持范围(0, 30]。
 - Cron表达式：设置更为复杂的函数执行计划，例如：周一到周五上午 08:30:00执行函数等，具体请参见[函数定时触发器Cron表达式规则](#)。
- 是否开启：是否开启定时触发器。
- 附加信息：如果用户配置了触发事件，会将该事件填写到TIMER事件源的“user_event”字段，详情请参见[支持的事件源](#)。

步骤5 单击“确定”，完成定时触发器的创建。

----结束

查看函数运行结果

函数的定时触发器创建以后，每隔一分钟执行一次函数，可以查看函数运行日志。

步骤1 返回函数工作流控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择函数，单击进入函数详情页。

步骤3 选择“监控 > 日志”，查询函数运行日志。

----结束

6.3 使用 APIG（专享版）触发器

本节介绍创建APIG触发器，使用API调用函数运行。供用户了解APIG触发器的使用方法。

关于APIG触发器事件源具体介绍请参见[支持的事件源](#)。

前提条件

已经创建API分组，此处以APIGroup_test分组为例，创建过程请参见[创建API分组](#)。

创建 APIG 触发器

步骤1 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 单击右上方的“创建函数”，进入“创建函数”页面。

步骤3 设置以下函数信息。

- 函数名称：输入您自定义的函数名称，例如：apig。
- 委托名称：选择“不使用任何委托”。
- 企业项目：选择“default”。
- 运行时语言：选择“Python 2.7”。

步骤4 单击“创建函数”，完成函数的创建。

步骤5 在“代码”页签下，复制如下代码至代码窗并单击“部署”。

```
# -*- coding:utf-8 -*-
import json
def handler(event, context):
    body = "<html><title>Functiongraph Demo</title><body><p>Hello, FunctionGraph!</p></body></html>"
    print(body)
    return {
        "statusCode":200,
        "body":body,
        "headers": {
            "Content-Type": "text/html",
        },
        "isBase64Encoded": False
    }
```

步骤6 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-3 创建触发器



步骤7 设置以下触发器信息。

表 6-1 触发器信息

字段	填写说明
触发器类型	选择“API网关服务（APIG专享版）”。

字段	填写说明
实例	选择所属实例，若无实例，可单击“创建实例”完成创建。
API名称	您自定义的API名称，例如：API_apig。
分组	API分组相当于一个API集合，API提供方以API分组为单位，管理分组内的所有API。 选择“APIGroup_test”。
发布环境	API可以同时提供给不同的场景调用，如生产、测试或开发。API网关服务提供环境管理，在不同的环境定义不同的API调用路径。 选择“RELEASE”，才能调用。
安全认证	API认证方式： <ul style="list-style-type: none"> App：采用Appkey&Appsecret认证，安全级别高，推荐使用，详情请参见APP认证。 IAM：IAM认证，只允许IAM用户能访问，安全级别中等，详情请参见IAM认证。 None：无认证模式，所有用户均可访问。 选择“None”。
请求协议	分为两种类型： <ul style="list-style-type: none"> HTTP HTTPS 选择“HTTPS”。
后端超时(毫秒)	输入“5000”。

步骤8 单击“确定”，完成触发器的创建。

图 6-4 创建触发器



说明

1. “调用URL”即APIG触发器调用地址。
2. API触发器创建完成后，会在API网关生成名为API_apig的API，单击API名称，跳转至API网关服务。

----**结束**

调用函数

步骤1 在浏览器地址栏输入APIG触发器的调用地址URL，按“Enter”。

步骤2 函数执行完毕，得到返回结果，如图6-5所示。

图 6-5 返回结果



说明

1. FunctionGraph函数对APIG调用的传入值为函数自带的事件模板，您可以参见表5-2。
2. FunctionGraph函数对来自APIG调用的返回结果进行了封装，APIG触发器要求函数的返回结果中必须包含body(String)、statusCode(int)、headers(Map)和isBase64Encoded(boolean)，才可以正确返回。

----结束

查看函数运行结果

步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择函数，单击进入函数详情页。

步骤3 选择“监控 > 日志”，查询函数运行日志。

----结束

6.4 使用 Kafka 触发器

本节介绍创建Kafka触发器，供用户了解Kafka触发器的使用方法。

使用Kafka触发器后，FunctionGraph会定期轮询Kafka实例指定Topic下的新消息，FunctionGraph将轮询得到的消息作为参数传递来调用函数，关于Kafka触发器的事件源介绍请参见[支持的事件源](#)。

说明

- 分布式消息服务Kafka版与开源Kafka的差异说明，请参见[Kafka与开源Kafka的差异](#)。
- 对于处理失败的Kafka数据，Kafka触发器会丢弃过大记录（>6MB）。

前提条件

进行操作之前，需要做好以下准备。

- 已经创建函数，创建过程请参见[使用空白模板创建函数](#)。
- 创建Kafka触发器，必须开启函数 workflow VPC访问，请参见[配置网络](#)。
- 已经创建Kafka实例，创建操作请参见[购买Kafka专享版实例](#)。
- 在Kafka实例下创建主题，创建操作请参见[Kafka实例创建Topic](#)。

创建 Kafka 触发器

- 步骤1** 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2** 选择待配置的函数，单击进入函数详情页。
- 步骤3** 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-6 创建触发器

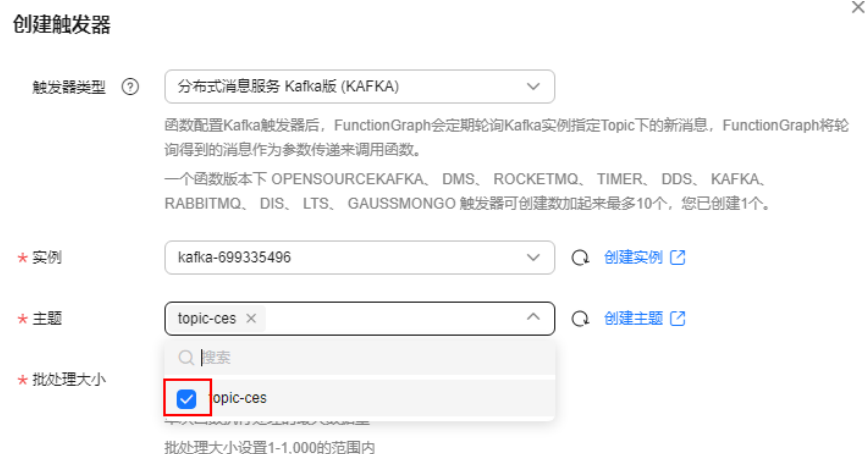


- 步骤4** 设置以下信息。
- 触发器类型：选择“分布式消息服务（Kafka）”。
 - 实例：选择已创建专享版Kafka实例。
 - 主题：选择专享版Kafka实例的Topic。
 - 批处理大小：每次从Topic消费的消息数量。
 - 用户名：Kafka实例开启SSL时需要填写。连接Kafka专享版实例的用户名。
 - 密码：Kafka实例开启SSL时需要填写。连接Kafka专享版实例的密码。
- 步骤5** 单击“确定”，完成kafka触发器的创建。

说明

- 开启函数流VPC访问后，需要在Kafka服务安全组配置对应子网的权限。如何开启VPC访问请参见[配置网络](#)。
- Kafka触发器当前支持选择多个Topic主题，从而避免Topic过多导致创建的触发器数量被限制。

图 6-7 支持多 Topic 选择



----结束

配置 Kafka 事件触发函数。

- 步骤1** 返回函数工作流控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 在函数详情页，选择函数版本。

步骤4 在“代码”页签下，单击“测试”，弹出“配置测试事件”对话框。

步骤5 填写如表6-2所示测试信息后，单击“保存”。

表 6-2 测试信息

参数	说明
配置测试事件	可创建新的测试事件也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择“分布式消息服务 Kafka版（KAFKA）”模板，使用系统内置Kafka事件模板。
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如kafka-123test。
测试事件	自动加载系统内置kafka事件模板，本例不做修改。

步骤6 单击“测试”，可以得到函数运行结果，函数会返回输入kafka消息数据。

----结束

6.5 使用 DIS 触发器

本节介绍创建DIS触发器，使用系统内置的事件模板配置dis事件，触发函数运行。供用户了解DIS触发器的使用方法。

关于DIS触发器事件源具体介绍请参见[支持的事件源](#)。

前提条件

进行操作之前，需要做好以下准备。

- 已经创建函数，创建过程请参见[创建函数](#)。
- 已经创建接入通道，此处以dis-function为例，创建过程请参见[创建DIS通道](#)。

设置函数委托

创建DIS触发器时，需要设置函数委托，委托权限需要包括DIS，委托的创建请参见[配置委托权限](#)。

由于创建HelloWorld函数的时候没有设置委托，所以需要先修改函数委托。

步骤1 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 在“设置 > 权限”页签，修改函数委托，将委托修改为[配置委托权限](#)创建的serverless-trust委托。

步骤4 单击“保存”，完成委托修改。

----结束

创建 DIS 触发器

步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

步骤4 设置以下信息。

- 触发器类型：数据接入服务 (DIS)。
- 通道名称：选择已创建的 DIS 通道，例如：dis-function。
- 起始位置：选择流中开始读取数据的位置。
 - TRIM_HORIZON：从最早被存储至分区的有效记录开始读取。
 - latest：从分区中的最新记录开始读取，此设置可以保证你总是读到分区中最新记录。
- 数据处理方式：“按字节数”和“按批处理”。

表 6-3 数据处理方式介绍

数据处理方式	说明
按字节数	需配置“最大字节数”，每次触发时获取记录的最大字节数。只有当分区中单条记录小于该值，才能被获取。设置范围：0KB-4MB。
按批处理	需配置“批处理大小”，单次拉取的最大数据条数。只有当分区中单条记录小于该值，才能被获取。设置范围：1-10000，同时拉取的数据条数总大小不超过4MB。

📖 说明

用户默认没有权限使用“按批处理”方式，如果需要使用该功能，请在[工单系统](#)提交工单添加白名单。

- 拉取周期：设置拉取流数据的周期。
- 串行处理数据：如果开启该选项，取一次数据处理完之后才会取下一次数据；否则只要拉取周期到了就会取数据进行处理。

说明

关闭“串行处理数据”开关后，您可以根据业务需要配置并发数（范围：1-80）。该参数的功能是：当DIS触发器配置为异步执行时，它可以控制DIS触发器异步调用函数的并发数，防止单个触发器流量较大导致单租户并发跑满，进而影响其他DIS触发器无法执行。（目前暂时只有北京四支持该功能）

图 6-8 关闭“串行处理数据”

创建触发器

触发器类型 ?

可以将FunctionGraph函数配置为自动轮询流并处理任何新记录，例如网站点击流、财务交易记录、社交媒体源、IT日志和位置跟踪事件等。
一个函数版本下 OPENSOURCEKAFKA、DMS、ROCKETMQ、TIMER、DDS、KAFKA、RABBITMQ、LTS、GAUSSMONGO、DIS 触发器可创建数加起来最多10个，您已创建1个。

* 通道名称 [申请接入通道](#)

* 起始位置

TRIM_HORIZON：从最早被存储至分区的有效记录开始读取。LATEST：从分区中的最新记录开始读取，此设置可以保证总是读到分区中最新记录。

* 最大字节数 ?

最大字节数设置0MB-2MB的范围内

* 拉取周期

拉取流数据的周期
拉取周期设置1s-60s的范围内

串行处理数据

如果开启该选项，取一次数据处理完之后才会取下一次数据；否则只要拉取周期到了就会取数据进行处理。

并发数

该触发器触发函数执行的最大并发数，即并行处理数据时可同时调用函数的最大次数

步骤5 单击“确定”，完成DIS触发器的创建。

----结束

修改 DIS 触发器配置

DIS触发器创建后，支持对部分参数进行修改。

步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击DIS触发器上的“编辑”，弹出“创建触发器”对话框

步骤4 您可以修改“最大字节数/批处理大小”、“拉取周期”、“串行处理数据”，修改完成后单击“确定”。

----结束

配置 DIS 事件触发函数

- 步骤1** 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2** 选择待配置的函数，单击进入函数详情页。
- 步骤3** 在函数详情页，选择函数版本。
- 步骤4** 在“代码”页签下，单击“测试”，弹出“配置测试事件”对话框。
- 步骤5** 填写如表6-4所示测试信息后，单击“保存”。

表 6-4 测试信息

参数	说明
配置测试事件	可创建新的测试事件也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择“数据接入服务（DIS）”模板，使用系统内置dis事件模板。
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符。例如输入dis-123test。
测试事件	自动加载系统内置dis事件模板，本例不做修改。

- 步骤6** 单击“测试”，可以得到函数运行结果，函数会返回输入DIS数据。

----结束

6.6 使用 SMN 触发器

本节介绍创建SMN触发器，发布消息，触发函数运行，供用户了解SMN触发器的使用方法。

关于SMN触发器事件源具体介绍请参见[支持的事件源](#)。

前提条件

- 已经创建SMN消息主题，此处以smn-test为例，创建过程请参见[创建消息主题](#)。
- 已经创建函数，创建过程请参见[创建函数](#)。

创建 SMN 触发器

- 步骤1** 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2** 选择待配置的函数，单击进入函数详情页。
- 步骤3** 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-9 创建触发器



步骤4 设置以下信息。

- 触发器类型：选择“消息通知服务 (SMN)”。
- 主题名称：选择主题名称，例如：smn-test。

步骤5 单击“确定”，完成SMN触发器的创建。

📖 说明

SMN触发器创建完成后，会在SMN消息主题生成消息订阅。

----结束

触发函数运行

在“消息通知服务”控制台，为“smn-test”主题发布消息，具体操作步骤请参见[向主题发布文本消息](#)。

发布消息的内容参见[表6-5](#)填写。

表 6-5 发布消息

字段	填写说明
消息标题	输入“SMN-Test”。
消息类型	选择“文本消息”。
消息内容	输入以下内容：{"message":"hello"}。

📖 说明

消息发布以后，会自动触发函数运行，具体示例事件请参见[支持的事件源](#)。

查看函数运行结果

步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择函数，单击进入函数详情页。

步骤3 选择“监控 > 日志”，查询函数运行日志。

----结束

6.7 使用 LTS 触发器

本节介绍创建LTS触发器，供用户了解LTS触发器的使用方法。

关于定时触发器事件源具体介绍请参见[支持的事件源](#)。

前提条件

- 已经创建函数，创建过程请参见[创建函数](#)。
- 已经创建LTS FullAccess权限的委托，创建过程请参见[配置委托权限](#)。

- 已经创建日志组，此处以LogGroup1为例，创建过程请参见[创建日志组](#)。
- 已经创建日志流，此处以LogTopic1为例，创建过程请参见[创建日志流](#)。
- 配置Agent，快速将ECS等服务器上日志采集到指定的日志组，详情请参见[安装ICAgent](#)。

创建 LTS 触发器

步骤1 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-10 创建触发器



步骤4 设置以下信息。

- 触发器类型：选择“云日志服务（LTS）”。
- 日志组：选择已创建的日志组，例如：LogGroup1。
- 日志流：选择已创建的日志流，例如：LogStream1。

步骤5 单击“确定”，完成LTS触发器的创建。

---结束

配置 LTS 事件触发函数

说明

当原始LTS事件消息超过75KB，会把原始LTS事件消息按照75KB维度拆分为多条消息触发执行函数。

步骤1 返回函数工作流控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 在函数详情页，选择函数版本。

步骤4 在“代码”页签下，单击“测试”，弹出“配置测试事件”对话框。

步骤5 填写如[表6-6](#)所示测试信息后，单击“保存”。

表 6-6 测试信息

参数	说明
配置测试事件	可创建新的测试事件也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择“云日志服务（LTS）”模板，使用系统内置LTS事件模板。

参数	说明
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如lts-123test。
测试事件	自动加载系统内置lts事件模板，本例不做修改。

步骤6 单击“测试”，可以得到函数运行结果，函数会返回输入LTS数据。

----结束

6.8 使用 CTS 触发器

本节介绍创建CTS触发器，通过增加自定义操作，触发函数运行，通过CTS云审计服务获取操作记录，供用户了解CTS触发器的使用方法。

关于CTS触发器事件源具体介绍请参见[支持的事件源](#)。

前提条件

已经在统一身份认证创建委托，创建过程请参见[配置委托权限](#)。

创建 CTS 触发器

步骤1 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 单击右上方的“创建函数”，进入“创建函数”页面。

步骤3 设置以下函数信息。

- 函数名称：输入您自定义的函数名称，例如：HelloWorld。
- 委托名称：选择“不使用任何委托”。
- 企业项目：选择“default”。
- 运行时语言：选择“Python 2.7”。

步骤4 单击“创建函数”，完成函数的创建。

步骤5 在“代码”页签下，复制如下代码至代码窗并单击“部署”。

```
# -*- coding:utf-8 -*-
"""
CTS trigger event:
{
  "cts": {
    "time": "",
    "user": {
      "name": "userName",
      "id": "",
      "domain": {
        "name": "domainName",
        "id": ""
      }
    }
  },
  "request": {},
  "response": {},
  "code": 204,
  "service_type": "FunctionGraph",
}
```

```

"resource_type": "",
"resource_name": "",
"resource_id": {},
"trace_name": "",
"trace_type": "ConsoleAction",
"record_time": "",
"trace_id": "",
"trace_status": "normal"
}
}
"""
def handler(event, context):
    trace_name = event["cts"]["resource_name"]
    timeinfo = event["cts"]["time"]
    print(timeinfo+' '+trace_name)

```

步骤6 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-11 创建触发器



步骤7 设置以下触发器信息。

表 6-7 触发器信息

字段	填写说明
触发器类型	选择“云审计服务（CTS）”。
通知名称	输入您自定义的通知名称，例如：Test。
服务类型	选择“FunctionGraph”。 说明 当服务类型选择全局级云服务时，例如对象存储服务（OBS）、统一身份认证服务（IAM）等，CTS触发器只能在国际站的香港局点才能触发。了解更多全局级云服务请咨询技术支持工程师。
资源类型	所选服务下对应的资源类型，如触发器、实例、函数等。
操作名称	所选资源类型下对应的操作，如创建、删除触发器等。

步骤8 单击“确定”，完成CTS触发器的创建。

----结束

配置 CTS 事件触发函数

步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 在HelloWorld函数详情页，选择函数版本，单击“测试”，弹出“配置测试事件”对话框。

步骤4 填写如表6-8所示测试信息后，单击“保存”。

表 6-8 测试信息

参数	说明
配置测试事件	可创建新的测试事件也可编辑已有的测试事件。 选择“创建新的测试事件”。
事件模板	选择“云审计服务（CTS）”模板，使用系统内置CTS事件模板。
事件名称	您自定义的事件名称，例如：cts-test。
测试事件	自动加载系统内置CTS事件模板，您可以根据实际情况修改。

步骤5 单击“测试”，可以得到函数运行结果记录。

----结束

6.9 使用 DDS 触发器

本节介绍创建DDS触发器，供用户了解DDS触发器的使用方法。

使用DDS触发器，每次更新数据库中的表时，都可以触发FunctionGraph函数以执行额外的工作，关于DDS触发器事件源具体介绍请参见[支持的事件源](#)。

前提条件

进行操作之前，需要做好以下准备。

- 已经创建函数，创建过程请参见[创建函数](#)。
- 创建DDS触发器，必须开启函数工作流VPC访问，请参见[配置网络](#)。
- 已经创建DDS文档数据库实例。
- 已经创建DDS文档数据库。

创建 DDS 触发器

步骤1 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-12 创建触发器



步骤4 设置以下信息。

- 触发器类型：选择“文件数据库服务（DDS）”。
- 文档数据库实例：选择已创建的DDS实例。
- 密码：DDS数据库实例管理员rwuser的密码。
- 数据库名称：输入DDS实例数据库名称。admin、local、config为保留数据库，不能使用。

- 集合名称：数据库集合名称。
- 批处理大小：每批从数据库读取的记录的数量。

步骤5 单击“确定”，完成DDS触发器的创建。

📖 说明

开启函数流VPC访问后，需要在DDS服务安全组配置对应子网的权限。如何开启VPC访问请参见[配置网络](#)。

----结束

配置 DDS 事件触发函数

步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 在HelloWorld函数详情页，选择函数版本，单击“测试”，弹出“配置测试事件”对话框。

步骤4 填写如[表6-9](#)所示测试信息后，单击“保存”。

表 6-9 测试信息

参数	说明
配置测试事件	可创建新的测试事件也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择“文档数据库服务（DDS）”模板，使用系统内置dds事件模板。
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如dds-123test。
测试事件	自动加载系统内置dds事件模板，本例不做修改。

步骤5 单击“测试”，可以得到函数运行结果，函数会返回输入DDS数据。

----结束

6.10 使用 GeminiDB Mongo 触发器

本节介绍创建GeminiDB Mongo触发器，供用户了解GeminiDB Mongo触发器的使用方法。

使用GeminiDB Mongo触发器，每次更新数据库中的表时，都可以触发FunctionGraph函数以执行额外的工作，关于GeminiDB Mongo触发器事件源具体介绍请参见[支持的事件源](#)。

前提条件

进行操作之前，需要做好以下准备。

- 已经创建函数，创建过程请参见[创建函数](#)。
- 创建GeminiDB Mongo触发器，必须开启函数工作流VPC访问，请参见[配置网络](#)。
- 已经创建GeminiDB Mongo云数据库实例，创建过程请参见[云数据库GeminiDB实例](#)。

创建 GeminiDB 触发器

步骤1 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-13 创建触发器



步骤4 设置以下信息。

- 触发器类型：选择“云数据库 GeminiDB Mongo”。
- GeminiDB Mongo：选择已创建的GeminiDB实例。
- 密码：GeminiDB数据库实例管理员rwuser的密码。
- 数据库：输入GeminiDB Mongo实例数据库名称。admin、local、config为保留数据库，不能使用。
- 集合：数据库集合名称。
- 批处理大小：每批从数据库读取的记录的数量。

步骤5 单击“确定”，完成GeminiDB触发器的创建。

📖 说明

开启函数流VPC访问后，需要在GeminiDB Mongo服务安全组配置对应子网的权限。如何开启VPC访问请参见[配置网络](#)。

----结束

配置 GeminiDB 事件触发函数

步骤1 返回函数工作流控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 在函数详情页，选择函数版本。

步骤4 在“代码”页签下，单击“测试”，弹出“配置测试事件”对话框。

步骤5 填写如[表6-10](#)所示测试信息后，单击“保存”。

表 6-10 测试信息

参数	说明
配置测试事件	可创建新的测试事件也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择“云数据库 GeminiDB Mongo”模板，使用系统内置 GeminiDB Mongo 事件模板。
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如 geminimongo-123test。
测试事件	自动加载系统内置 GeminiDB Mongo 事件模板，本例不做修改。

步骤6 单击“测试”，可以得到函数运行结果，函数会返回输入 GeminiDB Mongo 数据。

----结束

6.11 使用 APIG 触发器

本节介绍创建 APIG 触发器，使用 API 调用函数运行。供用户了解 APIG 触发器的使用方法。

关于 APIG 触发器事件源具体介绍请参见[支持的事件源](#)。

📖 说明

首次使用 API 网关的用户不再支持共享版服务，老用户仍可继续使用共享版服务。即 API 网关当前已不提供共享版，目前只有存量用户可以使用共享版。

前提条件

已经创建 API 分组，此处以 APIGroup_test 分组为例，创建过程请参见[创建 API 分组](#)。

创建 APIG 触发器

步骤1 登录 FunctionGraph 控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数列表页面。

步骤2 单击“创建函数”，进入“创建函数”页面。

步骤3 设置以下函数信息。

- 函数名称：输入您自定义的函数名称，例如：apig。
- 委托名称：选择“不使用任何委托”。
- 企业项目：选择“default”。
- 运行时语言：选择“Python 2.7”。

步骤4 单击“创建函数”，完成函数的创建。

步骤5 在“代码”页签下，复制如下代码至代码窗并单击“部署”。

```
# -*- coding:utf-8 -*-
import json
def handler(event, context):
    body = "<html><title>FunctionGraph Demo</title><body><p>Hello, FunctionGraph!</p></body></html>"
    print(body)
    return {
        "statusCode":200,
        "body":body,
        "headers": {
            "Content-Type": "text/html",
        },
        "isBase64Encoded": False
    }
```

步骤6 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-14 创建触发器



步骤7 设置以下触发器信息。

表 6-11 触发器信息

字段	填写说明
触发器类型	选择“API Gateway服务 (APIG)”。
API名称	您自定义的API名称，例如：API_apig。
分组	API分组相当于一个API集合，API提供方以API分组为单位，管理分组内的所有API。 选择“APIGroup_test”。
发布环境	API可以同时提供给不同的场景调用，如生产、测试或开发。API网关服务提供环境管理，在不同的环境定义不同的API调用路径。 选择“RELEASE”，才能调用。
安全认证	API认证方式： <ul style="list-style-type: none"> App：采用Appkey&Appsecret认证，安全级别高，推荐使用，详情请参见APP认证。 IAM：IAM认证，只允许IAM用户能访问，安全级别中等，详情请参见IAM认证。 None：无认证模式，所有用户均可访问。 选择“None”。
请求协议	分为两种类型： <ul style="list-style-type: none"> HTTP HTTPS 选择“HTTPS”。
后端超时(毫秒)	输入“5000”。

步骤8 单击“确定”，完成触发器的创建。

📖 说明

1. APIG触发器调用地址：`https://0ed9f61512d34982917a4f3cfe8ddd5d.apig.example.example.com/apig`。
2. API触发器创建完成后，会在API网关生成名为API_apig的API，单击API名称，跳转至API网关服务。

----结束

调用函数

步骤1 在浏览器地址栏输入APIG触发器的调用URL，按“Enter”。

步骤2 函数执行完毕，得到返回结果。

----结束

6.12 使用 APIC 触发器

本节介绍创建APIC触发器，使用API调用函数运行。供用户了解APIC触发器的使用方法。（当前特性仅亚太-新加坡区域支持）

关于APIC触发器事件源具体介绍请参见[支持的事件源](#)。

前提条件

已经创建API分组，此处以APIConnect_test分组为例，创建过程请参见[创建API分组](#)。

创建 APIC 触发器

步骤1 登录FunctionGraph控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数列表页面。

步骤2 单击“创建函数”，进入“创建函数”页面。

步骤3 设置以下函数信息。

- 函数名称：输入您自定义的函数名称，例如：`apic`。
- 委托名称：选择“不使用任何委托”。
- 企业项目：选择“default”。
- 运行时语言：选择“Node.js 10.16”。

步骤4 单击“创建函数”，完成函数的创建。

步骤5 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-15 创建触发器



步骤6 设置以下触发器信息。

表 6-12 触发器信息

字段	填写说明
触发器类型	选择“API Connect服务 (APIC)”。
实例	选择所属实例，若无实例，可单击“创建实例”完成创建。
API名称	您自定义的API名称，例如：API_apic。
分组	API分组相当于一个API集合，API提供方以API分组为单位，管理分组内的所有API。 例如：“DEFAULT”。
发布环境	API可以同时提供给不同的场景调用，如生产、测试或开发。API网关服务提供环境管理，在不同的环境定义不同的API调用路径。 选择“RELEASE”，才能调用。
安全认证	API认证方式： <ul style="list-style-type: none"> • App：采用Appkey&Appsecret认证，安全级别高，推荐使用，详情请参见APP认证。 • IAM：IAM认证，只允许IAM用户能访问，安全级别中等，详情请参见IAM认证。 • None：无认证模式，所有用户均可访问。 选择“None”。
请求协议	分为两种类型： <ul style="list-style-type: none"> • HTTP • HTTPS 选择“HTTPS”。
后端超时(毫秒)	输入“5000”。

