无服务器云函数

管理云函数触发器

产品文档

腾讯云

文档目录

管理云函数触发器

什么是函数触发器

最近更新时间：2018-07-03 10:14:43

腾讯云无服务器云函数是典型的事件触发（Event-Triggered）形态的无服务器运行环境，核心组件是 SCF 函数和事件源。其中，事件源是发布事件（Event）的腾讯云服务或用户自定义代码，SCF 函数是事件的处理者，而函数触发器就是管理函数和事件源对应关系的集合。例如以下场景：

- 图像/视频处理：用户上传图片时将图片切