步骤7 单击“确定”，完成触发器的创建。

说明

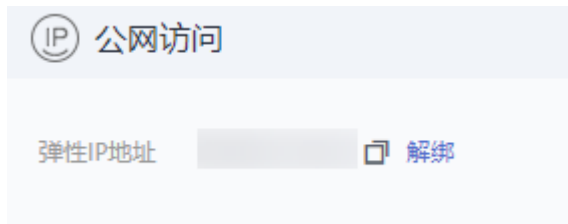
API触发器创建完成后，会在API网关生成名为API_apic的API，单击API名称，跳转至API网关服务。

----结束

调用函数

步骤1 进入应用与数据集成平台 ROMA Connect，找到所选实例（例如：Ac6-instance-NoDelete），查看公网IP。

图 6-16 公网 IP 地址



步骤2 在浏览器地址栏输入公网IP地址调用。

图 6-17 APIC 触发器调用地址



步骤3 函数执行完毕，得到返回结果。

图 6-18 返回结果



----结束

查看函数运行结果

步骤1 返回函数工作流控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 单击“nodejs-test”函数名称，进入函数详情页面。

步骤3 在函数详情页面，单击“日志”页签，查询函数运行日志。

步骤4 单击操作栏的“查看上下文”，查看日志详细信息。

----结束

6.13 使用 RabbitMQ 触发器

本节介绍创建RabbitMQ触发器，供用户了解RabbitMQ触发器的使用方法（当前只支持fanout路由模式）。使用RabbitMQ触发器后，FunctionGraph会定期轮询RabbitMQ实例指定交换机绑定的队列下的新消息，FunctionGraph将轮询得到的消息作为参数传递来调用函数，关于RabbitMQ触发器的事件源介绍请参见[支持的事件源](#)。

前提条件

- 已经创建函数，创建过程请参见[创建函数](#)。
- 创建RabbitMQ触发器，必须开启函数工作流VPC访问，请参见[配置网络](#)。
- 已经创建RabbitMQ实例，创建操作请参见[购买RabbitMQ实例](#)。

- 创建Vhost、Exchange和Queue。
 - a. 创建RabbitMQ Vhost，创建操作请参见[创建RabbitMQ Vhost](#)。
 - b. 创建RabbitMQ Exchange，创建操作请参见[创建RabbitMQ Exchange](#)。
 - c. 创建RabbitMQ Queue，创建操作请参见[创建RabbitMQ Queue](#)。
 - d. 绑定RabbitMQ Exchange和RabbitMQ Queue，绑定操作请参见[绑定RabbitMQ Exchange](#)、[绑定RabbitMQ Queue](#)。

📖 说明

Vhost是一个相对独立的RabbitMQ服务，用于管理Exchange、Queue。一个RabbitMQ实例下可以有多个Vhost，一个Vhost里可以有若干个Exchange和Queue，具体详情请参见[RabbitMQ业务使用流程](#)。

- 确认实例安全组规则是否配置正确。
 - a. 在RabbitMQ实例详情页面的“基本信息 > 网络”，单击安全组名称，跳转到安全组页面。
 - b. 选择“入方向规则”，查看安全组入方向规则。
 - i. 实例未开启SSL开关
 - 如果是VPC内访问，实例安全组入方向规则，需要允许端口5672的访问。
 - 如果是公网访问，需要允许端口15672的访问。
 - ii. 实例已开启SSL开关
 - 如果是VPC内访问，实例安全组入方向规则，需要允许端口5671的访问。
 - 如果是公网访问，需要运行端口15671的访问。

创建 RabbitMQ 触发器

步骤1 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-19 创建触发器



步骤4 设置以下信息。

- 触发器类型：选择“分布式消息服务RabbitMQ版 (RABBITMQ)”。
- *实例：选择已创建RabbitMQ实例。
- *密码：填写创建RabbitMQ实例的密码。
- *交换机名称：填写用户已创建的交换机名称，详情请参见[创建RabbitMQ Exchange](#)。
- 虚拟机名称：填写用户已创建的vhost，详情请参见[创建RabbitMQ Vhost](#)。
- *批处理大小：每次从Topic消费的消息数量。

步骤5 单击“确定”，完成RabbitMQ触发器的创建。

----结束

说明

开启函数流VPC访问后，需要在RabbitMQ服务安全组配置对应子网的权限。如何开启VPC访问请参见[配置网络](#)。

配置 RabbitMQ 事件触发函数

步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 在函数详情页，选择函数版本。

步骤4 在“代码”页签下，单击“测试”，弹出“配置测试事件”对话框。

步骤5 填写如[表6-13](#)所示测试信息后，单击“保存”。

表 6-13 测试信息

参数	说明
配置测试事件	可创建新的测试事件也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择“分布式消息服务 RabbitMQ版 (RABBITMQ)”模板，使用系统内置RabbitMQ事件模板。
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如kafka-123test。
测试事件	自动加载系统内置RabbitMQ事件模板，本例不做修改。

步骤6 单击“测试”，可以得到函数运行结果，函数会返回输入RabbitMQ消息数据。

----结束

6.14 使用开源 Kafka 触发器

本节介绍创建开源Kafka（OPENSOURCEKAFKA）触发器，供用户了解开源Kafka触发器的使用方法。

使用开源Kafka触发器后，FunctionGraph会定期轮询开源Kafka指定Topic下的新消息，FunctionGraph将轮询得到的消息作为参数传递来调用函数。

说明

- 分布式消息服务Kafka版与开源Kafka的差异说明，请参见[Kafka与开源Kafka的差异](#)。
- 对于处理失败的Kafka数据，Kafka触发器会丢弃过大记录（>6MB）。

前提条件

进行操作之前，需要做好以下准备。

- 已经创建函数。
- 创建Kafka触发器，必须开启函数工作流VPC访问，请参见[配置网络](#)。

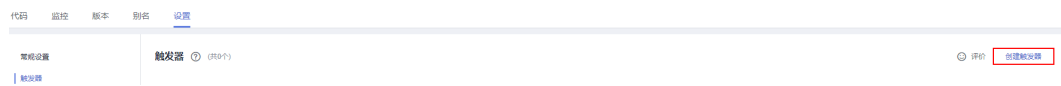
创建开源 Kafka 触发器

步骤1 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-20 创建触发器



步骤4 设置以下信息。

- 触发器类型：选择“开源Kafka（OPENSOURCEKAFKA）”。
- 连接地址：搭建kafka的broker地址列表，以逗号分隔。
- 主题：用户自建的topic。
- 批处理大小：单次函数拉取最大数据量。

步骤5 单击“确定”，完成开源kafka触发器的创建。

📖 说明

函数网络配置需要和创建kafka的ecs节点网络配置一样，包括vpc和子网。

---结束

激活 kafka 触发器

开源Kafka触发器创建完成后默认是停用状态，需要在触发器界面上单击“启动”。

📖 说明

如果启动失败可以联系技术支持工程师。

配置 Kafka 事件触发函数

步骤1 返回函数工作流控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 在函数详情页，选择函数版本。

步骤4 在“代码”页签下，单击“测试”，弹出“配置测试事件”对话框。

步骤5 填写如[表6-14](#)所示测试信息后，单击“保存”。

表 6-14 测试信息

参数	说明
配置测试事件	可创建新的测试事件也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择"开源Kafka (OPENSOURCEKAFKA)"模板，使用系统内置Kafka事件模板。
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如kafka-123test。
测试事件	自动加载系统内置kafka事件模板，本例不做修改。

步骤6 单击“测试”，可以得到函数运行结果，函数会返回输入kafka消息数据。

----结束

6.15 函数定时触发器 Cron 表达式规则

函数Cron表达式下支持如下几种配置方式。

- @every格式
@every *M*Unit，其中*N*表示一个正整数，Unit可以为ns, μs, ms, s, m, h，表示每隔*N*个Unit时间触发一次函数如表6-15所示。

表 6-15 表达式示例

表达式	含义
@every 30m	每隔30分钟触发一次函数
@every 1.5h	每隔1.5小时触发一次函数
@every 2h30m	每隔2小时30分钟触发一次函数

- 标准cron表达式
cron表达式格式要求“秒 分 时 日 月 周(可选)”，每个字段间以空格隔开，其中各字段说明如表6-16所示。

表 6-16 cron 表达式字段说明

字段	说明	取值范围	允许的特殊字符
CRON_TZ	可选。不设置则默认使用region所在时区。	-	-
秒	必选	0-59	, - * /
分钟	必选	0-59	, - * /

字段	说明	取值范围	允许的特殊字符
时	必选	0-23	, - * /
日(Day of month)	必选	1-31	, - * ? /
月	必选	1-12或者Jan-Dec (英文不区分大小写) 如表6-17所示。	, - * /
星期几(Day of week)	可选	0-6或者Sun-Sat (0表示星期天, 英文不区分大小写), 如表6-18所示。	, - * ? /

表 6-17 月份字段取值说明

月份	数字	英文简写
1月	1	Jan
2月	2	Feb
3月	3	Mar
4月	4	Apr
5月	5	May
6月	6	Jun
7月	7	Jul
8月	8	Aug
9月	9	Sep
10月	10	Oct
11月	11	Nov
12月	12	Dec

表 6-18 星期字段取值说明

星期	数字	英文简写
星期一	1	Mon
星期二	2	Tue
星期三	3	Wed

星期	数字	英文简写
星期四	4	Thu
星期五	5	Fri
星期六	6	Sat
星期日	0	Sun

cron表达式字段特殊字符说明如表6-19所示。

表 6-19 特殊字符说明

特殊字符	含义	说明
*	表示该字段中的所有值	在“分钟”字段中表示每一分钟都执行。
,	指定多个值（可以不连续）	在“月”字段中指定“Jan, Apr, Jul, Oct”或者“1,4,7,10”，表示1月，4月，7月和10月，在“星期几”字段中指定“Sat, Sun”或者“6,0”表示周六，周日。
-	指定一个范围	在“分钟”字段中使用0-3，表示从0分到3分
?	指定一个或另一个	仅“日”和“星期几”字段可以指定。例如，如果指定了一个特定的日期，但你不在乎该日期对应星期几，那么“星期几”字段就可以使用该特殊字符。
/	表示起步和步幅，n/m表示从n开始，每次增加m	在“分钟”字段1/3表示在满足其它字段情况下，从时间1分（例如00:01:00）开始，每隔3分钟触发一次。

cron表达式配置示例如表6-20所示。

表 6-20 cron 表达式配置示例

配置实例	Cron 表达式（以北京时区为例）
每天12点调度函数	CRON_TZ=Asia/Shanghai 0 0 12 * * *
每天12:30调度函数	CRON_TZ=Asia/Shanghai 0 30 12 * * *
每小时的26分，29分，33分调度函数	CRON_TZ=Asia/Shanghai 0 26,29,33 * * *
周一到周五的每天12:30调度函数	CRON_TZ=Asia/Shanghai 0 30 12 ? * MON-FRI

配置实例	Cron 表达式（以北京时区为例）
周一到周五的每天12:00 ~ 14:59每5分钟调度函数	CRON_TZ=Asia/Shanghai 0 0/5 12-14 ? * MON-FRI
一月到四月每天12:00调度函数	CRON_TZ=Asia/Shanghai 0 0 12 ? JAN,FEB,MAR,APR *

📖 说明

Cron表达式未设置时，默认以region所在时区运行。如果您的任务需要按照特定时区运行，可以通过CRON_TZ指定，例如在北京时间每个月一号的04:00触发函数执行，则可以使用CRON_TZ=Asia/Shanghai 0 0 4 1 * *。不同地域的时区表达式存在差异，请以实际情况为准。

6.16 使用 EG 触发器

6.16.1 创建 EG 触发器（RocketMQ 自定义事件源）

前提条件

进行操作之前，需要做好以下准备：

- 已经创建函数并开启VPC，创建过程请参见[创建函数](#)和[配置VPC网络](#)。
- 已经创建EG事件通道，创建过程请参见[创建eg事件通道](#)。
- 已经创建RocketMQ实例，创建过程请参见[购买RocketMQ实例](#)。
- 已经创建RocketMQ topic，创建过程请参见[创建topic](#)。
- 已经创建RocketMQ 消费组，创建过程请参见[创建消费组](#)。

📖 说明

- EG触发器当前仅支持亚太-曼谷、亚太-新加坡、非洲-约翰内斯堡、拉美-墨西哥城二、拉美-圣地亚哥。
- 函数绑定的VPC和RocketMQ实例的VPC需保持一致。开启函数流VPC访问后，需要在RocketMQ服务安全组配置对应子网的权限。

创建 EventGrid 触发器

步骤1 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-21 创建触发器



步骤4 设置以下信息。

- 触发器类型：选择“分布式消息服务RocketMQ版 (HC.ROCKETMQ)”。
- 触发器名称：填写自定义的名称。
- 事件通道：选择已创建的eg事件通道。
- 实例：选择已创建的RocketMQ实例。
- Topic：选择已创建的RocketMQ topic。
- 消费组：选择已经创建的RocketMQ 消费组。
- 用户名：RocketMQ实例开启acl时需要填写。连接RocketMQ实例的用户名。
- 密码：RocketMQ实例开启acl时需要填写。连接RocketMQ实例的密码。

步骤5 单击“确定”，完成EG触发器的创建。

----结束

配置 EventGrid 事件触发函数

步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 在函数详情页，选择函数版本。

步骤4 在“代码”页签下，单击“测试”，弹出“配置测试事件”对话框。

步骤5 填写如下表6-21所示测试信息后，单击“保存”。

表 6-21 测试参数

参数	说明
配置测试事件	可创建新的测试事件，也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择“分布式消息服务 RocketMQ版 (HC.ROCKETMQ)”模板，使用系统内置 HC.ROCKETMQ事件模板。
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如 rocketmq-123test。
测试事件	自动加载系统内置eg-RocketMQ事件模板，本例不做修改。

步骤6 单击“测试”，可以得到函数运行结果，函数会返回输入eg-rocketmq消息数据。

----结束

6.16.2 创建 EG 触发器（OBS 应用事件源）

前提条件

已创建OBS存储桶，此处以eventbucket桶为例。创建过程请参见[创建存储桶](#)。

说明

EG触发器当前仅支持亚太-曼谷、亚太-新加坡、非洲-约翰内斯堡、拉美-墨西哥城二、拉美-圣地亚哥。

创建 EventGrid 触发器

- 步骤1 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2 选择待配置的函数，单击进入函数详情页。
- 步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框。

图 6-22 创建触发器



- 步骤4 设置以下信息。
 - 触发器类型：选择“OBS应用事件源”。
 - 触发器名称：填写自定义的名称。
 - 桶：选择已创建的obs桶。
 - 事件类型：选择需要的事件类型。
 - 对象名前缀：用来限制以此关键字开头的对象的事件通知，该限制可以实现对OBS对象名的过滤。
 - 对象名后缀：用来限制以此关键字结尾的对象的事件通知，该限制可以实现对OBS对象名的过滤。
 - 对象名编码：是否对对象进行编码。

说明

当事件类型选择“指定对象版本号删除对象”时，需要开启多版本控制功能，具体详情请参见[多版本控制功能启用时的对象删除机制](#)。

- 步骤5 单击“确定”，完成EG触发器的创建。

----结束

配置 EventGrid 事件触发函数

- 步骤1 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2 选择待配置的函数，单击进入函数详情页。
- 步骤3 在函数详情页，选择函数版本。
- 步骤4 在“代码”页签下，单击“测试”，弹出“配置测试事件”对话框。

步骤5 填写如下表6-22所示测试信息后，单击“保存”。

表 6-22 测试参数

参数	说明
配置测试事件	可创建新的测试事件，也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择“空白”模板，代码请参见 EG示例事件 中“OBS应用事件源”。
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如myobs-123test。
测试事件	使用新创建的测试事件。

----结束

6.16.3 创建 EG 触发器（云服务事件源）

📖 说明

EG触发器当前仅支持亚太-曼谷、亚太-新加坡、非洲-约翰内斯堡、拉美-墨西哥城二、拉美-圣地亚哥。

创建 EventGrid 触发器

步骤1 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 选择待配置的函数，单击进入函数详情页。

步骤3 选择“设置 > 触发器”，单击“创建触发器”，弹出“创建触发器”对话框，本章节以创建对象存储OBS为例。

图 6-23 创建触发器



步骤4 设置以下信息：

- 触发器类型：选择“对象存储服务 OBS”。
- 触发器名称：填写自定义的名称。
- 事件类型：选择需要的事件类型。

步骤5 单击“确定”，完成EG触发器的创建。

----结束

配置 EventGrid 事件触发函数

- 步骤1** 返回函数 workflow 控制台，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2** 选择待配置的函数，单击进入函数详情页。
- 步骤3** 在函数详情页，选择函数版本。
- 步骤4** 在“代码”页签下，单击“测试”，弹出“配置测试事件”对话框。
- 步骤5** 填写如下表6-23所示测试信息后，单击“保存”。

表 6-23 测试参数

参数	说明
配置测试事件	可创建新的测试事件，也可编辑已有的测试事件。 选择默认值：“创建新的测试事件”。
事件模板	选择“空白”模板，代码请参见 EG示例事件 中“云服务事件源”。
事件名称	事件名称必须以大写或小写字母开头，支持字母（大写或小写），数字和下划线“_”（或中划线“-”），并以字母或数字结尾，长度为1-25个字符，例如 myobs-123test。
测试事件	使用新创建的测试事件。

---结束

7 调用函数

7.1 同步调用

同步调用指的是客户端触发函数后，需阻塞等待函数调用结果返回的场景。

当前的API网关APIG（共享版）、API网关APIG（专享版）、服务集成APIC是默认同步触发。另外，您也可以使用[同步执行函数](#)接口同步触发函数。

同步调用场景下，函数最大运行时长限制为15分钟。

7.2 异步调用

异步调用指的是客户端触发函数后，FunctionGraph持久化请求并立即返回，客户端不等待请求最终处理完成，用户无法实时感知请求处理结果。FunctionGraph最终将异步请求排队，在服务端空闲的情况下逐个处理。如果您希望获取异步请求结果通知或者设置异步请求失败重试，请参见[配置函数异步](#)。

- 以下触发器：默认异步调用，用户不可修改。

表 7-1 调用方式

事件源	调用方式
消息通知服务SMN	异步调用
对象存储服务OBS	异步调用
数据接入服务DIS	异步调用
定时器TIMER	异步调用
云日志服务LTS	异步调用
文档数据库服务DDS	异步调用
分布式消息服务Kafka	异步调用
分布式消息服务RabbitMQ版	异步调用

事件源	调用方式
云数据库GeminiDB Mongo	异步调用

- 以下触发器：API网关APIG、API网关APIG（专享版）、服务集成APIC可以在触发器对应服务页面配置成异步触发方式。您也可以使用[异步执行函数](#)API接口异步触发函数。异步调用场景下，函数最大运行时长限制为12小时（通过白名单配置）。

📖 说明

如果函数执行端到端时延超过90s，建议使用异步不使用同步，否则会因为网关限制，超过90s后无法收到同步响应。

示例

在已创建函数并配置APIG触发器的前提下，以APIG触发器为例，配置异步触发。

- 在函数列表中打开函数，单击“设置 > 触发器”。
- 单击已配置的APIG触发器名称，跳转到APIG服务页面。

图 7-1 单击触发器名称



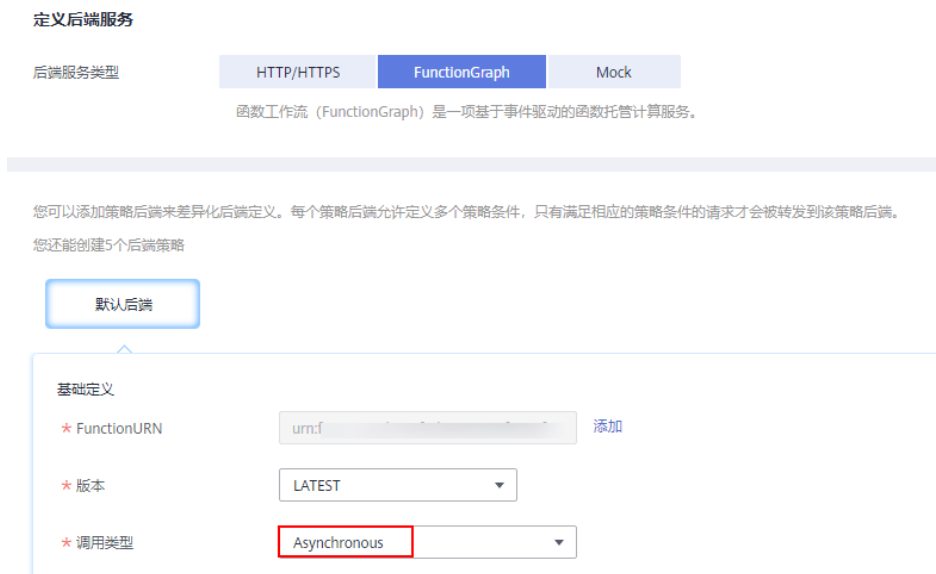
- 单击右上角的“编辑”。

图 7-2 单击“编辑”



- 单击“下一步”到“定义后端服务”页面，修改调用类型为“Asynchronous”。

图 7-3 修改调用类型



5. 单击“立即完成”，进行保存。

7.3 重试机制

函数在同步调用或异步调用执行失败时，您可以参见以下重试机制进行操作。

- 同步调用
同步调用执行失败，建议您自行尝试重试。
- 异步调用
异步调用可在界面配置最大重试次数和消息最大有效期，具体配置方法请参见[配置函数异步](#)。函数平台会根据您配置的最大重试次数和消息最大有效期（最大有效期为24小时），进行重试。重试次数和配置的最大重试次数一致，重试有效期和配置的消息最大有效期一致。

幂等性

在编程中，幂等性指应用程序或组件具备识别重复事件和防止重复、不一致或数据丢失的能力。若您想使函数保持幂等性，则需要通过函数逻辑设计来正确处理重复的事件。

幂等函数逻辑有助于减少以下问题：

- 不必要的 API 调用
- 代码处理时间
- 数据不一致
- 限制
- 延迟

请确保您的函数代码可以多次处理相同的事件，而不会导致重复的事务或其他不必要的副作用。如果函数不满足幂等性要求，则当函数调用异常，客户端重试或依赖函数内部重试时，可能会导致重复的事务或其他不必要的副作用。

8 监控

8.1 指标

8.1.1 监控信息说明

FunctionGraph函数实现了与云监控服务的对接，用户无需任何配置，即可查询函数监控信息。

查看函数监控信息

FunctionGraph会统计函数的运行时指标，显示的指标是函数运行时活动的聚合视图。要查看不同函数版本的指标，可在查看指标前切换函数版本，查询不同版本对应的监控信息。

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入函数详情页。
3. 选择“监控 > 指标”，选择时间粒度（最近1天、最近3天、自定义），查看函数运行状态。

📖 说明

可以查看的指标有：调用次数、错误次数、运行时间（包括最大运行时间、最小运行时间、平均运行时间）、被拒绝次数、资源统计。

指标说明

运行监控指标说明如[表8-1](#)所示。

表 8-1 监控指标说明表

指标	单位	说明
调用次数	次	函数总的调用请求数，包含了错误和被拒绝的调用。异步调用在该请求实际被系统执行时才开始计数。

指标	单位	说明
运行时间	毫秒	最大运行时间为某统计粒度（周期）下，即某一时间段内单次函数执行最大的运行时间。 最小运行时间为某统计粒度（周期）下，即某一时间段内单次函数执行最小的运行时间。 平均运行时间为某统计粒度（周期）下，即某一时间段内单次函数执行平均的运行时间。
错误次数	次	指发生异常请求的函数不能正确执行完并且返回 200，都计入错误次数。函数自身的语法错误或自身执行错误也会计入该指标。
被拒绝次数	次	由于并发请求太多，系统流控而被拒绝的请求次数。
资源统计	个	该函数的请求并发数和预留实例数。
内存使用量	MB	最大内存使用量为某统计粒度（周期）下，即某一时间段内单次函数执行最大的内存使用量。 最小内存使用量为某统计粒度（周期）下，即某一时间段内单次函数执行最小的内存使用量。 平均内存使用量为某统计粒度（周期）下，即某一时间段内单次函数执行平均的内存使用量。

8.1.2 FunctionGraph 服务的监控指标参考

功能说明

本节定义了FunctionGraph上报云监控服务的监控指标的命名空间，监控指标列表和维度定义，用户可以通过云监控服务提供管理控制台或API接口来检索FunctionGraph产生的监控指标和告警信息。

命名空间

SYS.FunctionGraph

函数监控指标

表 8-2 FunctionGraph 支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
count	调用次数	该指标用于统计函数调用次数。 单位：次	≥ 0 counts	函数	5分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期 (原始指标)
failcount	错误次数	该指标用于统计函数调用错误次数。 以下两种情况都会计入错误次数： <ul style="list-style-type: none"> 函数请求异常，导致无法执行完成且返回200。 函数自身语法错误或者自身执行错误。 单位：次	≥ 0 counts	函数	5分钟
failRate	错误率	该指标用于统计函数调用错误次数在总调用次数中的占比。 单位：%	$0\% \leq X \leq 100\%$	函数	5分钟
rejectcount	被拒绝次数	该指标用于统计函数调用被拒绝次数。 被拒绝次数是指并发请求太多，系统流控而被拒绝的请求次数。 单位：次	≥ 0 counts	函数	5分钟
concurrency	并发数	该指标用于统计函数同时调用处理的最大并发请求个数。 单位：个	≥ 0 counts	函数	5分钟
reservedinstancenum	预留实例个数	该指标用于统计函数配置的预留实例个数。 单位：个	≥ 0 counts	函数	5分钟
duration	平均运行时间	该指标用于统计函数调用平均运行时间。 单位：毫秒	≥ 0 ms	函数	5分钟
maxDuration	最大运行时间	该指标用于统计函数调用最大运行时间。 单位：毫秒	≥ 0 ms	函数	5分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期 (原始指标)
minDuration	最小运行时间	该指标用于统计函数最小运行时间。 单位：毫秒	≥ 0 ms	函数	5分钟
systemErrorCount	系统错误次数	该指标用于统计函数请求异常，导致无法执行完成的错误次数。单位：次	≥ 0 counts	函数	5分钟
functionErrorCount	函数错误次数	该指标用于统计函数自身语法错误或者自身执行错误次数。单位：次	≥ 0 counts	函数	5分钟
payPerUseInstance	弹性实例个数	该指标用于统计函数使用的弹性实例个数。单位：个	≥ 0 counts	函数	5分钟

表 8-3 函数流支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期 (原始指标)
totalCount	调用次数	用于统计函数流调用次数。 单位：次	≥ 0 counts	函数流	1分钟
errorCount	错误次数	该指标用于统计函数调用错误次数。 单位：次	≥ 0 counts	函数流	1分钟
running	正在运行数量	该指标用于统计正在运行状态的函数流。 单位：个	≥ 0 counts	函数流	1分钟
rejectCount	被拒绝次数	该指标用于统计函数流调用被拒绝次数。 单位：个	≥ 0 counts	函数流	1分钟
averageDuration	平均运行时间	该指标用于统计函数流调用平均耗时。 单位：毫秒	≥ 0 ms	函数流	1分钟

维度

key	value
package-functionname	应用名-函数名。 示例：default-myfunction_Python。
graph_name	函数流。
projectId	租户

8.1.3 创建告警规则

函数及触发器创建以后，可以实时监控函数被调用及运行情况。

监控函数

不同版本函数的监控信息做了区分，查询函数指标之前设置函数版本，可以查询不同版本对应的监控信息。

操作步骤

函数实现与云监控服务的对接，函数上报云监控服务的监控指标，用户可以通过云监控服务来查看函数产生的监控指标和告警信息。

- 步骤1** 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2** 单击函数名称，进入函数详情界面。
- 步骤3** 选择函数对应的版本或者别名，选择“监控 > 指标”。
- 步骤4** 单击“创建告警规则”，弹出“创建告警规则”对话框。
- 步骤5** 输入告警参数，单击“下一步”。如[图8-1](#)所示。

图 8-1 创建告警规则

创建告警规则

i 告警基础功能免费，触发产生的告警消息由SMN发送，可能产生少量费用，具体费用由SMN结算。[了解详情](#)

* 名称 ×

* 监控指标

* 告警策略

当聚合方式为原始值时，无需选择监控周期

* 告警级别 紧急 重要 次要 提示

发送通知

描述 0/255

步骤6 输入告警规则名称，单击“确定”。

----结束

⚠ 注意

删除函数后，已创建的告警规则在CES服务控制台里不会实时更新，可能会继续在CES服务控制台里显示最多 7 天。

监控指标说明

告警监控指标如表8-4所示。

表 8-4 函数告警监控指标说明表

指标名称	显示名	描述	单位	上限值	下限值	建议阈值	值类型	所属维度
count	调用次数	该指标用于统计函数调用次数	次	-	0	-	int	package-functionname
failcount	错误次数	该指标用于统计函数调用错误次数	次	-	0	-	int	package-functionname
rejectcount	被拒绝次数	该指标用于统计函数调用被拒绝次数	次	-	0	-	int	package-functionname
duration	平均运行时间	该指标用于统计函数调用平均运行时间	毫秒	-	0	-	float	package-functionname
maxDuration	最大运行时间	该指标用于统计函数调用最大运行时间	毫秒	-	0	-	float	package-functionname
minDuration	最小运行时间	该指标用于统计函数调用最小运行时间	毫秒	-	0	-	float	package-functionname

指标名称	显示名	描述	单位	上限值	下限值	建议阈值	值类型	所属维度
concurrency	并发数	该指标用于统计函数同时调用处理的最大并发请求个数	个	-	0	-	int	package-functionname
payPerUseInstance	弹性实例数	该指标用于统计函数排除预留实例后实际使用的实例数	个	-	0	-	int	package-functionname
failRate	调用错误率	该指标用于统计函数调用错误率	%	-	0	-	float	package-functionname
functionErrorCount	函数错误次数	该指标用于统计函数调用函数错误次数	次	-	0	-	float	package-functionname
memoryUsed	函数执行内存使用	该指标用于统计函数执行内存使用量	MB	-	0	-	float	package-functionname
duration_p50	函数执行时间p50	该指标用于统计函数执行时间的p50	毫秒	-	0	-	float	package-functionname
duration_p80	函数执行时间p80	该指标用于统计函数执行时间的p80	毫秒	-	0	-	float	package-functionname
duration_p95	函数执行时间p95	该指标用于统计函数执行时间的p95	毫秒	-	0	-	float	package-functionname
duration_p990	函数执行时间p990	该指标用于统计函数执行时间的p990	毫秒	-	0	-	float	package-functionname

指标名称	显示名	描述	单位	上限值	下限值	建议阈值	值类型	所属维度
duration_p999	函数执行时间p999	该指标用于统计函数执行时间的p999	毫秒	-	0	-	float	package-functionname
instances	实例数	该指标用于统计函数调用实例的个数	个	-	0	-	int	package-functionname
systemErrorCount	系统错误次数	该指标用于统计函数调用系统错误次数	次	-	0	-	int	package-functionname
reservedinstances	预留实例数	该指标用于统计函数运行的预留实例数	个	-	0	-	int	package-functionname
functionCost	资源使用量	该指标用于统计函数执行资源使用量，即函数执行内存使用 * 函数执行时间	MB	-	0	-	float	package-functionname

表 8-5 函数流告警监控指标说明表

指标名称	显示名	描述	单位	上限值	下限值	建议阈值	值类型	所属维度
ExecutionsStarted	工作流实例启动个数	该指标用于统计用户工作流实例开始运行的实例个数	个	-	0	-	int	ProjectId

指标名称	显示名	描述	单位	上限值	下限值	建议阈值	值类型	所属维度
ExecutionsAborted	工作流实例执行被中断个数	该指标用于统计用户工作流实例被终止运行的个数	个	-	0	-	int	ProjectId
ExecutionsTimedOut	工作流实例执行超时个数	该指标用于统计用户工作流实例超时运行的个数	个	-	0	-	int	ProjectId
ExecutionsSucceeded	工作流实例执行成功个数	该指标用于统计用户工作流实例成功运行完成的实例个数	个	-	0	-	int	ProjectId
ExecutionsFailed	工作流实例执行失败个数	该指标用于统计用户工作流实例运行失败的实例个数	个	-	0	-	int	ProjectId

8.2 日志

8.2.1 查看函数日志

FunctionGraph函数实现了与云日志服务的对接，用户无需任何配置，即可查询函数日志信息。

函数日志信息

在FunctionGraph函数控制台，可以通过以下两种方式查看函数日志。

- 在测试页签查看日志
函数创建完成后，可以测试函数，在执行结果页，可以查看函数测试日志。操作步骤请参见[在线调试](#)。
此处最多显示2KB字节日志，如果日志太多，可以去函数详情页日志页签查询日志。
- 在日志页签查看日志
在函数详情页“监控 > 日志”页签，查询日志信息，操作步骤请参见[管理函数日志](#)。

日志下载

应用运维服务（AOM）管理函数日志支持下载，选择版本和时间范围，单击“下载”即可下载该时间范围内的所有日志。

说明

- 一次最多只能下载5000条日志，请合理选择时间范围，避免下载的日志缺失。
- 当前日志时间戳打印的时间为 UTC 时间。

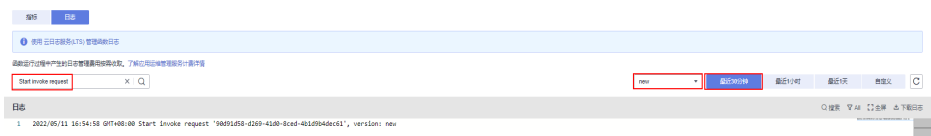
8.2.2 管理函数日志

须知

FunctionGraph V1版本的函数支持应用运维管理服务（AOM）管理函数日志及云日志服务（LTS）管理函数日志。

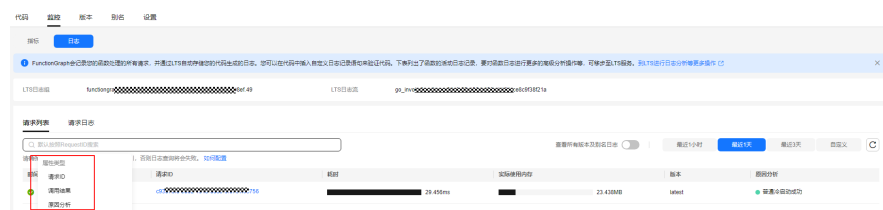
FunctionGraph V2版本的函数支持云日志服务（LTS）管理函数日志。

- 应用运维管理服务（AOM）管理函数日志页面



- 云日志服务（LTS）管理函数日志页面

图 8-2 日志页面



应用运维服务（AOM）管理函数日志

- 步骤1** 登录[函数工作流控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
- 步骤2** 单击函数名称，进入函数详情界面。

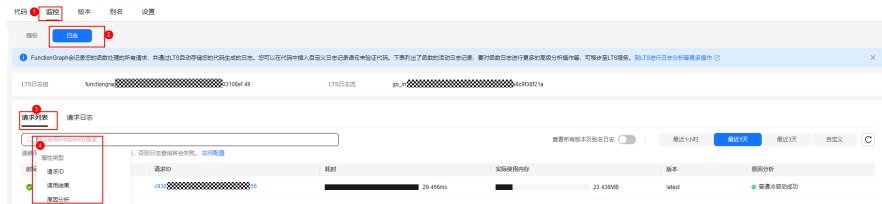
步骤3 选择“监控 > 日志”，在“日志”页签，输入查询条件。

说明

- 支持的日志查询条件：
 - 1、关键词精确搜索。关键词指相邻两个分词符之间的单词。
 - 2、关键词模糊匹配搜索，例如输入“*ROR*”或“ERR*”或“ER*OR”。
 - 3、短语精确搜索，例如输入“Start to refresh alm Statistic”。
 - 4、关键词的“与”、“或”组合搜索。格式为“query&&logs”或“query||logs”。
- 支持的时间条件：最近30分钟、最近1小时、最近1天及自定义（时间范围为一个月，比如2022/04/01 16:34:48~2022/05/01 16:34:48）。
- 支持选择版本查询日志。

步骤4 单击  ，查询日志。

图 8-3 查询日志



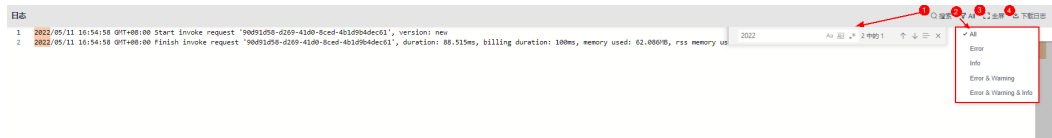
说明

日志查询结果包含的字段有：时间、请求ID、调用结果、耗时、内存、版本。

步骤5 在已查询的日志中还支持如下操作：

- ①按关键字搜索过滤日志
- ②按照日志状态搜索过滤日志：Error、Info、Error & Warning、Error & Warning & Info
- ③支持全屏展示
- ④支持日志下载

图 8-4 其他操作



----结束

云日志服务（LTS）管理函数日志

FunctionGraph支持开通云日志服务(LTS)，使用更丰富的函数日志管理功能。开通云日志服务后，FunctionGraph会自动创建1个日志组（functiongraph开头），创建函数后，会默认生成一个日志流（函数名称开头）。

您也可以针对某个函数自行关联日志组和日志流，管理函数日志，即进行函数调用后，调用日志会保存到指定的日志组和日志流下。具体操作请参见[配置日志组及日志流](#)。

说明

- 默认创建的20个日志流，您无法自定义。您可以在函数的“日志”页签下，按“F12”，找到query接口里的日志流ID，再到lts里找到对应的日志流ID。



- 若在LTS控制台误删函数日志组，之前的日志数据不可找回，FunctionGraph服务不感知该操作。此时您可以通过修改函数常规设置中的描述信息，保存后触发重建函数日志组。

步骤1 开通云日志服务（LTS）管理函数日志。

FunctionGraph V1版本开通：在“日志”页签，单击“使用云日志服务(LTS)管理函数日志”，页面直接切换至LTS管理函数日志的页面。

说明

FunctionGraph V1版本支持“切换旧版”，将会停用云日志服务(LTS)，并切换为应用运维管理服务(AOM)管理函数日志。函数运行过程中产生的日志管理费用将按需收取。

FunctionGraph V2版本开通：在“日志”页签，直接单击“点击开通”。继续单击“确认”，右上角弹出“开通成功”。

说明

FunctionGraph V2版本当前只支持使用LTS管理函数日志。

步骤2 设置查询条件。

- 请求列表：支持设置请求ID、调用结果（执行成功、执行失败）、原因分析（初始化失败、加载失败、系统错误、调用超时、内存超限、磁盘超限、代码异常）。
- 请求日志：支持关键字、请求ID、实例ID。

表 8-6 调用结果

调用结果	说明
执行成功	函数执行成功打印的日志。
执行失败	函数执行失败打印的日志，包含调用超时、内存超限、磁盘超限、代码异常四种情况。 若想查看调用超时的日志信息，请将“日志类型”切换为调用超时，另外3种执行失败下的日志类型查看方法相同。

表 8-7 原因分析

原因分析	说明
初始化失败	函数初始化失败打印的日志。
加载失败	runtime加载用户函数文件失败打印的日志
系统错误	内部错误。
调用超时	函数调用时间超过配置的“执行超时时间”打印的日志。
内存超限	函数内存大小超过配置的“内存”大小打印的日志。
磁盘超限	磁盘超出限制大小打印的日志。
代码异常	代码出现异常情况打印的日志。

📖 说明

- 支持的时间条件：最近1小时、最近1天、最近3天及自定义。
- 您可以单击“到LTS进行日志分析等更多操作”，前往LTS控制台管理函数日志。
- 用户普通实例的初始化阶段的日志大小限制为（10MB），超过大小限制的日志进行滚动更新，为您保留最新的日志。

---结束

下载日志

须知

- 当前仅使用应用运维服务（AOM）管理函数日志时，支持下载日志。
- FunctionGraph V1版本的函数支持应用运维管理服务（AOM）管理函数日志。
- FunctionGraph V2版本的函数支持云日志服务（LTS）管理函数日志，不支持日志下载。

步骤1 返回函数 workflow 控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数列表界面。

步骤2 单击函数名称，进入函数详情界面。

步骤3 单击“监控”，进入“日志”页签。

步骤4 在“请求日志”下，选择版本和时间范围，单击“下载日志”。

📖 说明

一次最多只能下载5000条日志，请合理选择时间范围，避免下载的日志缺失。

---结束

8.3 调用链管理

说明

当前特性仅“华东-上海一、华北-北京四”区域支持。

概述

用户在函数的“监控”页签，开启调用链能力。开启后可以在函数“监控 > 调用链”“页面或跳转至APM服务”应用监控 > 调用链”页面，查看函数调用链信息。当前仅支持Java8和Java11函数。

前提条件

- 调用链只支持512MB内存以上的函数，若函数内存低于512MB，无法使用调用链功能，请在“设置 > 常规信息”中，扩大函数“内存”后再使用调用链功能。
- 已开通应用性能管理APM服务的使用权限，具体请参见[APM权限管理](#)。若未开通，则无法获取调用链相关数据。

开通调用链

步骤1 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。

步骤2 单击函数名称，进入函数详情界面。

步骤3 选择“监控 > 调用链”。

步骤4 单击“点击开通”。

图 8-5 开通调用链监控



步骤5 此时调用链会从应用性能管理APM服务自动获取一个访问密钥AK/SK，如[图8-6](#)。

图 8-6 获取访问密钥

- 已有访问密钥：在“访问密钥ID”的下拉列表中选择“访问密钥ID”，选择完成后单击“确定”，完成开通。
- 无访问密钥：单击“创建”进入APM控制台，参见[新增访问密钥](#)，新增后的AK/SK会同步至FunctionGraph控制台。

步骤6 管理调用链。

- 单击右上方“关闭调用链”，可以直接关闭调用链,关闭调用链后将不能查看函数调用链。
- 若在APM控制台对AK/SK进行了编辑修改，返回FunctionGraph控制台，单击右上方“更新AKSK”，在弹框中更新AK/SK。

----结束

查询调用链详情

- 步骤1** 返回函数 workflow 控制台，在左侧导航栏选择“函数 > 函数列表”，单击已开启调用链的函数，进入函数详情页面。
- 步骤2** 在“监控”页签下，选择“调用链”。
- 步骤3** 在左侧设置查询条件，完成后单击“查询Trace”。

图 8-7 设置查询条件



- 时间：设置查询时间，注意起止查询时间间隔不能超过24小时。
- 响应时间：设置响应时间。
- 执行结果：选择“全部/执行成功/执行失败”。
- Trace ID：调用链的TraceID，填写该搜索条件后，其他搜索条件全部失效，只根据该TraceID搜索。

步骤4 在右侧查看调用链详情。

查看单条调用链详情：在查询到的调用链结果中，单击调用链名称，进入APM控制台查看。

图 8-8 单击调用链名称

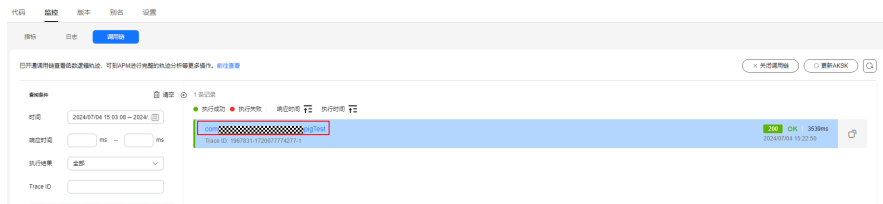
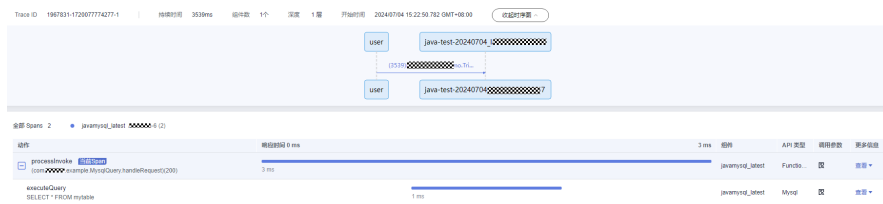
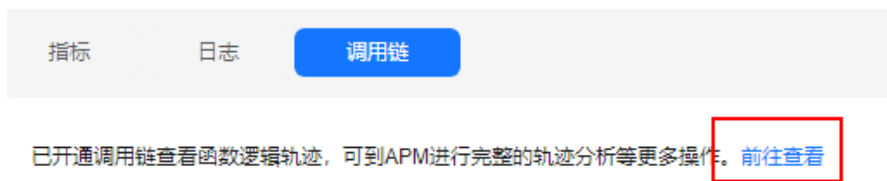


图 8-9 单条详情



查看所有调用链信息：单击“前往查看”，到APM进行完整的路径分析等更多操作。

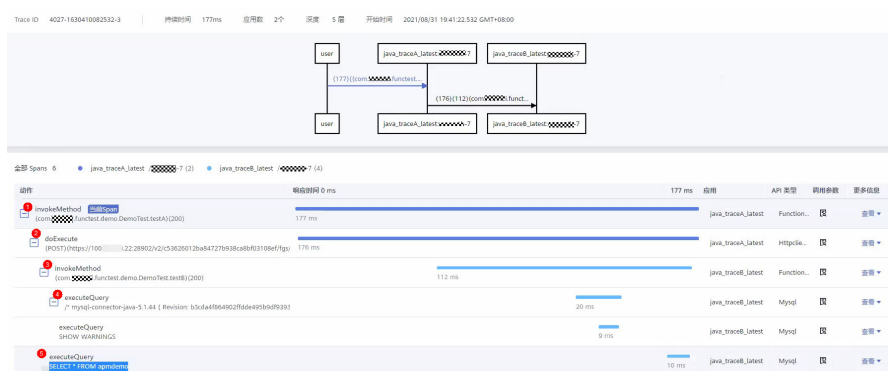
图 8-10 前往 APM



示例

1. 如下是在函数A (DemoTestA) 中通过HTTP请求方式调用函数B (DemoTestB) 的完整函数执行过程。

图 8-11 函数 A 调用函数 B 详情



- ①函数A方法的总耗时。
 - ②通过HTTP请求调用函数B。
 - ③进入函数B。
 - ④执行executeQuery。
 - ⑤执行Select查询语句。
2. 如下是在函数首次调用包含冷启动的完整函数执行过程。
Spans动作说明如下：
 - **load**: 是下载解压用户函数代码包和依赖包的时间。
 - **preload**: 运行时加载用户函数代码和初始化函数执行环境的时间。
 - **init**: 初始化函数的执行时间, 初始化函数只有在冷启动时才会被执行。
 - **processInvoke**: 函数的执行时间。

说明

您可以参见调用链, 进一步了解调用链详情。

----结束

9 函数管理

概述

函数是实现某一功能所需代码、运行时、资源、设置的组合，是可以独立运行的最小单元。函数通过Trigger触发，自行调度所需资源及环境，实现预期功能。

导出函数

FunctionGraph支持将已创建的函数导出。

- 步骤1** 登录FunctionGraph控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数页面。
- 步骤2** 在“函数”页面，单击函数名称，进入函数详情页面。
- 步骤3** 在函数详情页面，选择函数版本，单击操作列表中的“导出函数”，即可将该函数导出。

📖 说明

- 同一时段单个用户只能并发导出一个函数。
- 导出函数资源包大小50MB以内。
- 导出的函数资源名称为函数名+函数代码的MD5值.zip。
- 导出的函数资源中配置信息不包含别名信息。
- 当函数被禁用/启用后，该函数下所有版本的函数都会被禁用/启用。

----结束

禁用函数

用户可以根据实际情况将函数禁用，禁用期间函数不能执行。

- 步骤1** 返回函数 workflow 控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数页面。
- 步骤2** 在“函数列表”，单击要禁用的函数名称，进入“函数详情”页面。
- 步骤3** 单击“禁用函数”。
- 步骤4** 单击“确定”，函数被禁用。

📖 说明

- 只能禁用“latest”版本的函数，不能禁用已经发布的版本的函数。
- 基于已禁用的“latest”版本重新发布新版本，发布后的新版本也处于禁用状态且不能启用。
- 当函数处于禁用状态时可以修改代码，不能执行函数。

----结束

启用函数

用户可以根据实际情况将已禁用的函数重新启用。

步骤1 返回函数 workflow 控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数页面。

步骤2 在“函数列表”，单击要启用的函数名称，进入“函数详情”页面。

步骤3 单击“启用函数”，函数被启用。

----结束

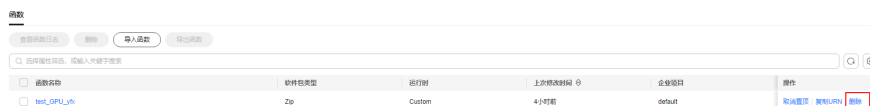
删除函数

对于已经不再使用的函数，可以进行删除操作，及时释放资源。

步骤1 返回函数 workflow 控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数列表界面。

步骤2 在“函数列表”中，单击待删除函数所在行右侧的“删除”，然后输入“DELETE”，单击“确定”完成函数删除。

图 9-1 删除函数



----结束

10 依赖包管理

10.1 配置依赖包

概述

函数代码一般包含公共库和业务逻辑两部分。对于公共库，您可以打包成依赖包单独管理，共享给多个函数使用，同时也减少了函数代码包部署、更新时的体积。

FunctionGraph也提供了一些[公共依赖包](#)，公共依赖包在平台内部缓存，消除了冷启动加载的影响，推荐您优先使用。

依赖包管理模块统一管理用户所有的依赖包，用户可以通过本地上传和obs地址的形式上传依赖包，并为依赖包命名。同时支持用户针对同一依赖包进行迭代更新，即同一依赖包可拥有多个版本，便于用户对依赖包进行系统化管理。

函数依赖包生成示例请参见[如何制作函数依赖包](#)。

说明

- 依赖包内文件名不能以~结尾。
- 依赖包当前文件限制数为30000。
- 如果函数配置了私有依赖包且依赖包很大的话，建议在函数详情页的“设置 > 常规设置”重新设置函数执行时间，在原基础上增加超时时间。

创建依赖包

步骤1 登录FunctionGraph控制台，在左侧导航栏选择“函数 > 依赖包管理”，进入“依赖包管理”界面。

步骤2 单击的“创建依赖包”，弹出“创建依赖包”对话框。

步骤3 设置以下信息。

表 10-1 依赖包配置参数说明

参数	说明
依赖包名称	自定义的依赖包名称，用于识别不同的依赖包。

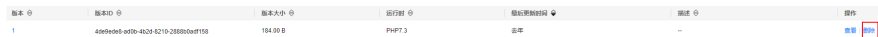
参数	说明
代码上传方式	分为上传ZIP文件和从OBS上传文件。 <ul style="list-style-type: none"> 上传ZIP文件：需单击“添加文件”，上传ZIP文件。 OBS链接URL：需填写“OBS链接URL”，OBS存储链接获取方法请参见OBS对象存储服务。
运行时语言	选择运行时语言。
描述	对于依赖包的描述信息，可以不填。

步骤4 单击“确定”，完成依赖包的创建。默认首次创建的依赖包版本为“1”。

步骤5 单击列表中的依赖包名称，进入版本历史界面，可以查看当前依赖包下的所有版本和版本相关信息。当前支持针对同一依赖包，进行不同版本的系统化管理。

- 单击“创建版本”，填写相关信息，可以创建新的依赖包版本。
- 单击具体的版本号，可以查看版本地址。
- 单击版本号所在行的删除，可以删除该版本。

图 10-1 删除依赖包版本



----结束

配置函数依赖

步骤1 返回函数 workflow 控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数列表界面。

步骤2 单击函数名称，进入函数详情界面。

步骤3 在“代码”页签，单击“添加依赖包”，弹出“选择依赖包”对话框。

步骤4 选择依赖包，单击“确定”。

表 10-2 依赖包配置说明

参数	说明
运行时语言	默认展示当前函数的运行时语言，无法修改。
依赖包源	根据实际业务，选择“公共依赖包”或“私有依赖包”。
依赖包名称	选择当前运行时语言下的依赖包。
版本	选择当前依赖包的具体版本。

📖 说明

- 一个函数最多可添加20个依赖包。
- 除了您自行创建的依赖包（私有依赖包）以外，FunctionGraph还提供了一些常见的公共依赖包，您可以直接选择使用。

----结束

删除依赖包

依赖包当前无法在界面直接删除，若需删除，请删除依赖包下的所有版本。当所有版本全部删除完成后，依赖包会自动删除。

- 步骤1** 返回函数 workflow 控制台，在左侧导航栏选择“函数 > 依赖包管理”，进入“依赖包管理”界面。
- 步骤2** 单击依赖包名称，进入版本历史管理界面。
- 步骤3** 单击版本号所在行的删除，可以删除该版本，存在多个版本请重复此操作。

图 10-2 删除依赖包版本



📖 说明

如果函数正在使用此依赖包，则无法删除。

----结束

10.2 引入依赖库

支持的依赖库说明

FunctionGraph支持引入标准库及第三方依赖库。

- **标准库**
对于标准库，无论是在线编辑或是线下开发打包上传至FunctionGraph，均可以直接在代码中引入，使用其功能。
- **FunctionGraph支持的非标准库**
FunctionGraph内置一些三方件，如表10-3、表10-4所示。像标准库一样，在编写代码时直接引入，使用其功能。

表 10-3 Node.js Runtime 集成的三方件

名称	功能	版本号
q	异步方法封装	1.5.1
co	异步流程控制	4.6.0
lodash	常用工具方法库	4.17.10
esdk-obs-nodejs	OBS sdk	2.1.5

名称	功能	版本号
express	极简web开发框架	4.16.4
fgs-express	在FunctionGraph和API Gateway之上使用现有的Node.js应用程序框架运行无服务器应用程序和REST API。提供的示例允许您使用Express框架轻松构建无服务器Web应用程序/服务和RESTful API。	1.0.1
request	简化http调用，支持HTTPS并默认遵循重定向	2.88.0

表 10-4 Python Runtime 支持的非标准库

模块	功能	版本号
dateutil	日期/时间处理	2.6.0
requests	http库	2.7.0
httplib2	httpclient	0.10.3
numpy	数学计算	pip2.7, numpy==1.16.6 pip3.10, numpy==1.24.2 pip3.9, numpy==1.18.5 pip3.6, numpy==1.18.5
redis	redis客户端	2.10.5
obsclient	OBS客户端	3.0.3
smnsdk	访问公有云smn服务	1.0.1

- 其他第三方库（除了上面表格列举的非标准三方库，FunctionGraph没有内置别的非标准三方库）

将依赖的第三方库打包，上传至OBS桶或在函数界面上传，最后可在函数代码中使用其功能。

引入依赖库示例

Python引入依赖库示例如下：

```
from com.obs.client.obs_client import ObsClient
```

Nodejs引入依赖库示例如下：

```
const ObsClient = require('esdk-obs-nodejs');
```

对于标准库和FunctionGraph支持的非标准库，可以直接引入。

对于FunctionGraph暂没有内置的非标准三方库，通过以下步骤引入。

1. 将依赖的库文件压缩成ZIP包，上传至OBS存储桶，获得依赖包的OBS存储链接。
2. 登录FunctionGraph控制台，在左侧导航栏选择“函数 > 依赖包管理”，进入“依赖包管理”界面。
3. 选择“创建依赖包”，弹出“创建依赖包”对话框。
4. 输入依赖包名称、运行时语言和OBS存储链接，单击“确定”。

📖 说明

OBS存储链接获取方法请参见[OBS对象存储服务](#)。（以下截图仅供参见，具体URL请以实际上传的文件包为准。）

图 10-3 获取 OBS 存储链接

名称	test.zip	存储类别		标准存储	修改存储类别
最后修改时间	2022/05/12 17:03:49 GMT+08:00	大小			
链接	https://.../test.zip	版本号	-		
加密状态	未加密				

图 10-4 设置依赖包

创建依赖包

* 依赖包名称

可包含字母、数字、下划线、点和中划线，长度不超过96个字符。以大/小写字母开头，以字母或数字结尾

* 代码上传方式

上传代码时，如果代码中包含敏感信息（如账户密码等），请您自行加密，以防止信息泄露。

* OBS链接URL

OBS（对象存储服务）代码链接url，文件必须为zip格式。OBS（对象存储服务）[点击进入](#)

* 运行时语言

描述

0/512

5. 进入函数详情页面，在“代码”页签，单击“添加依赖包”，选择4中创建的依赖包，单击“确定”。

图 10-5 添加依赖包

选择依赖包

 一个函数最多添加20个依赖包。没有找到合适的依赖包？ [创建依赖包](#)

* 运行时

* 依赖包源 公共依赖包 私有依赖包

* 依赖包名称

* 版本

 警告

各个依赖包和代码包之间尽量不要有相同的目录或文件，比如依赖包 depends.zip，里面有 index.py 这个文件，如果代码采用在线编辑方式，函数执行入口为 index.handler，这样在函数执行的时候会产生一个代码文件 index.py，跟依赖包里面的 index.py 文件同名，两个文件可能会因覆盖合并而出错。

10.3 公共依赖

10.3.1 什么是公共依赖

公共依赖是华为云为用户直接提供的依赖包，用户可以在函数界面直接导入相关依赖包实现自己代码的业务逻辑。

与私有依赖相比，公共依赖有如下优势：

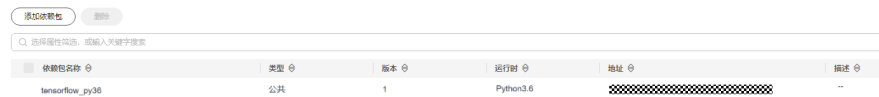
- 公共依赖为用户提供了开箱即用的依赖生态，不需要用户进行繁琐的依赖包构建和上传，仅需在函数界面直接导入即可使用。用户无需过多关注配置代码运行环境，可以投入更多的精力关注代码和业务逻辑
- FunctionGraph 将公共依赖缓存在平台内，相比于私有依赖，用户代码冷启动时不需要承担访问存储服务获取依赖文件的网络时延开销。
- 私有依赖的文件大小限制为 300M，对于较大的依赖包用户需要对文件进行多次拆分并上传；公共依赖可以突破 300M 的文件限制，导入和删除都更加方便。

10.4 公共依赖 Demo

10.4.1 使用 TensorFlow 进行线性回归

首先在 FunctionGraph 页面将 tensorflow 添加为公共依赖

图 10-6 tensorflow 添加为公共依赖



在代码中导入 tensorflow 并使用

```
import json
import random
# 导入 TensorFlow 依赖库
import tensorflow as tf
def handler (event, context):

    TRUE_W = random.randint(0,9)
    TRUE_b = random.randint(0,9)

    NUM_SAMPLES = 100

    X = tf.random.normal(shape=[NUM_SAMPLES, 1]).numpy()
    noise = tf.random.normal(shape=[NUM_SAMPLES, 1]).numpy()
    y = X * TRUE_W + TRUE_b + noise
    model = tf.keras.layers.Dense(units=1)

    EPOCHS = 20
    LEARNING_RATE = 0.002
    print("start training")
    for epoch in range(EPOCHS):
        with tf.GradientTape() as tape:
            y_ = model(X)
            loss = tf.reduce_sum(tf.keras.losses.mean_squared_error(y, y_))

        grads = tape.gradient(loss, model.variables)
        optimizer = tf.keras.optimizers.SGD(LEARNING_RATE)
        optimizer.apply_gradients(zip(grads, model.variables))

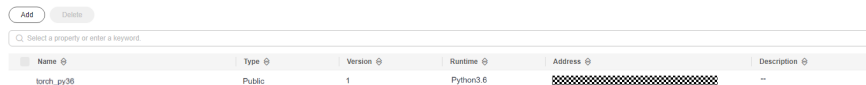
        print('Epoch [{}]/[{}], loss [{}:.3f]'.format(epoch, EPOCHS, loss))
    print("finished")
    print(TRUE_W,TRUE_b)
    print(model.variables)
    return {
        "statusCode": 200,
        "isBase64Encoded": False,
        "body": json.dumps(event),
        "headers": {
            "Content-Type": "application/json"
        }
    }

class Model(object):
    def __init__(self):
        self.W = tf.Variable(tf.random.uniform([1]))
        self.b = tf.Variable(tf.random.uniform([1]))
    def __call__(self, x):
        return self.W * x + self.b
```

10.4.2 使用 pytorch 进行线性回归

在 FunctionGraph 页面将 torch 添加为公共依赖

图 10-7 torch 添加为公共依赖



Name	Type	Version	Runtime	Address	Description
torch_py36	Public	1	Python3.6		--

在代码中导入 torch 并使用

```
# -*- coding:utf-8 -*-
import json
# 导入torch依赖
import torch as t
import numpy as np
def handler (event, context):
    print("start training!")
    train()
    print("finished!")
    return {
        "statusCode": 200,
        "isBase64Encoded": False,
        "body": json.dumps(event),
        "headers": {
            "Content-Type": "application/json"
        }
    }

def get_fake_data(batch_size=8):
    x = t.rand(batch_size, 1) * 20;
    y = x * 2 + (1 + t.randn(batch_size, 1)) * 3
    return x, y

def train():
    t.manual_seed(1000)

    x, y = get_fake_data()

    w = t.rand(1, 1)
    b = t.zeros(1, 1)
    lr = 0.001

    for ii in range(2000):
        x, y = get_fake_data()
        y_pred = x.mm(w) + b.expand_as(y)
        loss = 0.5 * (y_pred - y) ** 2
        loss = loss.sum()

        dloss = 1
        dy_pred = dloss * (y_pred - y)

        dw = x.t().mm(dy_pred)
        db = dy_pred.sum()
        w.sub_(lr * dw)
        b.sub_(lr * db)

    if ii % 10 == 0:
        x = t.arange(0, 20).view(-1, 1)

        y = x.float().mm(w) + b.expand_as(x)
```

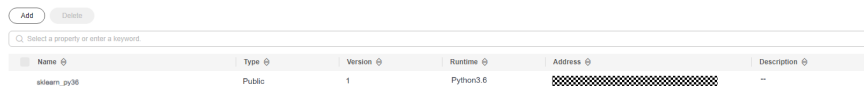
```
x2, y2 = get_fake_data(batch_size=20)

print("w=",w.item(), "b=",b.item())
```

10.4.3 sklearn

在 FunctionGraph 页面将 sklearn 添加为公共依赖

图 10-8 sklearn 添加为公共依赖



Name	Type	Version	Runtime	Address	Description
sklearn_py38	Public	1	Python3.8		--

在代码中导入 sklearn 并使用

```
# 导入 sklearn 相关内容
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
def handler(event, context):
    iris=datasets.load_iris()
    iris_X=iris.data
    iris_y=iris.target

    X_train,X_test,y_train,y_test=train_test_split(iris_X,iris_y,test_size=0.3)

    knn=KNeighborsClassifier()
    knn.fit(X_train,y_train)

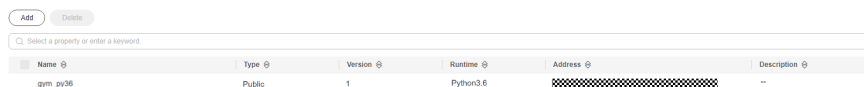
    print(knn.predict(X_test))

    return y_test
```

10.4.4 gym

在 FunctionGraph 页面将 gym 添加为公共依赖

图 10-9 gym 添加为公共依赖



Name	Type	Version	Runtime	Address	Description
gym_py38	Public	1	Python3.8		--

在代码中导入 gym 并使用

```
# 导入 gym 依赖
import gym
env = gym.make('CartPole-v0')
env.reset()

def handler(event,context):
    for _ in range(10):
        env.render()
        observation, reward, done, info, _ = env.step(env.action_space.sample()) # take a random action
        temp = env.step(env.action_space.sample()) # take a random action
        print('observation:{}, reward:{}, done:{}, info:{}'.format(observation, reward, done, info))
    env.close()
```

11 预留实例管理（旧）

什么是预留实例？

函数工作流提供了按量和预留两种类型的实例。

- 按量实例是由函数工作流根据用户使用函数的实际情况来创建和释放，当函数工作流收到函数的调用请求时，自动为此请求分配执行环境。
- 预留实例是将函数实例的创建和释放交由用户管理，当您为某一函数创建了预留实例，函数工作流收到此函数的调用请求时，会优先将请求转发给您的预留实例，当请求的峰值超过预留实例处理能力时，剩余部分的请求将会转发给按量实例，由函数工作流自动为您分配执行环境。

预留实例在创建完成后，会自动加载该函数的代码、依赖包以及执行初始化入口函数，且预留实例会常驻环境，消除冷启动对业务的影响。

📖 说明

用户默认没有权限使用预留实例，如果需要使用预留实例功能，请在[工单系统](#)提交工单添加白名单。

您可以直接创建或者通过函数创建预留实例，两者的区别如下：

表 11-1 两种方式创建预留实例的区别

创建方式	优点	缺点
直接创建	创建步骤简单，易操作	只能创建固定个数的预留实例，可能导致繁忙时预留实例不够用，或者空闲时，预留实例资源浪费
通过函数创建	支持创建不同时间段不同数量的预留实例，避免繁忙时预留实例不够用，或者空闲时，预留实例资源浪费	创建步骤繁杂

直接创建固定个数的预留实例

直接创建固定个数的预留实例前，确保FunctionGraph控制台已存在需要创建预留实例的目标函数，例如Objective-func。

1. 登录FunctionGraph控制台，在左侧导航栏选择“函数 > 预留实例列表”，进入“预留实例列表”界面。
2. 单击“配置预留实例”，弹出“配置预留实例”对话框。
3. 设置以下信息。

表 11-2 预留实例信息

参数	参数说明
应用	选择Objective-func所属的应用。
函数	选择“Objective-func”。
版本	选择Objective-func的版本。
预留实例个数	输入需要创建预留实例的个数。 您可以参见“资源统计”中统计的Objective-func实际运行使用的实例数，设置预留实例个数，或者根据Objective-func过往使用情况，设置预留实例个数。

4. 单击“确定”，完成预留实例的创建。

说明

预留实例创建完成后，只支持修改预留实例的个数。

通过函数创建数量可变的预留实例

用户在不同的时间段，业务使用的实例数可能不一样，您可以通过定时触发器调用函数，为各个时间段设置不同的预留实例数，避免在业务繁忙时未设置预留实例，导致函数被冷启动影响业务或者在业务空闲时设置多个预留实例，导致资源闲置。

通过函数创建数量可变的预留实例前，确保FunctionGraph控制台已存在需要创建预留实例的目标函数，例如Objective-func。

1. 返回函数 workflow 控制台，在左侧导航栏选择“函数 > 函数列表”，进入函数列表界面。
2. 单击“创建函数”，进入“创建函数”界面。
3. 输入以下信息。

参数	参数说明
模板	选择“使用空模板”。
函数名称	输入您自定义的函数名称，用于识别不同的函数。
所属应用	选择“default”。
委托名称	选择“不使用任何委托”。
描述	输入您对函数的描述信息，可以不填。
运行时语言	选择“Python 2.7”。
函数执行入口	输入“index.handler”。

参数	参数说明
代码上传方式	函数配置时，先选择“默认代码”。在配置“页签”选择“在线编辑”，输入如下代码。

```
# -*- coding:utf-8 -*-
import json
import requests
def handler(event, context):
    domainId = "https://{Endpoint}"
    url = "/v2/{project_id}/fgs/functions/{func-urn}/reservedinstances"
    token = context.getToken()
    requrl = domainId + url
    headerdata = {"Content-Type":"application/json","x-auth-token":token}
    r = requests.put(requrl, data=event["user_event"], headers=headerdata,verify=False)
    return r.json
```

注意将上述代码中参数部分根据实际情况进行替换：

- Endpoint: Objective-func所在的终端节点，可以从[地区和终端节点](#)获取。
- project_id: Objective-func所在的[项目ID](#)。
- func-urn: Objective-func的URN。

4. 单击“创建函数”，完成函数的创建。
5. 在“配置”页签，单击“创建委托”，进入“委托”界面。
6. 创建“FunctionGraph User”权限的委托，具体参见[配置委托权限](#)。
7. 返回“配置”页签，在“委托名称”中选择6中创建的委托，单击“保存”，保存配置的委托。
8. 在“触发器”页签，单击“创建触发器”，弹出“创建触发器”对话框。
9. 输入以下信息。

参数	参数说明
触发器类型	选择“定时触发器（TIMER）”。
定时器名称	输入您自定义的定时器名称，用于识别不同的定时器。
触发规则	选择“Cron表达式”，根据实际情况输入触发规则。
是否开启	默认选择开启，无需修改。
附加信息	结合“触发规则”，输入不同时间段需要的预留实例数。

图 11-1 创建定时触发器

创建触发器

触发器类型 ?

可以使用TIMER的计划事件功能定期调用您的代码，可以指定固定频率（分钟、小时、天数）或指定Cron表达式定期调用函数。
DDS、GAUSSMONGO、DIS、LTS、Kafka、TIMER触发器可创建数加起来最多10个，您已创建0个。

* 定时器名称

支持字母、数字、下划线和中划线，必须以字母开头，且长度不能超过64个字符

* 触发规则 固定频率 Cron表达式

[了解Cron表达式](#)

* 是否开启

附加信息 ?

0/2,048

如图11-1所示，创建了一个每隔3分钟创建2个预留实例的定时触发器。

10. 单击“确定”，完成触发器的创建。

触发器创建完成后，Objective-func通过您创建的触发规则在不同时间段，创建数量不同的预留实例数。

12 预留实例管理

概述

函数 workflow 提供了按量和预留两种类型的实例。

- 按量实例是由函数 workflow 根据用户使用函数的实际情况来创建和释放，当函数 workflow 收到函数的调用请求时，自动为此请求分配执行环境。
- 预留实例是将函数实例的创建和释放交由用户管理，当您为某一函数创建了预留实例，函数 workflow 收到此函数的调用请求时，会优先将请求转发给您的预留实例，当请求的峰值超过预留实例处理能力时，剩余部分的请求将会转发给按量实例，由函数 workflow 自动为您分配执行环境。

预留实例在创建完成后，会自动加载该函数的代码、依赖包以及执行初始化入口函数，且预留实例会常驻环境，消除冷启动对业务的影响。（注意：不要依赖预留实例本身的初始化函数去执行一次性业务。）

预留实例当前支持配置固定数量的预留实例，也支持配置定时伸缩的预留实例、配置按指标弹性伸缩的预留实例和配置智能推荐的预留实例。

说明

用户默认没有权限使用指标策略和智能推荐策略，如果需要使用该功能，请在[工单系统](#)提交工单添加白名单。

配置固定数量的预留实例

直接创建固定个数的预留实例前，确保 FunctionGraph 控制台已存在需要创建预留实例的目标函数。

1. 登录[函数 workflow 控制台](#)，在左侧的导航栏选择“函数 > 函数列表”。
2. 选择待配置的函数，单击进入配置详情页。
3. 选择“设置 > 并发”，单击“添加”，开始配置。

图 12-1 单击“添加”

预留实例策略配置 ?

函数 workflow 支持固定数量的预留策略和弹性的预留策略。



4. 参见表12-1，填写参数。

您可以给函数对应的版本或者别名创建指定数量的预留实例，其中预留实例的数量不能超过并发实例数配额和单函数最大实例数。

图 12-2 基础配置

基础配置

函数名称 test-lts

类型 版本 别名

选择版本 latest

预留实例数 0

配置预留实例数后，函数工作流会为您创建固定数目的函数实例，并且在您将预留实例数设置为 0 之前预留实例会持续运行。

闲置模式

预留实例在无调用的时候暂停CPU，节省资源，降低费用成本

表 12-1 基础配置说明

参数	说明
函数名称	展示当前配置预留实例的函数的名称。
类型	根据实际业务情况，选择“版本”或“别名”。
选择版本	仅当类型选择“版本”时，需设置此参数。
选择别名	仅当类型选择“别名”时，需设置此参数。
预留实例数	设置最小实例数，输入值不能超过1000。配置最小实例数后，函数工作流会为您创建固定数目的函数实例，并且在您将最小实例数设置为0之前预留实例会持续运行。
闲置模式	开启此参数，表示预留实例在无调用的时候暂停CPU，节省资源，降低费用成本。

📖 说明

- 别名和对应的版本不可以同时配置预留实例。比如，latest版本对应的别名为1.0，在latest版本下进行了预留实例配置，则在别名1.0下不能再进行预留实例配置，反之同理。
 - 闲置模式开启后，因为在最初阶段会涉及实例的初始化及模式转换，因此该段时间会以预留实例的非闲置模式计费标准进行计费。
 - 当函数调用并发数大于预留实例数量时，超出部分会分配给按量实例，这部分流量仍然有冷启动。
5. 配置完成后，单击“确定”，在“预留实例策略配置”列表展示已添加的“策略配置”。

图 12-3 列表展示



配置定时伸缩的预留实例

用户配置预留实例时，能够配置指定的时间段、cron表达式及其对应的预留实例数量。函数服务能够在该时间段中，根据cron表达式更新预留实例的数量，如果时间段超过了该时间段，则将预留实例数量调整到配置的固定值的预留实例数量。

1. 参见表12-1进行基础配置，完成后单击“添加策略”，进行弹性预留策略配置。

图 12-4 添加策略



2. 参见表12-2，填写参数。

图 12-5 添加策略

策略名称

Cron表达式(UTC)

生效时间

策略的生效时间范围

预留实例数

当前策略生效时创建的预留实例的个数
预留实例数必须大于或等于基础配置里的预留实例数。

函数并发执行实例数

并发数(个) 最近一小时

表 12-2 弹性策略配置说明

参数	说明
策略名称	自定义策略名称。
Cron表达式(UTC)	您可以参见 Cron表达式规则 ，填写此参数。
生效时间	生效时间为本地时间，即cron表达式的生效时间窗。 只有当时间在时间窗内时，该弹性策略才会生效，当该函数的所有弹性策略的生效时间窗都不生效时，那么预留实例数就会还原到基础配置中的最小实例数。
预留实例数	需要创建的预留实例数。 根据实际业务场景，填写当前策略生效时创建的预留实例的个数。 说明 最小实例数必须大于或等于基础配置里的最小实例数。

- 配置完成后，单击“确定”，在“预留实例策略配置”列表展示已添加的“策略配置”。

图 12-6 列表展示



- 单击“操作 > 编辑”，修改弹性策略信息、添加策略。
- 单击“操作 > 删除”，删除版本或别名下的预留实例策略。
- 预留实例将根据添加的弹性策略配置执行，您可以在“预留实例策略配置”列表，单击“限定符”，选择“弹性策略名称”，查看函数并发执行实例数。

说明

时间业务是可以配置多条定时策略，如配置早晨8点的时候，配置策略更新预留实例数为100，而21点的时候配置策略更新为10个。

配置按指标弹性伸缩的预留实例

用户配置预留实例时，能够根据业务的指标（当前只支持用户并发数）动态调整函数的预留实例数，配置指标策略时，需要为函数配置委托且该委托包含AOM服务的指标查询权限和函数服务的查询配置权限。

步骤1 参见表12-1进行基础配置，完成后单击“添加策略”，进行弹性预留策略配置。

图 12-7 添加策略

基础配置

函数名称 test-lts

类型 版本 别名

选择版本 latest

预留实例数 1

配置预留实例数后，函数工作流会为您创建固定数目的函数实例，并且在您将预留实例数设置为 0 之前预留实例会持续运行。

闲置模式

预留实例在无调用的时候暂停CPU，节省资源，降低费用成本

弹性预留策略 添加策略

策略名称	预留实例数	策略类型
暂无表格数据		

步骤2 参见表12-3，填写参数。

图 12-8 配置指标策略

< 返回 | 添加策略

策略类型 定时策略 指标策略 智能推荐策略

定时策略会在指定时间修改预留实例的个数。指标策略会根据设置的利用率指标调节修改预留实例的个数，智能推荐策略会根据预测模型动态调节预留实例个数。

策略名称

利用率目标值(%)
利用率等于实际使用实例个数与总预留实例个数的比值。

最小预留实例数
当策略生效时创建的预留实例的最小个数
最小预留实例数必须大于或等于基础配置里的预留实例数。

最大预留实例数
当策略生效时创建的预留实例的最大个数
最大预留实例数必须大于或等于最小预留实例数。-1表示不限制(预留实例的最大配额: 90)

表 12-3 指标策略配置说明

参数	说明
策略名称	自定义策略名称。
利用率目标值	取值为1-99，即用户实际使用到的预留实例数与该函数总的预留实例数的比例。当比例高于目标值时，函数服务会逐步扩容预留实例数；而当比例低于目标值时，函数服务会直接创建对应数量的预留实例，调整的周期为1分钟一次（指标生成也是1分钟一次，所以调整预留实例会滞后1-2分钟）。
预留实例数	需要创建的预留实例数。 指标策略更新预留实例数时的最新预留实例数，如根据策略只需要创建1个预留实例，而最小实例数配置为5时，则实际会创建5个预留实例。 说明 最小实例数必须大于或等于基础配置里的最小实例数。

表 12-4 委托权限配置

服务	普通权限	细粒度权限
FunctionGraph 服务	FunctionGraph ReadOnlyAccess	functiongraph:function:getConfig

步骤3 配置完成后，单击“确定”，在“预留实例策略配置”列表展示已添加的“策略配置”。

图 12-9 列表展示



步骤4 单击“操作 > 编辑”，修改弹性策略信息、添加策略。

步骤5 单击“操作 > 删除”，删除版本或别名下的预留实例策略。

步骤6 预留实例将根据添加的弹性策略配置执行，您可以在“预留实例策略配置”列表中，单击“限定符”，选择“弹性策略名称”，查看函数并发执行实例数。

说明

定时策略和指标策略可以一起添加。当一起添加时，该类型的策略的预留实例数的最小值不得低于另外一种策略最新更新的预留实例数。如8:59分，指标策略更新预留实例数为9个，而10点的定时策略配置的预留实例数是5个，那么10点最终更新的预留实例为9个；而下一条定时策略是11点生效，那么10点到11点指标策略更新的预留实例数最小不会低于5个。

----结束

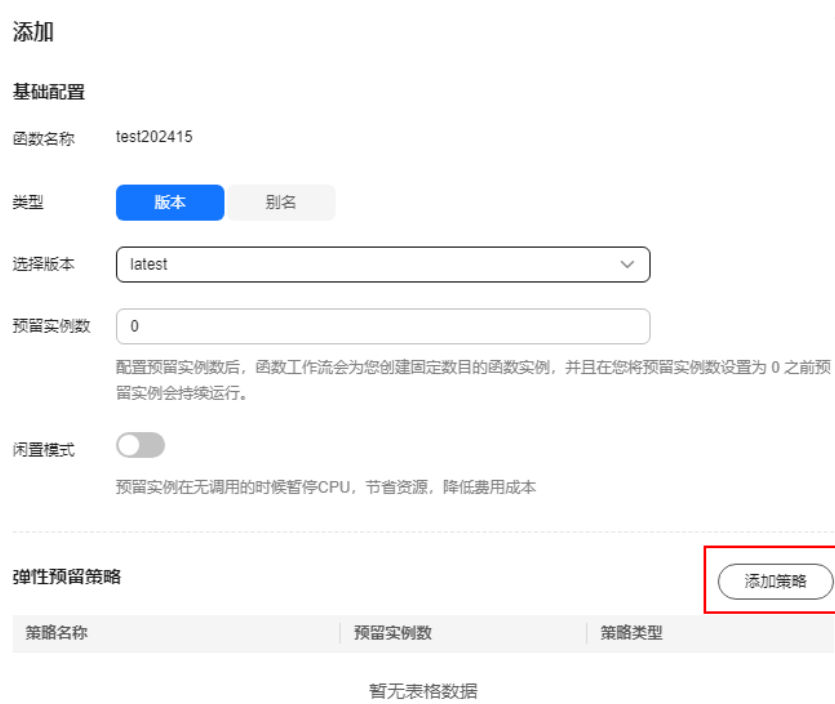
配置智能推荐的预留实例

FunctionGraph基于特征画像与负载预测技术，提供了预留实例的智能推荐策略，使预留实例随负载模式动态变化，波峰时提前扩容，波谷时释放多余预留实例。

用户配置预留实例时，能够选择智能推荐策略，支持高性能、均衡、低成本三种选项，由系统根据用户负载模式，基于负载预测动态调整预留实例数量，适应负载的波峰波谷变化，并对相应的预留实例成本与性能提供直观展示。（注：智能推荐策略与其他预留实例弹性策略不能共存，且同一版本或别名只能存在一个。）

1. 参见下图单击“添加策略”，进行弹性预留策略配置。

图 12-10 添加策略



2. 选择“智能推荐策略”，用户根据展示的预留实例性能与成本图结合自身需求，可选择高性能、均衡、低成本三种选项之一。

图 12-11 智能推荐策略



3. 选择完成后，单击“确定”，在“预留实例策略配置”列表展示已添加的“策略配置”。

图 12-12 预留实例策略配置



策略名称	预留实例数量	类型	策略名称	策略名称	操作
test	0	标准	strategy-name	策略名称	编辑 删除

4. 单击“操作 > 编辑”，查看和修改当前弹性策略信息。
5. 单击“操作 > 删除”，删除版本或别名下的预留实例策略。
6. 预留实例将根据选择的智能推荐策略选项执行，您可以在“预留实例策略配置”列表，单击“限定符”，选择“弹性策略名称”，查看预留实例成本及性能。

13 函数流管理

13.1 函数流简介

📖 说明

当前仅“华东-上海一、亚太-新加坡”区域支持函数流功能。

函数流是一个面向无服务器计算领域，编排无服务器分布式应用的工作流服务。基于该服务，用户可以通过Low Code以及可视化的方式将多个独立的无服务器函数用顺序、分支、并行等方式轻松编排成一个完整的应用，并提供监控和管理平台，用于诊断和调试应用。

本章节主要介绍函数流组件、组件编排规则、表达式运算符和配置示例。

组件说明

函数流提供多种类型的组件，用户可以通过拖拽组件、配置组件和连接组件进行可视化编排，实现函数任务流的编排。使用函数流功能，请先了解[表13-1](#)。

表 13-1 组件说明

类型	名称	说明
服务组件	函数	FunctionGraph函数，如何创建函数请参见 创建函数 。
	EG	事件网格服务（EventGrid），EG节点会发布已配置的事件至指定的EG事件通道，如何创建EG资源请参见 事件网格相关文档 。
流程控制器	回调节点	通过人工干预实现对执行中函数流的条件控制，函数流将阻塞在回调节点，直到用户调用回调接口以继续函数流执行，从而达到人工审批的效果。
	子流程	把已创建的“函数流”任务作为“子流程”组合成一个新的函数流任务。
	并行分支	用于创建多个并行分支的控制器，以便同时执行多个分支任务，并可根据分支执行结束后控制下一步流程。

类型	名称	说明
	开始节点	只能加入触发器，用于标识流程的开始，一个流程只能有一个开始节点。
	异常处理	用于控制函数执行失败后的下一步流程。
	循环节点	用于对数组中每个元素进行循环处理。每次循环会执行一次循环内部的子流程。
	时间等待	用于控制当前流程在指定时间延迟后再调用下一个流程。
	服务节点	用于对多个函数构成的复杂操作进行抽象，可以将多个函数操作合并成一个原子节点进行管理。
	条件分支	用于根据条件判断是否执行下一分支。
	结束节点	用于标识流程的结束。

编排规则

- 设计的函数流必须是一个有向无环图，从开始节点出发，开始节点后续必须且只能连接一个节点（除了异常处理和结束节点）；流程必须在某一个节点结束，结束流程有两种形式：
 - 流程中存在的节点没有任何后继节点，且后续节点非条件分支，并行分支或开始节点。
 - 流程中存在结束节点，且结束节点后续无其他节点。
- 组件设计规则

表 13-2 触发器和函数和 EG

参数	说明	创建函数流时，是否必选
触发器	<ul style="list-style-type: none"> 当前允许流程中配置0-10个触发器。 触发器必须配置在开始节点内。 触发器不允许连接其他任何节点，也不允许被其他节点连接。 	否
函数	<ul style="list-style-type: none"> 当前允许流程中配置0-99个函数节点。 当函数连接异常处理节点时，最多可以再连接一个非开始节点和非异常处理节点。 当函数不连接异常处理节点时，只能连接一个非开始节点。 	否

参数	说明	创建函数流时，是否必选
EG	<ul style="list-style-type: none"> 当前允许流程中配置0-10个EG节点。 当EG节点连接异常处理节点时，最多可以再连接一个非开始节点和非异常处理节点。 当EG节点不连接异常处理节点时，只能连接一个非开始节点。 	否

表 13-3 流程控制器

参数	说明	创建函数流时，是否必选
回调节点	回调节点限制规则参考表13-2中函数参数，但回调节点不可为服务节点的子节点	否
子流程	该节点选择已创建的函数流任务。	否
并行分支	<ul style="list-style-type: none"> 用于标识节点后面的分支会并行执行。 后继节点允许连接1-20个节点（除了异常处理，开始节点和结束节点），至少连接一个节点。 	否
开始节点	<ul style="list-style-type: none"> 用于标识流程开始，每个流程必须有且只能有一个开始节点。 开始节点后面必须接1个节点，后续节点类型不能是结束节点或者异常处理。 	必选
异常处理	后面可以接0-10个节点，后继节点不能是开始节点，结束节点和异常处理节点。	否
循环节点	<p>用来对数组中每个元素进行循环处理。每次循环会执行一次循环内部的子流程。</p> <p>循环节点内部子流程需要满足如下规则：</p> <ol style="list-style-type: none"> 只能有一个起始节点（没有前驱节点），起始节点只能使用函数，时间等待节点。 循环节点内部只允许编排函数，时间等待，异常处理节点。 	否
时间等待	后面可以连接0个或1个节点，节点类型不能是开始节点和异常处理节点。	否
服务节点	服务节点由多个函数节点组成，后续节点可以是结束节点或异常处理节点。	否
条件分支	后面可以连接2-20个后继节点，后继节点类型不能为开始节点，结束节点和异常处理节点。	否

参数	说明	创建函数流时，是否必选
结束节点	后面不能接任何节点。	否

表达式运算符说明

异常处理和条件分支的表达式结构为 [JsonPath] + [逻辑运算符] + [对比数据]，简单示例：\$.age >= 20

JsonPath说明

Operator	Supported	Description
\$	Y	执行查询的root，所有正则表达式由此启动。
@	Y	过滤正在处理的当前位置。
.	Y	子节点。
[(,)]	Y	数组索引。
[start:end]	Y	数组切片运算符。
[?()]	Y	过滤表达式。表达式必须计算为布尔值。

参见示例

- 简单取值：JSON数据样例

```
{
  "fruits": [ "apple", "orange", "pear" ],
  "vegetables": [{
    "veggieName": "potato",
    "veggieLike": true
  },
  {
    "veggieName": "broccoli",
    "veggieLike": false
  }
]}
```

\$.fruits表达式含义：取出fruits下对应的所有value。

\$.fruits解析结果：["apple","orange","pear"]

- 简单过滤：JSON数据样例

```
{
  "fruits": [ "apple", "orange", "pear" ],
  "vegetables": [{
    "veggieName": "potato",
    "veggieLike": true
  },
  {
    "veggieName": "broccoli",
    "veggieLike": false
  }
]}
```



```
    ]}
  }
```

表达式: `$.vegetables[?(@.veggieLike == true)].veggieName`

表达式含义: 取出key值vegetables对应的所有value, 并根据过滤条件输出veggieLike为True的veggieName。

取值结果: [potato]

逻辑运算符说明

使用以下数据作为例子中的输入参数:

```
{
  "name" : "apple",
  "weight": 13.4,
  "type": [3,4,6,8],
  "obj": {
    "a" : 1
  }
}
```

支持的运算符如下:

符号	作用	例子	返回值	备注
==	相等	<code>\$.name == 'apple'</code>	true	支持的数据类型包括: int,float,string, bool,nil
!=	不等	<code>\$.name != 'apple'</code>	false	支持的数据类型包括: int,float,string, bool,nil
<	小于	<code>\$.weight < 12</code>	false	只支持数字类型
>	大于	<code>\$.weight > 12</code>	true	只支持数字类型
<=	小于等于	<code>\$.weight <= 13.4</code>	true	只支持数字类型
>=	大于等于	<code>\$.weight >= 13.4</code>	true	只支持数字类型
*	通配符	<code>\$.weight == '*'</code>	true	只支持在== 比较中使用
	或	<code>\$.name == 'apple' \$.weight < 12</code>	true	支持使用 () 的复杂与或逻辑
&&	且	<code>\$.name == 'apple' && \$.weight < 12</code>	false	支持使用 () 的复杂与或逻辑

说明

- 字符串格式常量需要使用 ‘ ’ 包含，例如： ‘apple’
- jsonpath表达式中不能出现上述保留字符 '=', '!', '<', '>', '|', '&'

配置示例

示例一：并行分支使用参考

1. 新建三个函数， Runtime均使用python 3.9， 代码功能及内容如下所示 。

- 函数1： 函数执行返回result的值为函数调用事件内的input输入值

```
import json
def handler (event, context):
    input = event.get('input',0)
    return {
        "result": input
    }
```

- 函数2： 函数执行返回result的值为函数调用事件内的input输入值+2的结果值

```
import json
def handler (event, context):
    input = event.get('input',0)
    return {
        "result": input+2
    }
```

- 函数3： 函数执行返回result的值为函数调用事件内的input输入值平方的结果值

```
import json
def handler (event, context):
    input = event.get('input',0)
    return {
        "result": input*input
    }
```

2. 在函数流编排区域拖拽组件，“并行分支”节点配置如下。

图 13-1 并行分支节点配置



3. 函数节点配置如表2 触发器和函数所示，在3个函数节点均需配置

图 13-2 函数节点配置



- 保存函数流，启动执行时，定义输入值如下所示。

```
{
  "input": 3
}
```

- 单击函数流任务名称，查看执行结果。

图 13-3 执行结果



示例二：服务节点使用参考：串行模式

- 在函数编排区域编排与拖拽组件，服务节点选择“串行模式”。

图 13-4 服务节点配置



- 函数节点分别选择示例一中创建的函数2、函数3，函数流的配置如下。

图 13-5 函数 2 配置



图 13-6 函数 3 配置



- 保存函数流，启动执行时，定义输入值如下所示。

```
{
  "input": 3
}
```

- 单击函数流任务名称，查看执行结果。

图 13-7 执行结果



示例三：服务节点使用参考：并行模式

- 新建两个函数， Runtime均使用Python 3.9， 代码内容相同。

```
import json
def handler (event, context):
    print(event)
    return {"result": "success"}
```

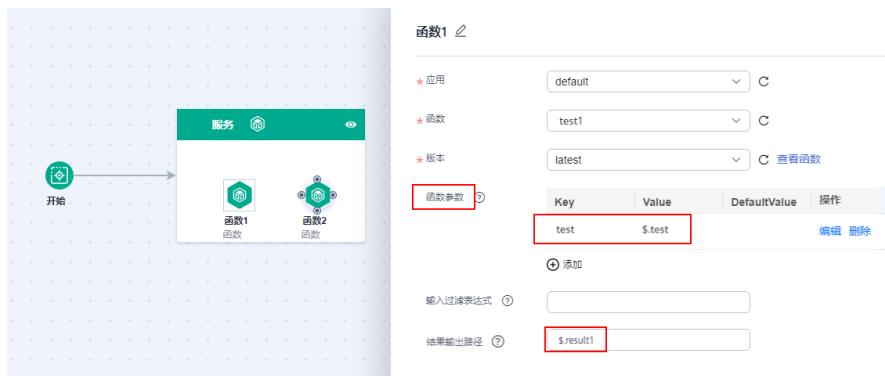
- 在函数流编排区域拖拽组件， 服务节点的配置如下。

图 13-8 服务节点配置



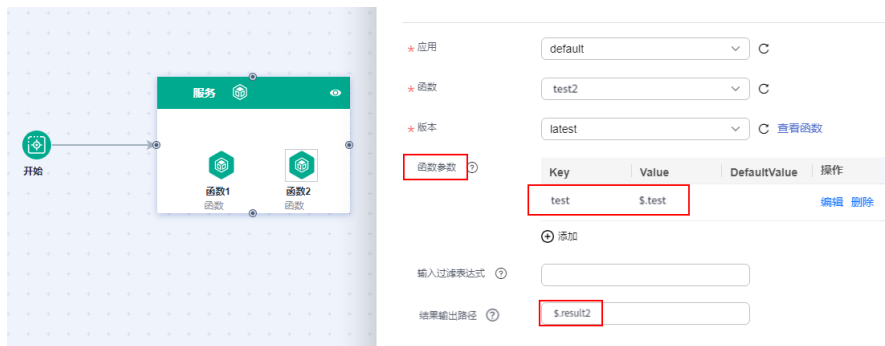
- 函数1节点配置如下。

图 13-9 函数 1 节点配置



4. 函数2节点配置如下。

图 13-10 函数 2 节点配置



5. 保存函数流，启动执行时，定义输入值如下所示。

```
{
  "test": 123
}
```

6. 单击函数流任务名称，查看执行结果，可以看到两个函数执行结果合并，分别为result1和result2。

图 13-11 执行结果



说明

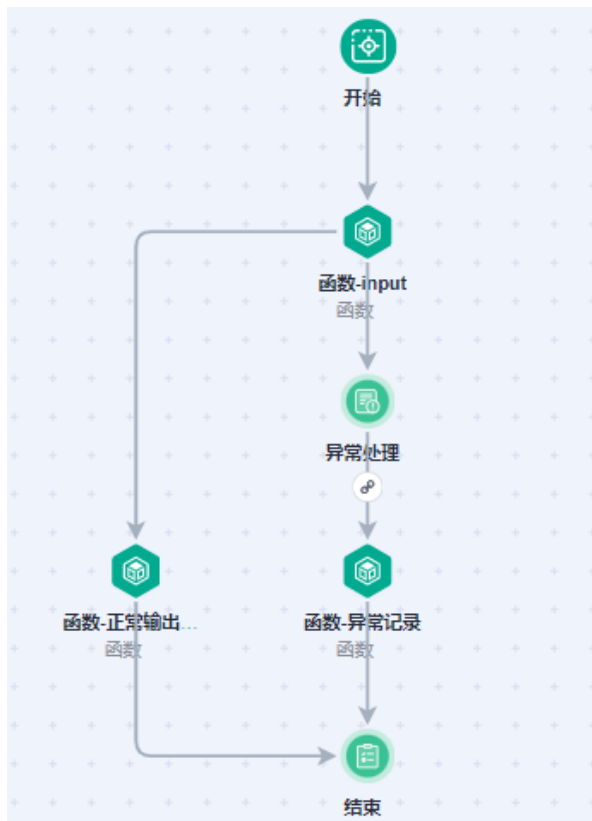
如果两个函数执行返回的输出值结构一致，会导致函数执行结果被覆盖。

示例四：异常处理使用参考

当函数流里面的函数执行异常时，可以通过“异常处理”来处理执行失败的函数并可添加重试。函数执行失败可分为多种情况：函数执行异常；函数内部业务失败并在返回内容中添加了错误码，例如status，200代表成功，500和404等代表失败。

1. 在函数流编排区域拖拽组件，各节点功能如下。

- 函数-input: 从event取出input输入值, 作为函数返回status的输出值;
- 异常处理: 开启重试机制, 当函数返回的status为500或404时进行重试, 重试间隔1s, 最大重试次数8次;
- 函数-异常记录: 当经过8次重试函数返回的status依旧为500或404时, 进行异常记录;
- 函数-正常输出: 如果“函数-input”返回的status不为500或404时, 执行“函数-正常输出”。



2. 配置异常处理, 重试条件: `$.status==500||$.status==404`。

图 13-12 配置异常处理

异常处理 [🔗](#)

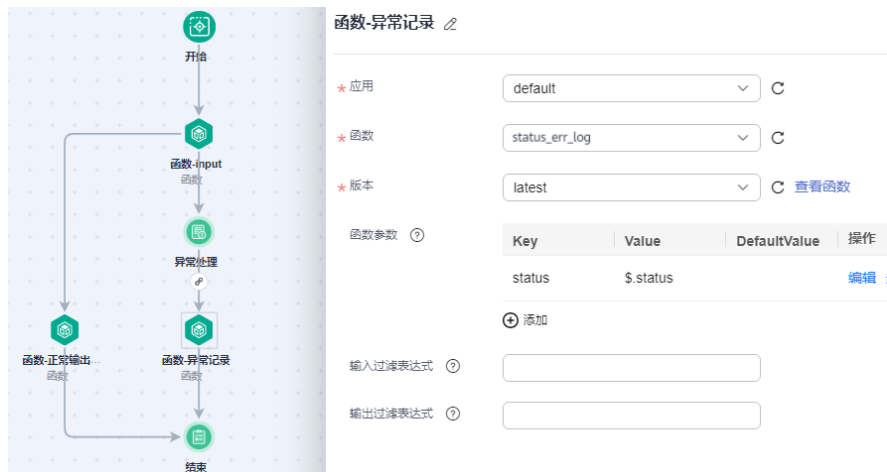
* 是否重试

* 重试条件(JSONPath) [?](#)

重试间隔(1-30秒) [?](#)

最大重试次数(1-8) [?](#)

3. 添加重试之后依旧失败的处理逻辑即“函数-异常记录”。



4. 输入200执行成功。

输入值

```
1 {
2   "input": 200
3 }
```

输出值

```
1 {
2   "result": 200
3 }
```

节点日志

节点名称	状态	请求ID	耗时	开始时间
函数-传入status	执行成功	32d3306b6f4670a312066c3fd2b37...	402ms	2022/04/12 19:55:34...
函数-正常输出结果	执行成功	9bef34254330b2e8d1c134210205b...	474ms	2022/04/12 19:55:34...

5. 输入500，异常处理逻辑进行重试和记录。

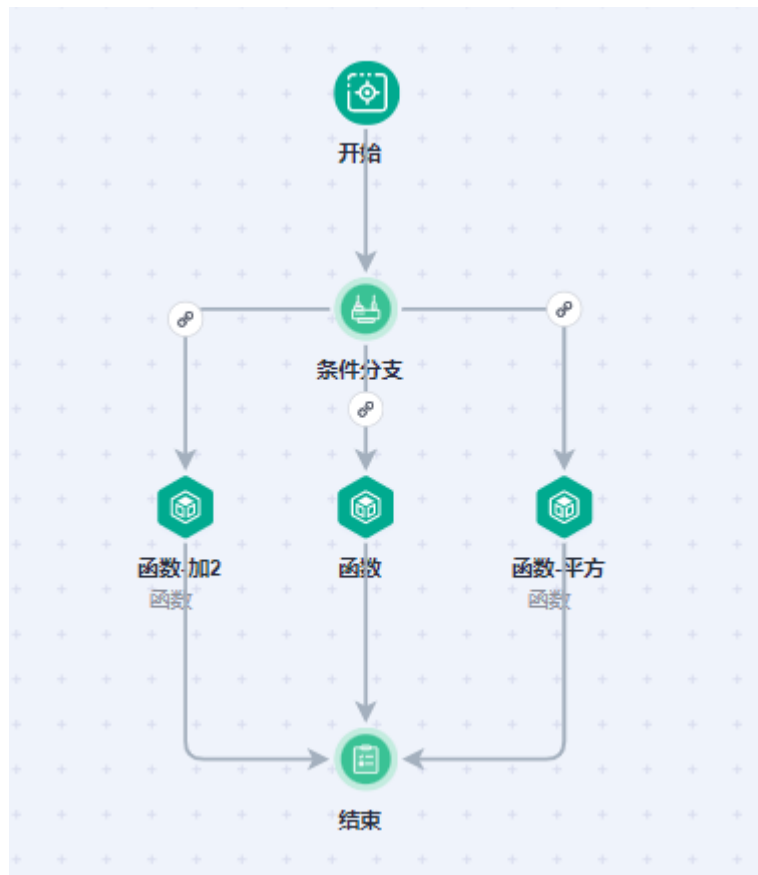
输入值	输出值
<pre>1 { 2 "input": 404 3 }</pre>	<pre>1 { 2 "result": "log err 404" 3 }</pre>

节点日志

节点名称	状态	请求ID	耗时	开始时间
函数-传入status	异常捕获	0bafc4552314b93dce481bdd1e0b...	8466ms	2022/04/12 19:53:40...
函数-异常记录	执行成功	9dd36bbda1ada20b8358e7b7d533...	458ms	2022/04/12 19:53:48...

示例五：条件分支节点使用参考

- 通过函数流的“条件分支”实现：
 - 如果输入值小于2时，则result为输入值加2；
 - 如果大于2则平方；
 - 如果等于2则输出原值；
- 具体设计如下：

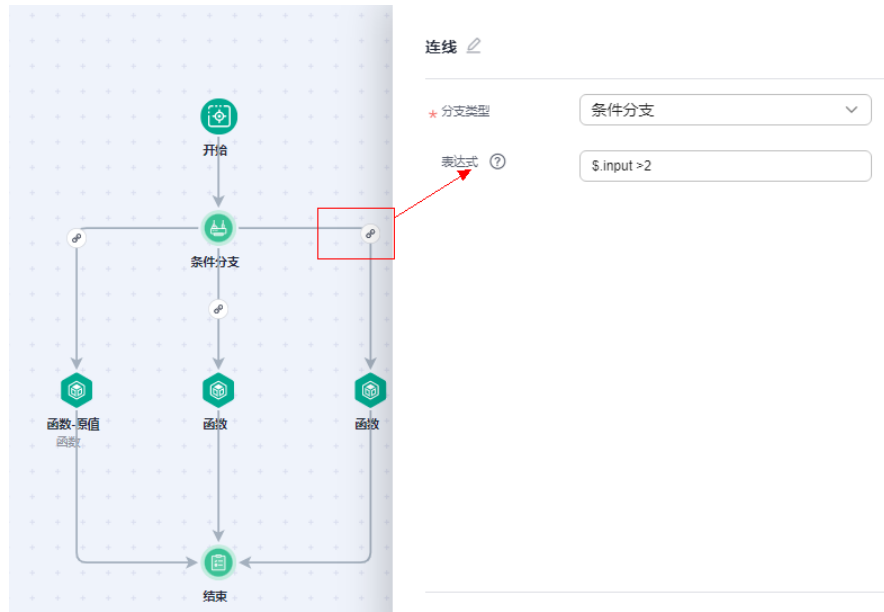


分支类型: 默认分支

连线

分支类型: 条件分支

表达式: \$.input <2



输入3，输出3的平方：

输入值	输出值
<pre>1 { 2 "input": 3 3 }</pre>	<pre>1 { 2 "result": 9 3 }</pre>

- 输入2，则输出原值：

输入值	输出值
<pre>1 { 2 "input": 2 3 }</pre>	<pre>1 { 2 "result": 2 3 }</pre>

输入1，则加2：

```
输入值
1 {
2   "input": 1
3 }

输出值
1 {
2   "result": 3
3 }
```

示例六：时间等待节点使用参考

- 通过函数流的“时间等待”组件实现对函数的延迟调用，函数流设计实现如下：



- 时间等待默认为60s，如下：

时间等待

* 延迟时间 (秒)

- 执行如下：

输入值	输出值
<pre>1 { 2 "input": "test" 3 }</pre>	<pre>1 { 2 "invoke_time": "2022-04-12 15:46:24" 3 }</pre>

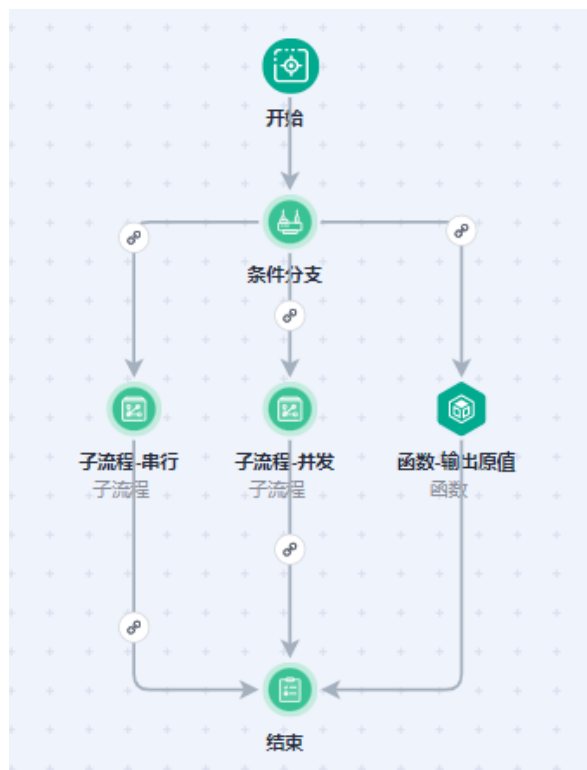
节点日志

节点名称	状态	请求ID	耗时	开始时间
函数-打印时间1	执行成功	afc441e9ba1a6f0593097da897282...	471ms	2022/04/12 15:45:23...
时间等待	执行成功	97366f04dc1c2830ef780a163384a...	60001ms	2022/04/12 15:45:23...
函数-打印时间2	执行成功	24e38165d8d9df8270ddad55cc17b...	462ms	2022/04/12 15:46:23...

示例七：子流程使用参考

函数 workflow 服务可以把一些“函数流”作为“子流程”组合成一个新的函数流，这样可以抽取出一一些函数流作为公共流进行使用，减少函数编排的重复开发工作流。

- 设计一个函数流：当输入值 $input < 2$ 时，执行子流程串行执行（输入值先加2再平方）；当输入值 $input = 2$ ，走默认分支，输出原值；当输入值 $input > 2$ 时，执行子流程并行执行（并行输出原值、输入值加2和输入值的平方值），具体如下：





连线 [↗](#)

* 分支类型

表达式

子流程-串行 [↗](#)

* 选择子流程 [C 查看子流程](#)

* 是否等待子流程完成

输入过滤表达式

输出过滤表达式



连线 [↗](#)

* 分支类型

表达式

子流程-并发

- * 选择子流程: flow_yin_bingfa [查看子流程](#)
- * 是否等待子流程完成:
- 输入过滤表达式:
- 输出过滤表达式:

函数-输出原值

- * 应用: default
- * 函数: maths_original_value
- * 版本: latest [查看函数](#)

函数参数	Key	Value	DefaultValue	操
	input	\$.input		编

输入过滤表达式:
输出过滤表达式:

- 输入3，走子流程并行，结果如下：

输入值	输出值
<pre>1 { 2 "input": 3 3 }</pre>	<pre>1 { 2 "result": [3 { 4 "result": 3 5 }, 6 { 7 "result": 5 8 }, 9 { 10 "result": 9 11 } 12] 13 }</pre>

- 输入1，走子流程串行，结果如下：

输入值	输出值
<pre>1 { 2 "input": 1 3 }</pre>	<pre>1 { 2 "result": 9 3 }</pre>

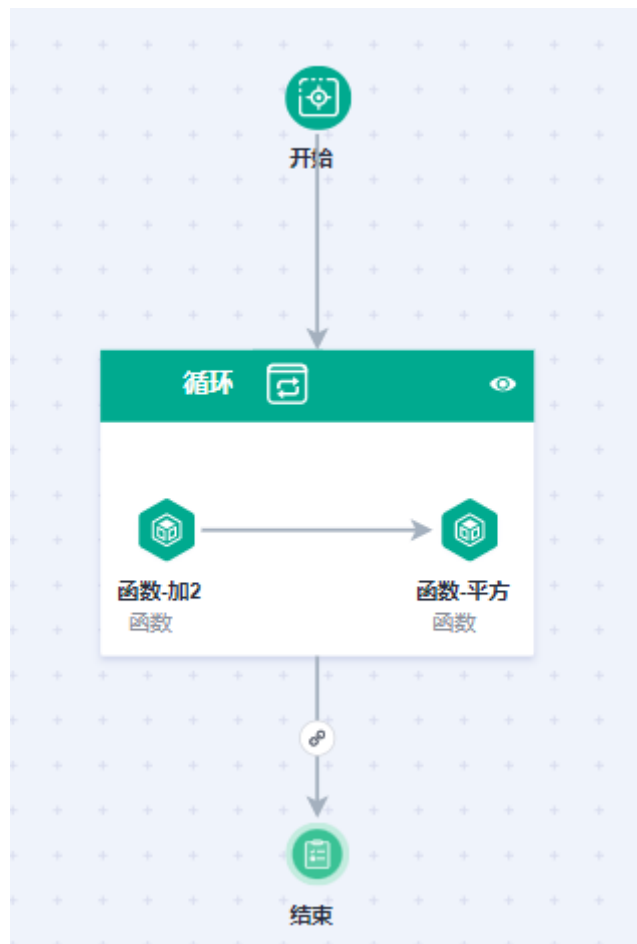
- 输入2，走默认分支，输出原值，结果如下：

输入值	输出值
1 {	1 {
2 "input": 2	2 "result": 2
3 }	3 }


示例八：循环节点使用参考

使用“循环”组件来对输入数组中每个元素进行循环处理。每次循环会执行一次循环内部的子流程。

- 设计一个函数流，输入一个整数数组，通过循环对整数数组的每一个元素先加2再平方，具体实现如下：




- 配置遍历数组路径\$.inputs，迭代变量名称\$.item



循环

- * 遍历数组路径
- * 迭代变量名称
- * 结果输出路径
- 并发迭代数目
- 并发迭代时间间隔(秒)
- 输入过滤表达式
- 输出过滤表达式

- 函数-加2配置，函数参数配置中的Value要与“循环”里面的迭代变量名称一致：



函数-加2

- * 应用
- * 函数
- * 版本

Key	Value	DefaultValue	操作
input	\$.item		保存 取消

添加

输入过滤表达式

输入值

```

1 {
2   "inputs": [
3     1,
4     2,
5     3,
6     4
7   ]
8 }
```

输出值

```

1 {
2   "result": [
3     {
4       "result": 9
5     },
6     {
7       "result": 16
8     },
9     {
10      "result": 25
11    },
12    {
13      "result": 36
14    }
15  ]
16 }
```

13.2 创建函数流任务

本章节主要介绍如何创建函数流任务和编排函数流任务。您可以根据实际业务场景来创建标准函数流或快速函数流。

- 标准模式面向普通的业务场景，支持长时间任务，支持执行历史持久化和查询，只支持异步调用，在函数流运行记录页面查询执行结果。

- 快速模式面向业务执行时长较短，需要极致性能的场景，只支持流程执行时长低于5分钟的场景，不支持执行历史持久化（比如不支持查询执行节点的历史信息），支持同步和异步调用。通过同步执行函数流接口进行函数流的同步执行，接口直接返回函数流执行结果，同时日志页面查看上报到LTS的函数流执行日志。

📖 说明

快速函数流限时免费，欢迎体验！

前提条件

- 已经在FunctionGraph控制台创建函数，创建过程请参见[创建函数](#)。
- 使用函数流功能前，请先了解[组件说明](#)、[编排规则](#)和[表达式运算符说明](#)。

操作步骤

步骤1 登录FunctionGraph控制台，进入“函数流”页面。

步骤2 在“函数流”页面，单击“创建标准函数流”或“创建快速函数流”，进入新建函数流页面。

📖 说明

如果通过DWR侧创建的函数流，在console页面是没有编辑和删除权限，如需相关操作请在DWR侧执行。

步骤3 编排函数流任务，请您根据实际应用进行函数流编排。

- 在函数流页面，通过拖拽组件进行流程编排。
以[图13-13](#)为例，将开始节点、函数、结束节点拖入编辑框内，并用连接线连接好。

图 13-13 编排函数流



- 单击编辑框中函数节点进行编辑。配置函数参数，参数说明如[表13-4](#)所示，带*参数为必填项。

📖 说明

配置函数前确保已创建好函数，示例中的函数节点选择[函数2](#)（函数执行返回result的值为函数调用事件内的input输入值+2的结果值），参见[图13-14](#)配置。

图 13-14 函数节点配置

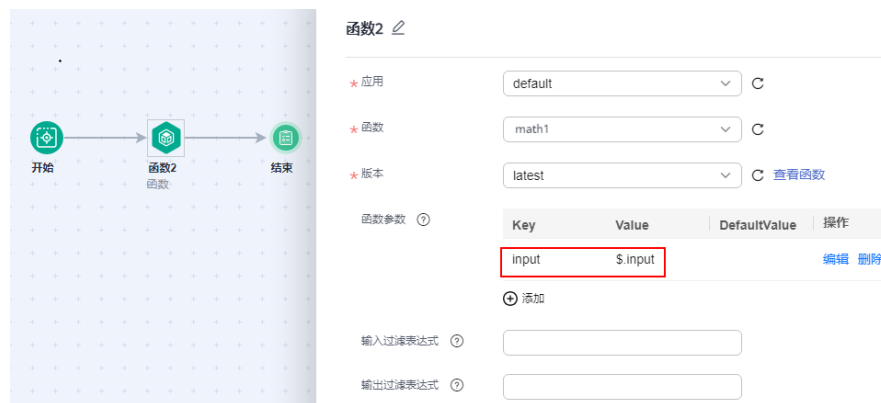


表 13-4 函数参数说明

参数	说明
*应用	函数所属应用，用户创建函数时可以进行分组，每个函数应用下面可以创建多个函数，在函数创建时可以指定其归属于某个函数应用。
*函数	FunctionGraph中对应的函数。 说明 <ul style="list-style-type: none"> 配置的函数节点返回的数据格式必须是json格式，否则会解析失败。 仅对于Go函数支持返回流式数据：在函数详情页的“设置 > 高级设置”下，打开“返回流式数据”开关即可。
*版本	FunctionGraph中函数对应的版本。
*调用方式	函数流执行函数节点时的调用方式，默认为同步调用 说明 <ul style="list-style-type: none"> 同步调用不支持长时间函数，最大执行时长为15分钟 异步调用支持长时间函数，单函数节点的最大执行时长以函数服务支持的最大执行时长为准
函数参数	流程中以json格式作为body参数在执行时传入函数。 Key: 填写参数 Value:填写参数值 DefaultValue: 设置默认值，参数未获取到值时，默认获取默认值 操作: 编辑或删除设置的参数
输入过滤表达式(JSONPath)	基于上一个流程的json输出参数，可以使用JSONPath格式来选择性的过滤出当前流程的输入参数。

参数	说明
输出过滤表达式(JSONPath)	基于当前流程的json输出参数，可以使用JSONPath格式来选择性的过滤出下一流程的输出参数。

3. 单击编辑框中的EG节点进行编辑。配置EG参数，参数说明如表13-5所示，带*参数为必填项。

说明

配置EG节点前确保已创建好EG自定义事件源和自定义事件通道，参见图13-15。

图 13-15 EG 节点配置



表 13-5 EG 节点参数说明

参数	说明
*事件通道	事件通道负责接收来自事件源的事件。函数流仅支持编排自定义事件通道：您自行创建的事件通道，用于接收自定义事件源产生的事件。详情请参考事件网格服务关于事件通道的介绍。
*事件源	事件源是事件的来源，函数流仅支持自定义的应用作为事件源，通过自定义的事件通道发布事件到事件网格。详情请参考事件网格服务关于事件源的介绍
事件内容格式	参数“事件负载内容”的内容格式。 说明 目前只支持application/json格式。
事件负载内容	事件内容。
事件发生主题	事件发生的主题或对象，用以标识哪个具体对象发生了当前事件

参数	说明
输入过滤表达式(JSONPath)	基于上一个流程的json输出参数，可以使用JSONPath格式来选择性的过滤出当前流程的输入参数。
输出过滤表达式(JSONPath)	基于当前流程的json输出参数，可以使用JSONPath格式来选择性的过滤出下一流程的输出参数。

4. 若您的函数流任务中配置了流程控制器，请参见表13-6进行配置，带*参数为必填项。

表 13-6 流程控制器参数说明

类型	参数	说明
回调节点	*回调超时时间（分钟）	回调节点的最长等待时间，超时无回调响应，该节点执行失败。
	*回调类型	函数流可供回调的服务节点类型。 说明 目前只支持函数节点。
	*应用	函数所属应用，用户创建函数时可以进行分组，每个函数应用下面可以创建多个函数，在函数创建时可以指定其归属于某个函数应用。
	*函数	FunctionGraph中对应的函数。 说明 <ul style="list-style-type: none"> 配置的函数节点返回的数据格式必须是json格式，否则会解析失败。 仅对于Go函数支持返回流式数据：在函数详情页的“设置 > 高级设置”下，打开“返回流式数据”开关即可。
	*版本	FunctionGraph中函数对应的版本。
	*调用方式	函数流执行函数节点时的调用方式，默认为同步调用 说明 <ul style="list-style-type: none"> 同步调用不支持长时间函数，最大执行时长为15分钟 异步调用支持长时间函数，单函数节点的最大执行时长以函数服务支持的最大执行时长为准

类型	参数	说明
	函数参数	<p>流程中以json格式作为body参数在执行时传入函数。</p> <p>Key: 填写参数</p> <p>Value:填写参数值</p> <p>DefaultValue: 设置默认值, 参数未获取到值时, 默认获取默认值</p> <p>操作: 编辑或删除设置的参数</p>
	输入过滤表达式	基于上一个流程的json输出参数, 可以使用JSONPath格式来选择性的过滤出当前流程的输入参数。
	输出过滤表达式	基于当前流程的json输出参数, 可以使用JSONPath格式来选择性的过滤出下一流程的输出参数。
子流程	选择子流程	选择已创建的函数流任务。
	是否等待子流程完成	默认选择“是”。
	输入过滤表达式 (JSONPath)	基于上一个流程的json输出参数, 可以使用JSONPath格式来选择性的过滤出当前流程的输入参数。
	输出过滤表达式 (JSONPath)	基于当前流程的json输出参数, 可以使用JSONPath格式来选择性的过滤出下一流程的输出参数。
并行分支	*分支执行完成条件	<ul style="list-style-type: none"> - 所有分支执行完成: 2个或2个以上分支时选择该条件 - 一个分支执行完成: 只有1个分支时选择该条件 - 指定数目分支执行完成: 2个或2个以上分支时其中某个分支可以选择该条件
	输入过滤表达式 (JSONPath)	基于上一个流程的json输出参数, 可以使用JSONPath格式来选择性的过滤出当前流程的输入参数。
	输出过滤表达式 (JSONPath)	基于当前流程的json输出参数, 可以使用JSONPath格式来选择性的过滤出下一流程的输出参数。
	指定分支执行完成数目	当“分支执行完成条件”选择指定数目分支执行完成时, 支持自定义执行完成的数目。
	*结果输出路径	输入并行分支执行结果输出位置, 输入值作为key, 并行分支执行结果作为value, 以json形式输出。若未填写, 默认输出路径为: result。

类型	参数	说明
开始节点	加入触发器	用于标识流程的开始，一个流程只能有一个开始节点。如何创建函数流触发器，请参见 创建函数流触发器 。
异常处理	*是否重试	默认关闭，开启后可以控制函数执行失败后的下一步流程。 <ul style="list-style-type: none"> - 重试条件(JSONPath)：例如： \$.status == 500 - 重试间隔(1-30秒)：默认重试间隔1S - 最大重试次数(1-8)：默认重试次数3次
	*异常分支表达式	JSONPath表达式，表达式结果为true时，异常捕获成功，函数流执行至异常分支。
循环节点	*遍历数组路径 (JSONPath)	需要遍历的数组类型变量地址。
	*迭代变量名称	每次循环迭代，引用数组中元素的参数名称。
	*结果输出路径 (JSONPath)	指定全部迭代分支执行结果数组的输出位置。
	并发迭代数目	并发运行迭代分支的数目，限制0-100, 0代表并发拉起的数目无限制。
	并发迭代时间间隔 (秒)	每次迭代间隔的时间。
	输入过滤表达式 (JSONPath)	基于上一个流程的json输出参数，可以使用JSONPath格式来选择性的过滤出当前流程的输入参数。
	输出过滤表达式 (JSONPath)	基于当前流程的json输出参数，可以使用JSONPath格式来选择性的过滤出下一流程的输出参数。
时间等待	*延迟时间 (秒)	默认60秒。
服务节点	执行模式	定义服务节点中函数的执行顺序。 <ul style="list-style-type: none"> - 串行模式：服务中的函数节点按照连线顺序依次执行，可以严格保证函数的执行顺序 - 并行模式：服务中的函数节点并行执行，不保证内部函数节点的执行顺序
	输入过滤表达式	通过JSONPath表达式对节点的输入信息进行过滤。

类型	参数	说明
	输出过滤表达式	通过JSONPath表达式对节点的输出信息进行过滤。
条件分支	*分支类型	- 条件分支 - 默认分支 当一个分支选择条件分支时，必须要有一个分支选择默认分支。
	表达式	选择“条件分支”，需要输入JSONPath类型表达式。使用方法请参见 表达式运算符说明 。
	输入过滤表达式	通过JSONPath表达式对节点的输入信息进行过滤。
	输出过滤表达式	通过JSONPath表达式对节点的输出信息进行过滤。
结束节点	流程结束的标志	后面不能接任何节点。

5. 流程中的所有节点参数配置完成后，单击右上角的“保存”。

说明

函数流中的节点改动后，必须先保存信息，再启动函数流任务。


6. 在新建函数流页面，单击左上角函数流名称右边的，进行参数配置，最后单击“确定”。

表 13-7 输入配置信息

参数	说明
*名称	输入函数流名称。
*企业项目	选择企业项目。
合并参数	将上一个节点的输出与下一个节点的输入合并为输入。
描述	输入函数流的简要描述。

图 13-16 编辑函数流



7. 单击“启动”，在弹出的启动执行页面，支持输入定义值或者直接启动。此处选择“输入定义值”。

```
{
  "input":3
}
```

图 13-17 启动执行配置

启动执行

定义输入值

直接启动

您可以编写JSON格式的内容，定义业务流执行的输入值

```

1 {
2   "input":3
3 }
4
5
6
7
8
9
10
11
12
```

开始执行

取消

说明

输入定义值必须是JSON格式的内容。

8. 单击“开始执行”，页面右上角提示“启动函数流xxx成功”。
9. 单击函数流任务名称，进入函数流任务详情，查看函数流执行结果。

----结束

查看函数流任务

步骤1 返回函数 workflow 控制台，进入“函数流”页面。

步骤2 在函数流页面，支持设置函数流任务卡片显示和列表显示。

图 13-18 设置显示方法



- 卡片显示

图 13-19 卡片显示



- 列表显示

图 13-20 列表显示

名称	描述	企业项目	模式	更新时间	操作
flow_callback_EaeTHQWvR4m	desc	--	标准	2024/04/15 11:17:50 GMT+08:00	复制 ID 编辑 更多
flow_callback_8K3Bwepp2Tex	desc	--	标准	2024/04/12 21:07:37 GMT+08:00	复制 ID 编辑 更多
flow_project_sash85	--	default	标准	2024/04/12 11:46:46 GMT+08:00	复制 ID 编辑 更多
wankow_RD59KUsuGYH	desc	--	标准	2024/02/11 08:31:25 GMT+08:00	复制 ID 编辑 更多
wankow_gPhvenZ9SRH	test_flow_description	--	标准	2024/02/11 05:06:31 GMT+08:00	复制 ID 编辑 更多
flow_callback_sZVnwbEvg	--	default	标准	2024/02/11 05:38:23 GMT+08:00	复制 ID 编辑 更多
flow_callback_shtZRAJ4N	desc	--	标准	2024/02/10 09:38:19 GMT+08:00	复制 ID 编辑 更多
flow_gpWw3uDtLd	desc	--	标准	2024/02/10 09:04:28 GMT+08:00	复制 ID 编辑 更多
wankow_DMB7amngK5s	desc	--	标准	2024/02/10 08:25:34 GMT+08:00	复制 ID 编辑 更多
flow_callback_u99gKytTjUq	desc	--	标准	2024/02/10 05:11:52 GMT+08:00	复制 ID 编辑 更多

步骤3 在函数流页面，可以查看所有已创建的函数流任务，您可以执行如表13-8中的管理操作。

表 13-8 操作说明

操作项	说明
编辑	在函数流任务卡片中，单击“编辑”，进入函数流编辑页面，更新修改函数流任务信息。

操作项	说明
启动	在函数流任务卡片中，单击“更多 > 启动”，启动该函数流任务。
删除	<ul style="list-style-type: none"> 支持单个删除任务：在函数流任务卡片中，单击“更多 > 删除”，删除该函数流任务。 批量删除任务：勾选多个函数流任务，单击左上角的“删除”，删除所选函数流任务。

步骤4 单击某一个函数流任务名称，查看任务详情。

- 查看任务基本信息
在“基本信息”页签，查看任务名称、ID、更新时间和创建时间等信息。
- 查看任务运行记录
在“运行记录”页签，查看任务的执行历史记录、输入值、输出值、节点日志等信息。
若需要修改任务信息，单击右上角的“函数流设计”，进入函数流编辑页面，更新修改任务。
- 查看函数流监控指标
在“监控”页签，查看任务的调用次数、运行时间、错误次数、运行中信息。监控指标说明如表5 函数流监控指标所示。

表 13-9 函数流监控指标

指标	单位	说明
调用次数	次	函数流总的调用请求数，包含了正确、错误和运行中的调用。异步函数流在请求被系统执行时才开始计数。
运行时间	毫秒	时间段内单次函数流执行平均的运行时间。
错误次数	次	指发生异常请求的函数流不能正确执行完，会计入错误次数。
运行中	个	正在运行中的函数流的数量。
被拒绝次数	次	指被限流无法执行函数流的次数

---结束

13.3 函数流执行历史管理

标准函数流执行历史查询

步骤1 登录FunctionGraph控制台，在左侧导航栏选择“函数流”，进入“函数流”页面。

步骤2 在“函数流”流程列表页面，单击需要查看执行历史的流程，进入流程详情页面。

步骤3 切换至“运行记录”页签，查看执行历史运行记录。

步骤4 左侧为执行历史记录列表，展示最近100次执行记录，支持根据流程执行ID进行查询。



- 单击左侧的执行记录，中间画布展示流程的执行结果，如果节点执行成功，图标背景为绿色，如果执行失败背景为红色。

图 13-21 执行失败



- 画布下方输入输出展示区默认展示流程的输入和输出，单击上方任意节点，展示节点的输入和输出。



说明

对于函数流及函数流各个节点，若输出值中有字段的值为null，则该字段会被直接过滤，不予展示。

- 最下方日志展示流程从开始到结束所有节点的执行记录。

图 13-22 节点日志

节点名称	状态	请求ID	耗时	开始时间
函数	执行失败	XXXXXXXXXXXXXXXXXXXX	451 ms	2024/04/15 18:55:18 GMT+08:00

总数: 1

----结束

快速函数流执行日志查询

- 返回函数 workflow 控制台，在左侧导航栏选择“函数流”，进入“函数流”页面。
- 在“函数流”列表页面，单击需要查看执行历史的流程，进入函数流“基本信息”页面。
- 切换至“运行记录”页签，查看执行历史日志。
- 可根据请求ID过滤日志，或自定义时间过滤日志，单击请求ID查看执行日志详情。

图 13-23 查看执行历史



----结束

失败流程重试

- 步骤1** 返回函数 workflow 控制台，在左侧导航栏选择“函数流”，进入“函数流”页面。
- 步骤2** 在“函数流”流程列表页面，单击需要查看执行历史的流程，进入流程详情页面。
- 步骤3** 切换至“运行记录”页签，查看执行历史运行记录。
- 步骤4** 在失败的记录右侧单击重试图标，重试成功后会生成一条执行记录。



----结束

运行中流程终止

- 步骤1** 返回函数 workflow 控制台，在左侧导航栏选择“函数流”，进入“函数流”页面。
- 步骤2** 在“函数流”流程列表页面，单击需要查看执行历史的流程，进入流程详情页面。
- 步骤3** 切换至“运行记录”页签，单击执行中任务的停止图标，终止成功后流程会进入取消状态。

----结束

13.4 创建函数流触发器

本节介绍创建函数流触发器，函数流触发器当前支持APIG触发器、定时触发器、SMN触发器。

创建定时触发器

- 步骤1** 登录FunctionGraph控制台，进入“函数流”页面。
- 步骤2** 在“函数流”流程列表页面，选择需要创建触发器的流程，单击“编辑”，进入编辑页面。

步骤3 单击“开始”节点，在右侧弹出的属性页面添加触发器，触发器类型选择“定时触发器”。

图 13-24 定时触发器

创建触发器

触发器类型 ? 定时触发器 (TIMER) ▼

可以使用TIMER的计划事件功能定期调用您的代码，可以指定固定频率（分钟、小时、天数）或指定Cron表达式定期调用函数。
DDS、GAUSSMONGO、DIS、LTS、KAFKA、TIMER、ROCKETMQ触发器可创建数加起来最多10个，您已创建0个。

* 定时器名称

支持字母、数字、下划线和中划线，必须以字母开头，且长度不能超过64个字符

* 触发规则 固定频率 Cron表达式

单位为分钟时，输入值不能超过60；单位为小时时，输入值不能超过24；单位为天时，输入值不能超过30。

* 是否开启

附加信息 ?

0/2,048 ↕

步骤4 填写触发器配置信息。如表13-10所示，带*参数为必填项。

表 13-10 定时触发器配置信息

配置项	说明
*触发规则	定时触发器的触发规则，当前只支持Cron表达式。
*Cron表达式	用于表示任务调度的表达式，能够表示特定周期进行的特定的时间、日期等。具体请参见 函数定时触发器Cron表达式规则 。
附加信息	附加信息为json格式，输入必须包含input，在input内输入需要的json体。input的内容会作为流程的输入参数。

步骤5 单击“创建”，完成定时触发器创建。

----结束

创建 APIG（共享版）触发器

说明

首次使用API网关的用户不再支持共享版服务，老用户仍可继续使用共享版服务。即API网关当前已不提供共享版，目前只有存量用户可以使用共享版。

函数流APIG触发器目前仅支持IAM认证方式。

- 步骤1** 返回函数 workflow 控制台，进入“函数流”页面。
- 步骤2** 在“函数流”流程列表页面，选择需要创建触发器的流程，单击“编辑”，进入编辑页面。
- 步骤3** 单击“开始”节点，在右侧弹出的属性页面添加触发器，触发器类型选择“APIG触发器（共享版）”。

图 13-25 APIG（共享版）触发器

创建触发器

触发器类型 ?

可以通过HTTPS或者HTTP调用FunctionGraph函数，将各个API操作（如GET和PUT）映射到特定的FunctionGraph函数，当向该API发送HTTPS或者HTTP请求时，API 网关会调用相应的FunctionGraph函数。

当使用APIG触发器时，函数返回体必须为如下json格式：

```
{
  "statusCode": 200,
  "isBase64Encoded": false,
  "headers": {
    "Content-Type": "application/json;charset=UTF-8"
  },
  "body": "hello world"
}
```

 (isBase64Encoded为true时，body必须为base64编码格式)

* API名称
支持汉字，英文，数字，下划线，且只能以英文和汉字开头，3-64字符

* 分组 [创建分组](#)

* 发布环境 [创建发布环境](#)

* 安全认证
IAM：IAM认证，只允许IAM用户能访问，安全级别中等；
App：采用Appkey&Appsecret认证，安全级别高，推荐使用；
None：无认证模式，所有用户均可访问，不推荐使用。

* 请求协议

* 后端超时(毫秒)
后端超时设置1-60,000毫秒的范围内

- 步骤4** 填写触发器配置信息。如表13-11所示，带*参数为必填项。

表 13-11 APIG 触发器（共享版）信息

字段	填写说明
*分组	API分组相当于一个API集合，API提供方以API分组为单位，管理分组内的所有API。 选择“APIGroup_test”。

字段	填写说明
*发布环境	API可以同时提供给不同的场景调用，如生产、测试或开发。 API网关服务提供环境管理，在不同的环境定义不同的API调用路径。 选择“RELEASE”，才能调用。
*API类型	API类型：公开、私有
*路径	接口请求的路径。 格式如：/users/projects
*请求方式	接口调用方式：GET、POST、DELETE、PUT、PATCH、HEAD、OPTIONS、ANY 其中ANY表示该API支持任意请求方法。

步骤5 单击“创建”，完成APIG（共享版）触发器创建。

----结束

创建 APIG（专享版）触发器

📖 说明

- 函数流APIG触发器目前仅支持IAM认证方式。
- 前提条件：需要预先创建APIG专享版实例，具体请参见[购买专享版实例](#)。

步骤1 返回函数 workflow 控制台，进入“函数流”页面。

步骤2 在“函数流”流程列表页面，选择需要创建触发器的流程，单击“编辑”，进入编辑页面。

步骤3 单击“开始”节点，在右侧弹出的属性页面添加触发器，触发器类型选择“APIG触发器(专享版)”。

图 13-26 APIG（专享版）触发器

创建触发器

触发器类型 ②

可以通过HTTPS或者HTTP调用FunctionGraph函数，将各个API操作（如GET和PUT）映射到特定的FunctionGraph函数。当向该API发送HTTPS或者HTTP请求时，API网关会调用相应的FunctionGraph函数。

当使用APIG触发器时，函数返回体必须为如下json格式：

```
{
  "statusCode": 200,
  "isBase64Encoded": false,
  "headers": {
    "Content-Type": "application/json;charset=UTF-8",
    "body": "hello world"
  }
}
```

 (isBase64Encoded为true时，body必须为base64编码格式)

* API名称

支持汉字，英文，数字，下划线，且只能以英文和汉字开头，3-64字符

* 实例 [创建实例](#)

* 分组 [创建分组](#)

* 发布环境 [创建发布环境](#)

* 安全认证

IAM: IAM认证，只允许IAM用户能访问，安全级别中等；
 App: 采用Appkey&Appsecret认证，安全级别高，推荐使用；
 None: 无认证模式，所有用户均可访问，不推荐使用。

* 请求协议

* 后端超时(毫秒)

后端超时设置1-60,000毫秒的范围内

步骤4 填写触发器配置信息。如表13-12，带*参数为必填项。

表 13-12 APIG 触发器（专享版）信息

字段	填写说明
*实例	专享版APIG实例名称。
*分组	API分组相当于一个API集合，API提供方以API分组为单位，管理分组内的所有API。 选择“APIGroup_test”。
*发布环境	API可以同时提供给不同的场景调用，如生产、测试或开发。API网关服务提供环境管理，在不同的环境定义不同的API调用路径。 选择“RELEASE”，才能调用。
*API类型	API类型：公开、私有
*路径	接口请求的路径。 格式如：/users/projects

字段	填写说明
*请求方式	接口调用方式：GET、POST、DELETE、PUT、PATCH、HEAD、OPTIONS、ANY 其中ANY表示该API支持任意请求方法。

步骤5 单击“创建”，完成APIG（专享版）触发器创建。

----结束

13.5 流式文件处理

本章节主要介绍如何使用函数流实现流式大文件处理。您可以根据实际业务场景来创建快速函数流实现。

背景与价值

Serverless Workflow由于自身可编排、有状态、持久化、可视化监控、异常处理、云服务集成等特性，适用于很多应用场景，比如：

- 复杂度高需要抽象的业务（订单管理，CRM 等）
- 业务需要自动中断 / 恢复能力，如多个任务之间需要人工干预的场景（人工审批，部署流水线等）
- 业务需要手动中断 / 恢复（数据备份 / 恢复等）
- 需要详细监控任务执行状态的场景
- 流式处理（日志分析，图片 / 视频处理等）

当前大部分 Serverless Workflow 平台更多关注控制流程的编排，忽视了工作流中数据流的编排和高效传输，上述场景[创建函数流触发器](#)中，由于数据流相对简单，所以各大平台支持都比较好，但是对于文件转码等存在超大数据流的场景，当前各大平台没有给出很好的解决方案。华为云FunctionGraph函数工作流针对该场景，提出了 Serverless Streaming 的流式处理方案，支持毫秒级响应文件处理。

技术原理

华为云FunctionGraph函数工作流提出 Serverless Streaming 的流式可编排的文件处理解决方案，步骤与步骤之间通过数据流驱动，更易于用户理解。本章通过图片处理的例子解释该方案的实现机制。

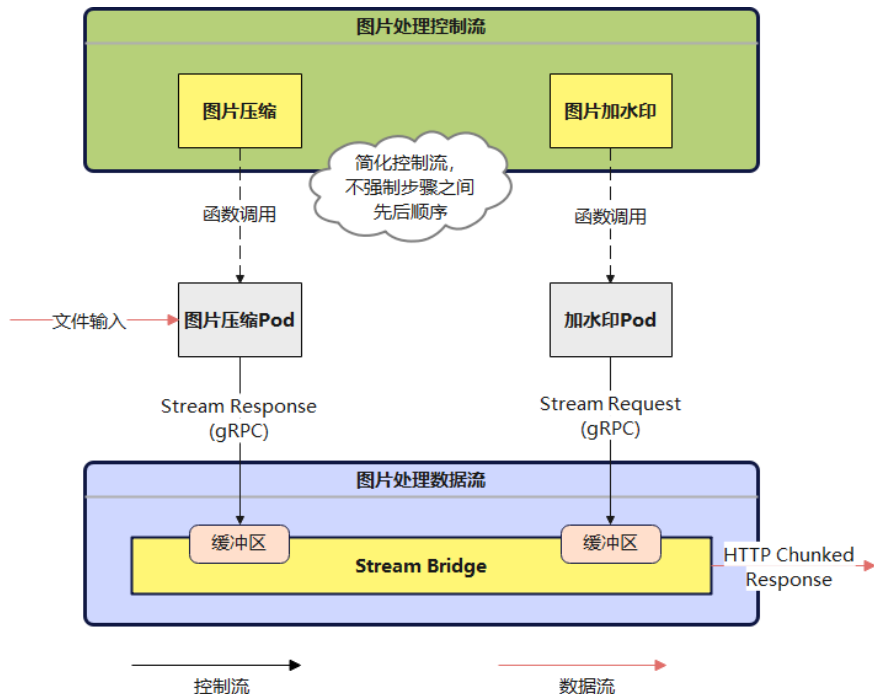
如果需要驱动一个工作流执行，工作流系统需要处理两个部分：

- 控制流：控制工作流的步骤间流转，以及步骤对应的 Serverless 函数的执行。确保步骤与步骤之间有序执行。
- 数据流：控制整个工作流的数据流转，通常来说上一个步骤的输出是下一个步骤的输入，比如上述图片处理工作流中，图片压缩的结果是打水印步骤的输入数据。

在普通的服务编排中，由于需要精准控制各个服务的执行顺序，所以控制流是工作流的核心部分。然而在文件处理等流式处理场景中，对控制流的要求并不高，以上述图片处理场景举例，可以对大图片进行分块处理，图片压缩和加水印的任务不需要严格

的先后顺序，图片压缩处理完一个分块可以直接流转到下一个步骤，而不需要等待图片压缩把所有分块处理完再开始加水印的任务。

基于上述理解，华为云FunctionGraph工作流的 Serverless Streaming 方案架构设计如下图所示：



在 Serverless Streaming 的流程中，弱化控制流中步骤之间的先后执行顺序，允许异步同时执行，步骤与步骤之间的交互通过数据流驱动。其中数据流的控制通过 Stream Bridge 组件来实现。同时函数 SDK 增加流式数据返回接口，用户不需要将整个文件内容返回，而是通过 gRPC Stream 的方式将数据写入到 Stream Bridge，Stream Bridge 用来分发数据流到下一个步骤的函数 Pod 中。

操作步骤

步骤1 创建一个图片压缩的函数，其中代码在处理返回数据通过 `ctx.Write()` 函数将结果以流式数据的形式返回：

📖 说明

目前只支持go函数！

```

func ImageTransform(payload []byte, ctx context.RuntimeContext) (interface{}, error) {
    reader, writer := io.Pipe()
    file, err := downloadOriginFile(ctx)
    if err != nil {
        return "nok", err
    }

    go func() {
        defer writer.Close()
        encodeImageFile(writer, file, ctx)
    }()

    defer reader.Close()
    // Write result stream back to client
    err = ctx.Write(reader)
    return "ok", err
}

// downloadOriginFile used to download the origin video file
func downloadOriginFile(ctx context.RuntimeContext) (*os.File, error) {
}

// encodeImageFile used to encode the origin image file to target resolution, result output will writes to writer stream
func encodeImageFile(writer io.Writer, originFile *os.File, ctx context.RuntimeContext) {
}

```

FunctionGraph 通过 `ctx.Write()` 函数提供了流式返回的能力，对开发者来说，只需要将最终结果通过流的方式返回，而不需要关注网络传输的细节。

步骤2 在 FunctionGraph 的函数流控制台完成 workflow 编排，举例如下。



步骤3 调用工作流的同步执行接口，获取最终结果的文件流，数据将以 chunked 流式返回的方式返回到客户端。

----结束

14 扩大资源配额

概述

为防止资源滥用，平台限制了各服务资源的配额，对用户的资源数量和容量做了限制。如您最多可以创建多少台弹性云服务器、多少块云硬盘。

如果当前资源配额限制无法满足使用需要，您可以申请扩大配额。

查看配额

1. 登录管理控制台。
2. 在页面右上角，选择“资源 > 我的配额”。系统进入“服务配额”页面。

图 14-1 进入服务配额页面



3. 您可以在“服务配额”页面，查看各项资源的总配额及使用情况。如果当前配额不能满足业务要求，请参见后续操作，申请扩大配额。

申请扩大配额

1. 登录管理控制台。
2. 在页面右上角，选择“资源 > 我的配额”。系统进入“服务配额”页面。

图 14-2 进入“服务配额”页面



3. 单击“申请扩大配额”。
4. 在“新建工单”页面，根据您的需求，填写相关参数。
其中，“问题描述”项请填写需要调整的内容和申请原因。
5. 填写完毕后，勾选协议并单击“提交”。

15 GPU 函数管理

15.1 Serverless GPU 使用介绍

15.1.1 概述

Serverless GPU是一种高度灵活、高效利用、按需分配GPU计算资源的新兴云计算服务。GPU能力Serverless化，通过提供一种按需分配的GPU计算资源，在一定范围内有效地解决原有GPU长驻使用方式导致的低资源利用率、高使用成本和低弹性能力等痛点问题。本文将介绍Serverless GPU的详细功能和优势。

传统GPU长驻使用方式存在许多问题，例如，需要提前规划好资源需求并容易造成资源浪费。而Serverless GPU则提供了一种更加灵活的方式来利用GPU计算资源，用户只需选择合适的GPU型号和计算资源规模，就可以帮助用户有效地解决GPU长驻使用方式导致的资源浪费、高成本、低弹性等问题，为用户提供更加便捷、高效的GPU计算服务，有效承载AI模型推理、AI模型训练、音视频加速生产、图形图像加速等加速工作负载。

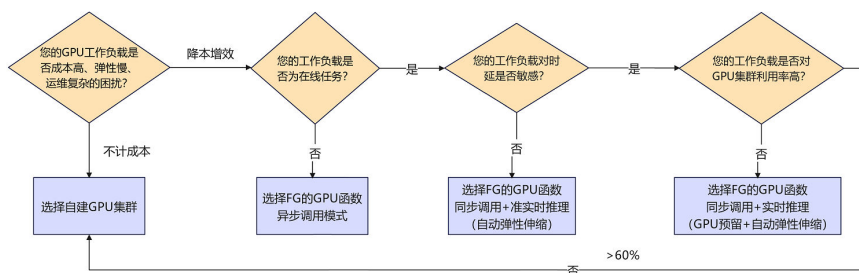
GPU函数主要用于：仿真、模拟、科学计算、音视频、AI和图像处理等场景下，使用GPU硬件加速，从而提高业务处理效率。

表 15-1 GPU 函数规格

卡型	vGPU 显存 (GB)	vGPU 算力 (卡)	特点描述
----	--------------	-------------	------

NVIDIA-T4	1~16 取值说明：必须是整数。	说明：由系统自动分配，无需手动配置	T4是一款独特的GPU产品，专为AI推理工作负载而设计，如处理视频，语音，搜索引擎和图像的神经网络。T4配备16GB GDDR6，GPU中集成320个Turing Tensor Core和2560个Turing CUDA Core，这款全新GPU具有突破性的性能，以及FP32/FP16/INT8/INT4等多种精度的运算能力，FP16的峰值性能为65T，INT8为130T，INT4为260T。
-----------	---------------------	-------------------	--

图 15-1 GPU 云产品选型决策指引



说明

- 目前该功能仅支持华东-上海一。
- GPU函数不支持的网段：192.168.64.0/18，192.168.128.0/18，10.192.64.0/18，10.192.128.0/18。

15.1.2 应用场景

15.1.2.1 准实时推理场景

本章节介绍什么是准实时推理场景，以及如何使用GPU按量实例和如何基于GPU按量实例构建使用成本较低的准实时推理服务。

特征

在准实时推理应用场景中，工作负载具有以下一个或多个特征：

- **调用稀疏**
日均调用几次到几次，日均GPU实际使用时长远低于6~10小时，GPU存在大量闲置。

- **单次处理耗时长**
准实时推理业务的处理耗时一般在秒级~分钟级。例如，典型的CV任务处于秒级别，典型的视频处理和AIGC场景均处于分钟级别。
- **容忍冷启动**
业务可以容忍GPU冷启动耗时，或者业务流量波形对应的冷启动概率低。

功能优势

函数计算为准实时推理工作负载提供以下功能优势：

- **原生Serverless使用方式**
函数计算平台默认提供的按量GPU实例使用方式，会自动管理GPU计算资源。根据业务的请求数量，自动弹性GPU实例，最低0个实例，最大可配置实例数量。
- **规格最优**
函数计算平台提供的GPU实例规格，根据业务需求，选择卡型并配置使用的显存和内存的大小，为您提供最贴合业务的实例规格。
- **成本最优**
函数计算平台提供的按量付费能力，对于低GPU资源利用率的工作负载，降本幅度可达70%以上。

15.1.2.2 实时推理场景

特征

在实时推理应用场景中，工作负载具有以下一个或多个特征：

- **低延迟**
单次请求的处理时效性要求高，RT（Response Time）延迟要求严格，90%的长尾延时普遍在百毫秒级别。

功能优势

函数计算为实时推理工作负载提供以下功能优势：

- **预留GPU实例**
函数计算平台提供了默认的按量GPU实例之外的另一种GPU使用方式——预留GPU实例。如果您希望消除冷启动延时的影响，满足实时推理业务低延迟响应的要求，可以通过配置预留GPU实例来实现。更多关于预留模式的信息，请参见[预留实例管理](#)。
- **服务质量优先，服务成本次优**
预留GPU实例的计费周期不同于按量GPU实例，预留GPU实例是以实例存活生命周期进行计费，而不考虑实例的活跃与闲置（不按请求计费）。因此，相较于按量GPU实例，总体使用成本较高，但相较于长期自建GPU集群，降本幅度达50%以上。
- **规格最优**
函数计算平台提供的GPU实例规格，允许您根据自己的工作负载选择不同的卡型，独立配置GPU/MEM。最小GPU规格小至1 GB显存/算力，将为您提供最贴合业务的实例规格。

- **突发流量支撑**

函数计算平台提供充足的GPU资源供给，当业务遭遇突发流量时，函数计算将以秒级弹性供给海量GPU算力资源，避免因GPU算力供给不足、GPU算力弹性滞后导致的业务受损。

15.1.2.3 离线异步任务场景

特征

在离线异步应用场景中，工作负载具有以下一个或多个特征：

- **执行时间长**

业务的处理耗时一般在分钟~小时级，Response Time不敏感。

- **提交后立即返回**

在触发调用后立即得到返回，从而不因长耗时处理阻塞业务主逻辑的执行。

- **实时感知任务状态**

无

- **并行处理**

离线GPU任务需要处理大量数据，对GPU资源供给要求高，通过API调用并行运行加快处理速度。

- **数据源集成**

离线GPU任务对数据源的需求多种多样，处理过程中需要与多种存储产品（例如对象存储OBS）和多种消息产品（例如消息队列）进行频繁交互。

功能优势

函数计算为离线异步应用类工作负载提供以下功能优势：

- **业务架构简化**

对于长耗时，采用异步处理，提高系统响应速度、资源利用率和可用性。

- **充足的GPU资源供给**

函数计算平台提供充足的GPU资源供给，适合忙闲流量分明（长时空闲、短时繁忙）、忙闲流量不可预知的离线业务。

- **数据源集成**

函数计算支持多种数据源触发方式，例如对象存储OBS、消息队列等。

15.2 部署方式

GPU 函数目前支持自定义镜像方式部署和自定义运行时方式部署。

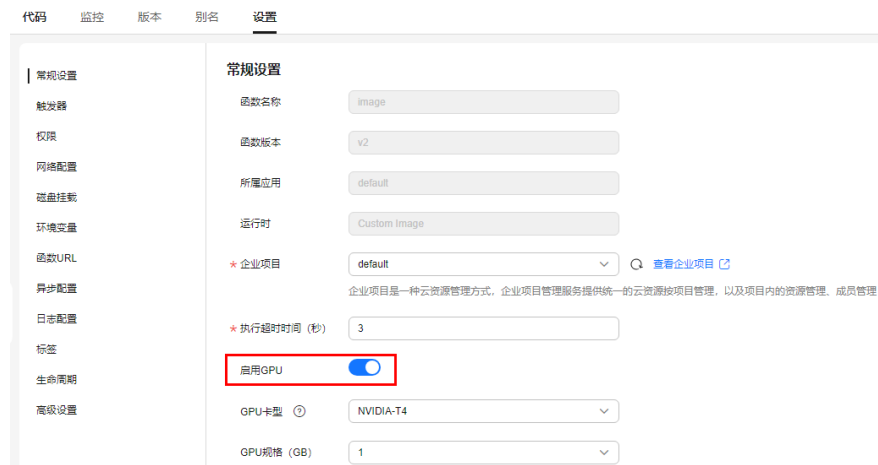
15.2.1 自定义镜像方式部署

GPU 型号仅支持 NVIDIA Tesla 系列。例如：Tesla 系列 T4 卡型。

自定义镜像函数部署详见[使用容器镜像部署函数](#)。

自定义镜像函数，可以在设置->常规设置中，启用GPU。

图 15-2 启用 GPU



15.2.2 定制运行时方式部署

支持使用定制运行时方式部署 GPU 函数。

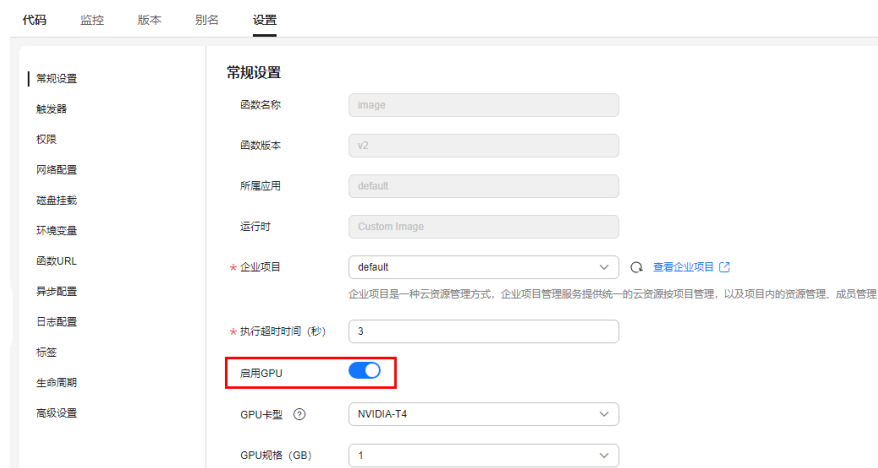
定制运行时详见[定制运行时语言](#)

定制运行时部署方式内置 python(Python 2.7.15)、python3(Python 3.6.8)、python3.7(Python 3.7.4)、python3.9(Python 3.9.2)。

约束：内置 cuda 11.6，函数需要基于 cuda11.6 版本开发，使用其他版本的 cuda 请考虑使用自定义镜像函数。

定制运行时函数，可以在设置->常规设置中，启用 GPU。

图 15-3 启用 GPU



15.3 函数模式

GPU函数均支持两种函数模式：按量模式和预留模式。详见[预留实例管理](#)。

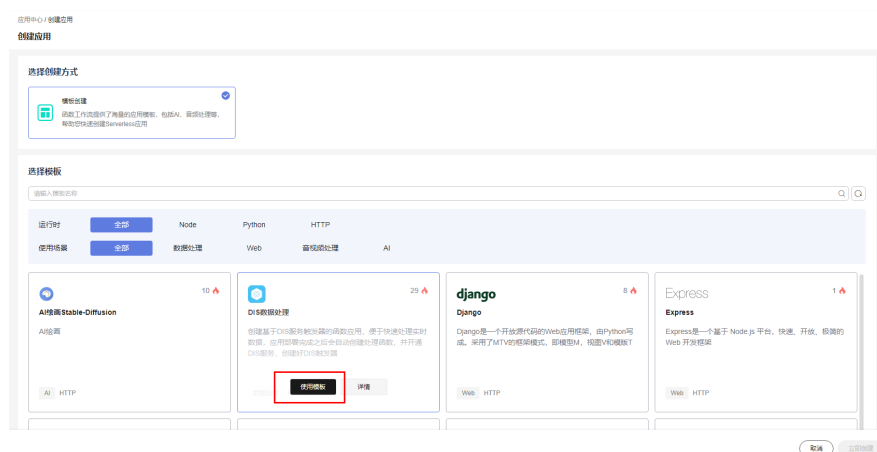
16 应用中心

应用中心使用资源编排服务来部署应用需要的周边资源（包含函数、委托、触发器等），使这些资源相互配合，共同执行任务。（目前仅华北-北京四、华东-上海一暂时支持该功能。）

创建步骤

- 步骤1** 登录[函数工作流控制台](#)，在左侧导航栏选择“应用中心”。
- 步骤2** 单击右上方的“创建应用”，进入“选择应用模板”页面。
- 步骤3** 左侧导航栏分为“运行时”和“使用场景”两个筛选条件，您可根据业务需求筛选相应模板，本章节以创建DIS数据处理模板为例进行介绍。确定好模板后，单击“使用模板”进入应用配置页面。

图 16-1 选择应用模板



步骤4 在“应用配置”页，填写如下信息：

- 区域：默认。
- 项目：默认。
- 应用名称：自定义。
- 运行时语言：默认。

- 委托名称：根据实际情况选择是否使用委托，例如**步骤3**中创建的DIS数据处理模板，需要您创建委托授权函数访问DIS服务，具体如何创建委托请参考**配置委托权限**。
- 描述：自定义。

信息配置完成后，单击“立即创建”。

---结束

参数介绍

应用程序创建成功后，可在应用中心列表中单击应用程序名称查看详细信息。以下是主要参数信息的介绍：

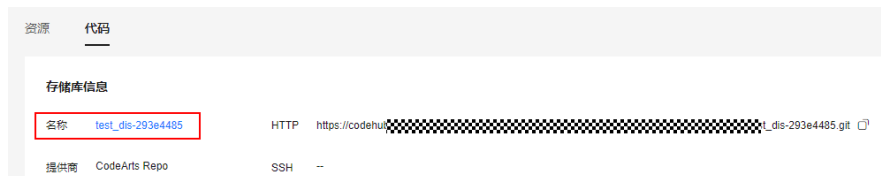
- 资源栈名称：通过该链接可以跳转到资源栈部署成功后的任务详情页面。
- 应用ID：当前部署的应用程序在系统中唯一标识，可以通过该标识和应用名称定位问题。
- DIS数据服务：通过该链接可以跳转到已创建好的DIS数据服务详情页面。
- 函数服务：通过该连接可以跳转到已创建好的函数详情页面。

图 16-2 总览页参数



- 存储库信息中“名称”：通过该链接可跳转到相关函数代码托管的代码仓，可用于浏览及下载相关代码。

图 16-3 代码页参数



常见问题排查方法

1. 创建代码仓库失败，提示如图16-4所示。

图 16-4 创建代码仓失败



排查方法：请检查您账户是否开通了CodeArts服务，具体请参考[登录软件开发生产线 \(CodeArts\)](#) 检查并开通服务。如果开通后问题仍存在，请联系华为云函数 workflow 服务工程师进一步帮助。

- 2. 堆栈部署失败，提示如图16-5所示。

图 16-5 堆栈部署失败



排查方法：单击应用程序名称链接进入应用“总览”页，单击“资源栈名称”链接，跳转到资源编排服务页面，单击“查看失败原因”链接，查看具体问题。

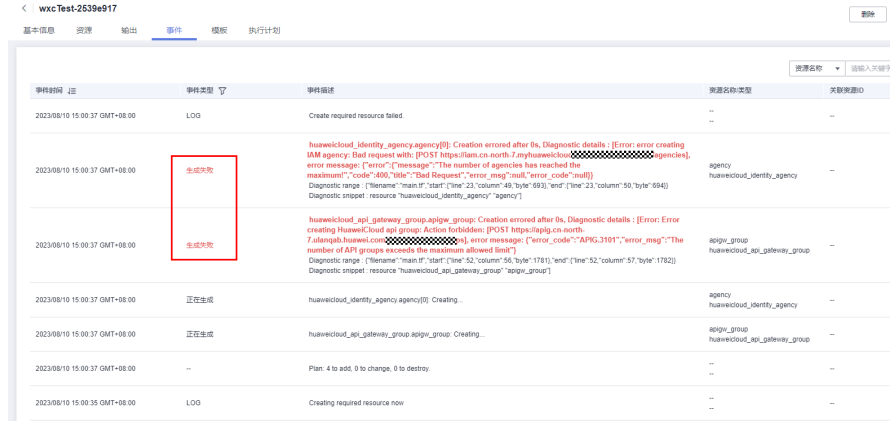
图 16-6 总览页



图 16-7 资源编排服务页面



图 16-8 查看失败原因



3. 权限不足，提示如图16-9所示。

图 16-9 权限不足



排查方法：首次创建应用时报错“权限不足”，请配置当前账号的委托权限，然后重试。

4. 删除失败，提示如图16-10所示。

图 16-10 删除失败



排查方法：单击应用程序名称链接进入应用“总览”页，单击“资源栈名称”链接，跳转到资源编排服务页面，单击“查看失败原因”链接，查看具体问题。以上图提示为例，删除失败原因是API组中包含API导致删除失败，通过进入应用程序“总览”页，单击函数服务的“物理资源名称/ID”链接，进入函数详情页，选择“设置”->“触发器”，查看API名称。然后进入API网关服务控制台，将该API下线并删除，最后再删除重试。

图 16-11 单击函数服务链接

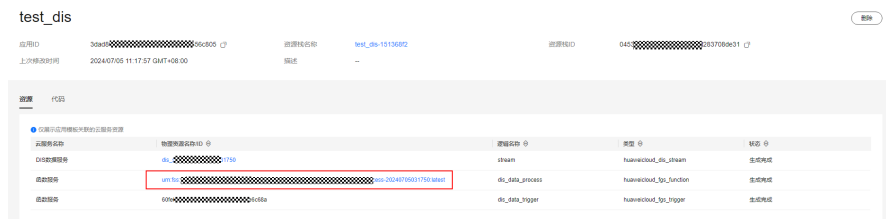
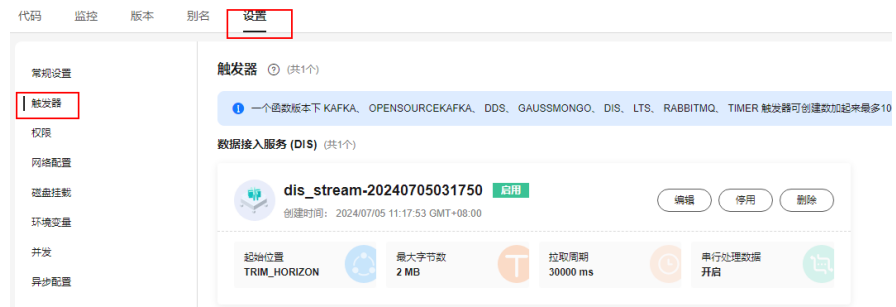


图 16-12 查看 API 名称



17 共享

共享函数资源简介

基于资源访问管理（Resource Access Manager，简称RAM）服务，函数 workflow 服务可以实现跨账号共享函数资源，资源所有者将资源同时共享给多个其他账号使用，资源使用者接受共享邀请后就可以访问和使用共享的函数资源，就像拥有它们一样。资源所有者可以依据最小权限原则和不同的使用诉求，选择不同的共享权限，资源使用者只能对资源进行权限内的访问，保证共享资源在满足资源使用者业务诉求的同时，提升资源管理的安全性。关于RAM服务的更多信息请参见[什么是资源访问管理](#)。

当您的账号由华为云组织管理时，您还可以利用此优势更轻松地共享资源。如果您的账号在组织中，则您可以与单个账号共享，也可以与组织或OU中的所有账号共享，而不必枚举每个账号，具体请参见[启用与组织共享资源](#)。

约束与限制

共享函数资源的前提条件和约束与限制如下所示：

- 您的账号中必须拥有该函数资源，即您必须为该资源的所有者。您无法共享已与您共享的函数资源。
- 当您需要与您的组织或组织单元共享函数资源时，则您必须启用与组织共享资源功能。更多信息请参考[启用与组织共享资源](#)。
- 单个资源使用者最多只能被共享50个函数资源。

创建函数资源共享

资源所有者需要在RAM管理控制台中创建共享资源，详情请参见[创建共享](#)。

📖 说明

- 指定共享资源时，资源类型选择“functiongraph:function”。
- 共享函数资源创建完成后，需要使用者在一定时间内接受共享申请，才可以使用该函数资源，详细请参见[接受/拒绝共享邀请](#)。

查看共享详情

登录函数 workflow 控制台，选择“函数 > 函数列表 > 共享函数”，可以查看其他账号共享的函数。

图 17-1 查看共享资源



说明

- 如果您是函数资源的所有者，您可以通过共享名称，在RAM管理控制台，找到对应的共享，查看共享内的资源情况、资源的权限以及资源的使用者，具体操作请参见[查看共享](#)。
- 如果您是函数资源的使用者，您可以通过共享名称，在RAM管理控制台，找到对应的共享，查看共享内的资源情况、资源的权限以及资源的所有者，具体操作请参见[查看共享给您的资源](#)。

使用共享

步骤1 返回函数 workflow 控制台，选择“函数 > 函数列表 > 共享函数”。

步骤2 在其他账号共享的函数列表中，单击函数名称，可以查看和执行函数。

----结束

说明

您在使用共享前，需要在RAM管理控制台，接受邀请，详情参见[接受/拒绝共享邀请](#)。

停止共享

- 资源所有者如果不再需要某个共享时，可以随时将其删除，删除共享不会删除共享的资源。共享删除后，共享资源指定的使用者将无法继续使用该共享中的资源，详情请参见[删除共享](#)。
- 资源所有者可以随时更新资源共享实例，支持更新资源共享实例的名称、描述、标签、共享的资源、共享权限以及共享使用者，详情请参见[更新共享](#)。
- 资源使用者如果不再需要访问共享给您的资源，可以随时退出共享。退出共享后，将失去对共享资源的访问权限。

只有当共享资源的指定使用者是华为云账号而不是组织内共享时，才可以退出此共享。如果共享资源的指定使用者是组织，而资源使用者的账号由组织管理，则无法退出此共享，详情请参见[退出共享](#)。

共享函数资源的操作权限说明

所有者和使用者对共享函数资源的使用操作权限不同，具体如[表17-1](#)所示。

表 17-1 共享函数资源的使用操作权限

资源	资源所有者的操作权限	资源使用者的操作权限
函数	所有者拥有函数的全部操作权限。	使用者可以查看和执行共享的函数。

支持共享的资源类型和区域

当前函数 workflow 服务支持共享的资源类型和区域如[表17-2](#)所示：

表 17-2 函数 workflow 服务支持共享的资源类型和区域

云服务	资源类型	支持共享的区域
FunctionGraph	function: 函数	亚太-曼谷 亚太-雅加达 华东-上海一 华北-北京四 中国-香港 亚太-新加坡 土耳其-伊斯坦布尔 中东-利雅得

计费说明

不涉及。

18 可编程 CDN 函数

18.1 创建 CDN 函数

概述

函数 workflow 服务为您提供可编程 CDN 功能，您可以通过函数 workflow 完成云上调试，并下发至 CDN 云下环境。（当前仅支持亚太-新加坡）

说明

用户默认没有创建 CDN 函数的权限，如需使用请联系华为云函数 workflow 服务工程师申请白名单。

前提条件

已创建具备 CDN 调用函数的委托权限。

委托授权

函数使用 CDN 功能前，您需要创建一个委托，这样 CDN 才能调用函数 workflow 服务中创建的函数。

- 步骤1** 登录 [CDN 控制台](#)，在左侧菜单栏中选择“域名管理”。
- 步骤2** 在域名列表中，单击需要修改的域名名称或域名所在行的“设置”，进入域名配置页面。
- 步骤3** 选择“函数管理”页签，如果您首次使用函数管理功能，单击“委托授权”，进入委托授权界面。
- 步骤4** 委托授权配置完成后，单击“确定”，系统将为您在 [IAM 管理控制台](#) 委托页面创建名为“CDNAccessFunctionGraph”委托关系，CDN 将有权限调用您函数 workflow 中的函数。

说明

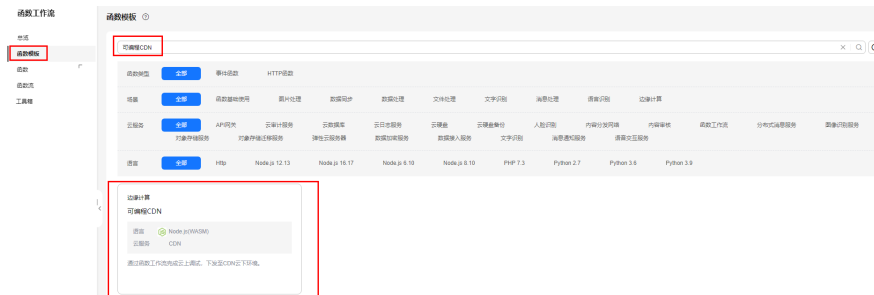
请勿删除 CDN 与 FunctionGraph 的委托关系，否则会导致 CDN 无法调用 FunctionGraph 中的函数。

----结束

操作步骤

- 步骤1** 登录[函数 workflow 控制台](#)，在左侧菜单栏中选择“函数模板”。
- 步骤2** 右上角搜索框输入“可编程CDN”，查询到“可编程CDN”函数模板，单击“使用模板”。

图 18-1 可编程 CDN 模板



- 步骤3** 配置函数基本信息。
 - 区域：自定义。
 - 项目：默认。
 - 函数名称：自定义。
 - 企业项目：默认。

信息确认无误后，单击“创建函数”。

- 步骤4** 在函数详情页中，选择“代码”，进行函数编写。
- 步骤5** 选择“版本”，单击“发布新版本”，配置“版本号”和“描述”，完成后单击“确定”。

图 18-2 发布函数新版本

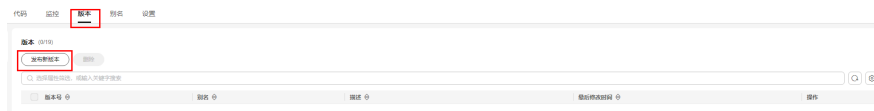


图 18-3 配置版本号

发布新版本

i 新版本将以LATEST版本内容发布，版本发布后代码和配置将无法编辑。单个函数下您最多可以发布20个版本（含latest版本）。 [了解更多](#) ×

版本号

非必填项，若不指定则默认按时间生成版本号，如v20180116-202722
可包含字母、数字、中划线、下划线和点，长度不超过42个字符，以字母/数字开头和结尾

描述

0/512

----结束

18.2 CDN 函数管理

注意事项

函数作用在不同的阶段，CDN控制台其他配置（以下简称配置）与函数管理功能同时配置时优先级也不同，具体如下：

- 客户端请求：配置优先级高，先执行函数再执行配置，配置覆盖函数，最终生效的是CDN控制台的配置。

- 节点回源：函数优先级高，先执行配置再执行函数，函数覆盖配置，最终生效的是函数管理功能。

操作步骤

步骤1 登录**CDN控制台**，在左侧菜单栏中选择“域名管理”，在域名列表中单击需要配置函数的域名，进入详情页。

图 18-4 域名管理



步骤2 选择“函数管理”，如果没有创建委托权限，请参见[委托授权](#)；如已创建，请先单击“编辑”，再单击“添加阶段”。

图 18-5 添加阶段



步骤3 配置函数作用阶段及规则。

- 函数作用阶段选择“客户端请求”，详情请参见[表18-1](#)。
- 匹配类型选择“所有文件”、函数选择刚才创建的函数“programmable-cdn”、函数版本选择“V1”、优先级填写“1”，详情请参见[表18-2](#)，最多可添加40条规则。

图 18-6 配置函数作用阶段及规则



表 18-1 函数作用阶段说明

阶段	说明
客户端请求	函数作用在客户端发起请求到CDN节点收到请求之间，例如访问URL重写。
节点回源	函数作用在CDN节点收到请求且节点无缓存，回源请求资源之前，例如回源URL改写。

表 18-2 规则参数说明

参数	说明
匹配类型	<p>所有文件：所有文件都遵循当前设置的规则。</p> <p>文件名后缀：指定后缀类型的文件遵循当前设置的规则。</p> <p>目录路径：指定目录内的文件遵循当前设置的规则。</p> <p>全路径：完整目录下的文件遵循当前设置的规则。</p>
匹配内容	<p>匹配类型为所有文件：此时无需填写内容。</p> <p>匹配类型为文件名后缀：</p> <ul style="list-style-type: none"> 输入首字符为“.”。 多个文件名后缀使用“,”分隔，如.jpg,.zip,.exe。 输入的文件名后缀总数不超过20个。 <p>匹配类型为目录路径：</p> <ul style="list-style-type: none"> 输入要求以“/”作为首字符。 多个目录路径使用“,”分隔，如/test/folder01,/test/folder02。 输入的目录路径总数不超过20个。 <p>匹配类型为全路径：</p> <ul style="list-style-type: none"> 输入要求以“/”作为首字符，如/test/index.html或/test/*.jpg。 支持匹配指定目录下的具体文件，或者带通配符“*”的文件。
函数	选择已经在FunctionGraph创建的函数，如果暂未创建函数，可单击“创建函数”前往FunctionGraph服务创建。
函数版本	选择函数对应的版本。
优先级	<p>如果一个阶段有多条规则，优先级可以决定先执行哪条规则。</p> <ul style="list-style-type: none"> 取值为1~100的整数，数值越大优先级越高。 优先级具有唯一性，两条规则不能使用同一个优先级。

步骤4 配置完成后，单击“保存”。

部署过程大约持续5分钟，请耐心等待，完成后会将函数配置部署到边缘节点。部署成功后您可以通过访问CDN来触发函数，从而在CDN边缘节点运行脚本。

图 18-7 保存配置



----结束

19 CLI 命令参考

19.1 KooCLI 工具介绍

FunctionGraph服务提供命令行界面工具，用来执行函数管理、触发器管理、别名管理、函数调用等操作。

获取 CLI 工具

CLI工具支持64位的Linux x86系统及64位Windows系统，以及Mac系统，具体下载地址如表19-1所示。

表 19-1 下载地址

系统	软件包及校验文件	参考文档
Linux	各系统类型软件包及校验文件下载请参见 CLI软件包和校验文件下载 。	KooCLI介绍
Windows		
Mac		

19.2 安装 KooCLI 命令行工具

1. 安装KooCLI命令行工具，请参见[在Linux系统上安装KooCLI](#)。
2. 获取访问密钥（Access Key ID和Secret Access key）
 - 如果您有登录密码，可以登录控制台，可以在我的凭证中获取自己的访问密钥AK/SK。请参见：[新增访问密钥](#)。可以下载得到AK/SK文件，文件名一般为：credentials.csv。如下图所示，文件包含了用户名称（User Name），AK（Access Key Id），SK（Secret Access Key）。

图 19-1 credentials.csv 文件内容

A	B	C
User Name	Access Key Id	Secret Access Key
CI	PI	zr175uCy

- 如果您没有登录密码，不能登录控制台，在访问密钥异常丢失或者需要重置时，可以请账号管理员在IAM中生成您的访问密钥，并发送给您。请参见：[管理IAM用户访问密钥](#)
3. 获取Region Name，请参见：[地区和终端节点](#)。

图 19-2 获取区域

区域名称	区域
非洲-约翰内斯堡	af-south-1

4. 初始化KooCLI命令行工具
 执行如下命令初始化KooCLI命令行工具：
 hcloud configure init

需要输入Access Key ID、Secret Access key和Region Name，初始化成功如图3所示。

图 19-3 初始化成功

```
[root@ecs-function-deploy ~]# hcloud configure init
开始初始化配置,其中"Secret Access Key"输入内容匿名化处理,获取参数可参考'https://usermanual-hcli/hcli_09.htm'
Access Key ID [required]:
Secret Access Key [required]: *****
Region Name:
*****
*****
*****      初始化配置成功      *****
*****
*****
```

5. 执行如下命令，查看FunctionGraph支持命令。如图4所示，“Available Operations”中返回了FunctionGraph服务支持的operation列表。
 hcloud FunctionGraph --help

图 19-4 FunctionGraph 服务支持的 operation 列表

```
[root@ecs-function-deploy ~]# hcloud FunctionGraph --help
KooCLI(Koo Command Line Interface) Version 3.2.8 Copyright(C) 2020-2022 www.huaweicloud.com

Usage:
  hcloud FunctionGraph <operation> --param1=value1 --param2=value2 ...

Service:
  FunctionGraph

Available Operations:
  AsyncInvokeFunction          InvokeFunction              ShowLtsLogDetails
  AsyncInvokeReservedFunction ListDependencies            ShowTenantMetric
  BatchDeleteFunctionTriggers ListEvents                  ShowTracing
  BatchDeleteWorkflows        ListFunctionAsyncInvocations ShowVersionAlias
  CancelAsyncInvocation        ListFunctionAsyncInvokeConfig ShowWorkflow
  CreateDependency             ListFunctionStatistics      ShowWorkflowMetric
  CreateEvent                  ListFunctionTriggers        ShowWorkflowExecution
  CreateFunction                ListFunctionVersions        StartSyncWorkflowExecution
  CreateFunctionTrigger         ListFunctions                StartWorkflowExecution
  CreateFunctionVersion         ListQuotas                   StopWorkflow
  CreateVersionAlias            ListStatistics               UpdateDependency
  CreateWorkflow                ListVersionAliases          UpdateEvent
  DeleteDependency              ListWorkflowExecutions      UpdateFunctionAsyncInvokeConfig
  DeleteEvent                   ListWorkflows                UpdateFunctionCode
  DeleteFunction                RetryWorkflow                UpdateFunctionConfig
  DeleteFunctionAsyncInvokeConfig ShowDependency               UpdateFunctionMaxInstanceConfig
  DeleteFunctionTrigger         ShowEvent                    UpdateFunctionReservedInstances
  DeleteVersionAlias            ShowFunctionAsyncInvokeConfig UpdateTracing
  EnableLtsLogs                 ShowFunctionCode             UpdateTrigger
  ExportFunction                ShowFunctionConfig           UpdateVersionAlias
  ImportFunction                 ShowFunctionTrigger          UpdateWorkflow

`hcloud FunctionGraph <operation> --help`
operation
```

执行如下命令，查看“调用函数”操作的帮助信息。如图5所示，返回如下信息。

```
hcloud FunctionGraph InvokeFunction --help
```

图 19-5 “调用函数”操作的帮助信息

```
hcloud FunctionGraph InvokeFunction --help
KooCLI (Koo Command Line Interface) Version 3.2.8 Copyright (C) 2020-2022 www.huaweicloud.com
Service:
  FunctionGraph
Description:
  .....
Method:
  POST
Params:
  --cli-region
  required string ..... cli-region
  --function_urn
  required string path ..... URN, ..... FunctionGraph .....
  --project_id
  required string path ..... ID, ..... cli-project-id
  --{*}
  required object body
  --X-Cff-Request-Version
  optional string header ..... v0,v1
  v0: .....
  v1: ..... json, sdk .....
  --X-Cff-Log-Type
  optional string header ..... :tail( ..... 4k ..... ) ..... ( ..... )
```

19.3 函数调用

调用函数首先需要获取函数的urn，如图19-6所示。

图 19-6 获取函数 URN



同步调用

执行同步调用示例命令，参数说明请参见表1。

```
hcloud FunctionGraph InvokeFunction --cli-region="ap-southeast-1" --X-Cff-Log-Type="tail" --X-Cff-Request-Version="v1" --function_urn="urn:fss:cn-east-3:*****:function:default:hcloud-invoke:latest" --project_id="*****" --key="value"
```

表 19-2 参数说明

参数	是否必须	描述
--cli-region	是	调用函数所在region。
--function_urn	是	函数的URN (Uniform Resource Name) ， 唯一标识函数。
--project_id	是	租户Project ID。

参数	是否必须	描述
--X-Cff-Log-Type	否	tail (header里面返回函数执行后的4K日志)，或者为空 (不返回日志)。
X-CFF-Request-Version	否	返回体格式：取值v0, v1。 <ul style="list-style-type: none"> v0: 默认返回文本格式。 v1: 默认返回json格式，sdk需要使用此值。
Body	是	通过--key="value"方式指定请求体，其对应的json结构：{ "key":"value"}。

输出结果如[图2](#)所示，输出的响应参数请参见[表2](#)。

图 19-7 输出结果

```

{
  "result": {
    "statusCode": 200,
    "isBase64Encoded": false,
    "body": "{\"key\":\"value\", \"lubanops-gtrace-id\": \"\", \"lubanops-ndomain-id\": \"\", \"lubanops-nenv-id\": \"\", \"lubanops-nspan-id\": \"\", \"lubanops-ntrace-id\": \"\", \"lubanops-sevent-id\": \"\", \"headers\": {\"Content-Type\": \"application/json\"}}",
    "log": "2022-07-05T02:44:32Z Start invoke request 'cd852b47-86b1-4021-9f74-1661af8bf68b', version: latest\n2022-07-05T02:44:32Z Finish invoke request 'cd852b47-86b1-4021-9f74-1661af8bf68b', duration: 1.077ms, billing duration: 2ms, memory used: 24.422MB\nbilling memory: 128MB",
    "status": 200,
    "request_id": "cd852b47-86b1-4021-9f74-1661af8bf68b",
    "error_code": ""
  }
}

```

表 19-3 响应参数

参数	参数类型	描述
request_id	String	请求ID
result	String	函数执行结果
log	String	函数执行返回日志
status	Integer	函数执行返回状态
error_code	String	错误码

异步调用

执行异步调用示例命令，参数说明请参见[表3](#)。

```

hcloud FunctionGraph AsyncInvokeFunction --cli-region="cn-east-3" --function_urn="urn:fss:cn-east-3:*****:function:default:hcloud-invoke:latest" --project_id="*****" --key="value"

```

表 19-4 参数说明

参数	是否必须	描述
--cli-region	是	调用函数所在region。

参数	是否必须	描述
--function_urn	是	函数的URN (Uniform Resource Name) , 唯一标识函数。
--project_id	是	租户Project ID。
请求体	是	通过--key="value"方式指定请求体, 其对应的json结构: { "key":"value"}。

输出结果如[图3](#)所示, 输出的响应参数请参见[表2](#)。

图 19-8 输出结果

```
{
  "request_id": "3977431d-c254-4e65-979e-30541936651b"
}
```

表 19-5 响应参数说明

参数	参数类型	描述
request_id	String	请求ID

20 审计

20.1 云审计服务支持的 FunctionGraph 操作列表

云审计支持的FunctionGraph操作列表如表20-1所示。

表 20-1 云审计服务支持的 FunctionGraph 操作列表

操作名称	资源类型	事件名称
创建函数	Functions	createFunction
删除函数	Functions	deleteFunction
修改函数信息	Functions	updateFunctionConfig
发布函数版本	FunctionVersions	publishFunctionVersion
删除函数版本别名	FunctionVersionsAlias	deleteVersionAlias
删除函数触发器	Trigger	deleteTrigger
创建函数触发器	Trigger	createTrigger
停用函数触发器	Trigger	disableTrigger
启用函数触发器	Trigger	enableTrigger

20.2 在 CTS 事件列表查看云审计事件

操作场景

用户进入云审计服务创建管理类追踪器后，系统开始记录云服务资源的操作。在创建数据类追踪器后，系统开始记录用户对OBS桶中数据的操作。云审计服务管理控制台会保存最近7天的操作记录。


本节介绍如何在云审计服务管理控制台查看或导出最近7天的操作记录。




- [在新版事件列表查看审计事件](#)
- [在旧版事件列表查看审计事件](#)

使用限制

- 单账号跟踪的事件可以通过云审计控制台查询。多账号的事件只能在账号自己的事件列表页面去查看，或者到组织追踪器配置的OBS桶中查看，也可以到组织追踪器配置的CTS/system日志流下面去查看。
- 用户通过云审计控制台只能查询最近7天的操作记录。如果需要查询超过7天的操作记录，您必须配置转储到对象存储服务(OBS)或云日志服务(LTS)，才可在OBS桶或LTS日志组里面查看历史事件信息。否则，您将无法追溯7天以前的操作记录。
- 云上操作后，1分钟内可以通过云审计控制台查询管理类事件操作记录，5分钟后才可通过云审计控制台查询数据类事件操作记录。
- CTS新版事件列表不显示数据类审计事件，您需要在旧版事件列表查看数据类审计事件。
- 云审计控制台对用户的操作事件日志保留7天，过期自动删除，不支持人工删除。

在新版事件列表查看审计事件

1. 登录管理控制台。
2. 单击左上角 ，选择“管理与监管管理与部署 > 云审计服务 CTS”，进入云审计服务页面。
3. 单击左侧导航树的“事件列表”，进入事件列表信息页面。
4. 事件列表支持通过高级搜索来查询对应的操作事件，您可以在筛选器组合一个或多个筛选条件：
 - 事件名称：输入事件的名称。
 - 事件ID：输入事件ID。
 - 资源名称：输入资源的名称，当该事件所涉及的云资源无资源名称或对应的API接口操作不涉及资源名称参数时，该字段为空。
 - 资源ID：输入资源ID，当该资源类型无资源ID或资源创建失败时，该字段为空。
 - 云服务：在下拉框中选择对应的云服务名称。
 - 资源类型：在下拉框中选择对应的资源类型。
 - 操作用户：在下拉框中选择一个或多个具体的操作用户。
 - 事件级别：可选项为“normal”、“warning”、“incident”，只可选择其中一项。
 - normal：表示操作成功。
 - warning：表示操作失败。
 - incident：表示比操作失败更严重的情况，例如引起其他故障等。
 - 企业项目ID：输入企业项目ID。
 - 访问密钥ID：输入访问密钥ID（包含临时访问凭证和永久访问密钥）。
 - 时间范围：可选择查询最近1小时、最近1天、最近1周的操作事件，也可以自定义最近7天内任意时间段的操作事件。

5. 在事件列表页面，您还可以导出操作记录文件、刷新列表、设置列表展示信息等。
 - 在搜索框中输入任意关键字，按下Enter键，可以在事件列表搜索符合条件的数据。
 - 单击“导出”按钮，云审计服务会将查询结果以.xlsx格式的表格文件导出，该.xlsx文件包含了本次查询结果的所有事件，且最多导出5000条信息。
 - 单击按钮，可以获取到事件操作记录的最新信息。
 - 单击按钮，可以自定义事件列表的展示信息。启用表格内容折行开关，可让表格内容自动折行，禁用此功能将会截断文本，默认停用此开关。
6. 关于事件结构的关键字段详解，请参见[事件结构](#)“云审计服务事件参考 > 事件结构”章节和[事件样例](#)“云审计服务事件参考 > 事件样例”章节。
7. （可选）在新版事件列表页面，单击右上方的“返回旧版”按钮，可切换至旧版事件列表页面。

在旧版事件列表查看审计事件


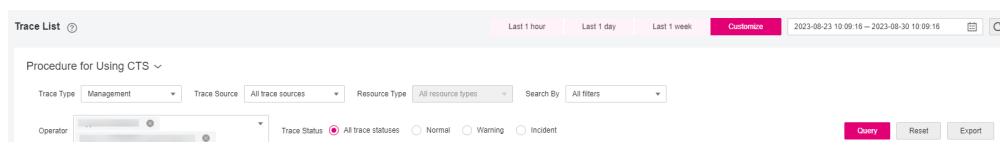


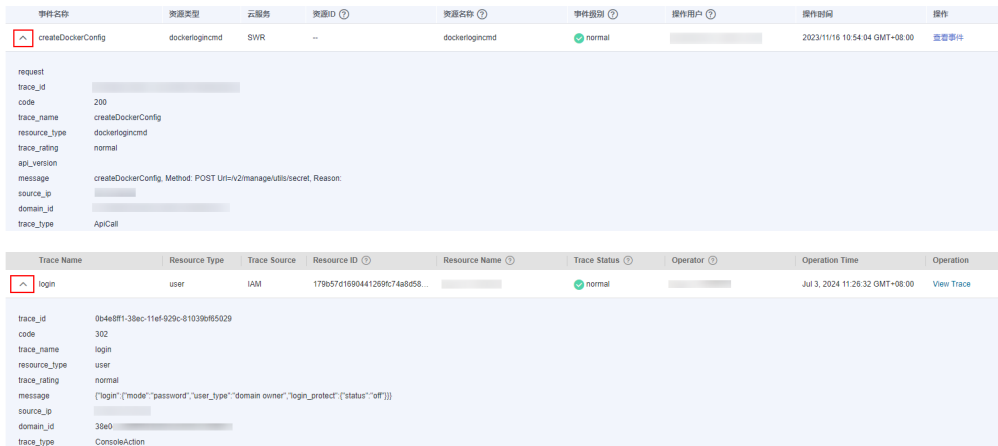
1. 登录管理控制台。
2. 单击左上角，选择“管理与监管管理与部署 > 云审计服务 CTS”，进入云审计服务页面。
3. 单击左侧导航树的“事件列表”，进入事件列表信息页面。
4. 用户每次登录云审计控制台时，控制台默认显示新版事件列表，单击页面右上方的“返回旧版”按钮，切换至旧版事件列表页面。
5. 事件列表支持通过筛选来查询对应的操作事件，如[图20-1](#)所示。当前事件列表支持四个维度的组合查询，详细信息如下：

图 20-1 筛选框

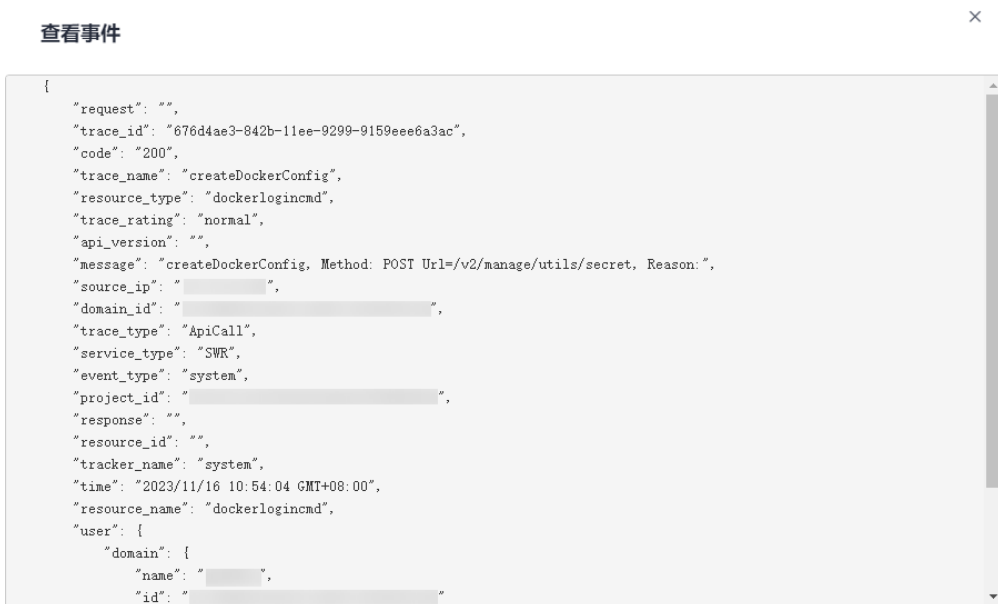


- 事件类型、云服务、资源类型和筛选类型，在下拉框中选择查询条件。
 - 筛选类型按资源ID筛选时，还需手动输入某个具体的资源ID。
 - 筛选类型按事件名称筛选时，还需选择某个具体的事件名称。
 - 筛选类型按资源名称筛选时，还需选择或手动输入某个具体的资源名称。
- 操作用户：在下拉框中选择某一具体的操作用户，此操作用户指用户级别，而非租户级别。
- 事件级别：可选项为“所有事件级别”、“Normal”、“Warning”、“Incident”，只可选择其中一项。
- 时间范围：可选择查询最近1小时、最近1天、最近1周的操作事件，也可以自定义最近7天内任意时间段的操作事件。

6. 选择完查询条件后，单击“查询”。
7. 在事件列表页面，您还可以导出操作记录文件和刷新列表。
 - 单击“导出”按钮，云审计服务会将查询结果以CSV格式的表格文件导出，该CSV文件包含了本次查询结果的所有事件，且最多导出5000条信息。
 - 单击按钮，可以获取到事件操作记录的最新信息。
8. 在需要查看的事件左侧，单击展开该记录的详细信息。



9. 在需要查看的记录右侧，单击“查看事件”，会弹出一个窗口显示该操作事件结构的详细信息。



10. 关于事件结构的关键字段详解，请参见《云审计服务用户指南》中的[事件结构](#)“云审计服务事件参考 > 事件结构”章节和[事件样例](#)“云审计服务事件参考 > 事件样例”章节。
11. （可选）在旧版事件列表页面，单击右上方的“体验新版”按钮，可切换至新版事件列表页面。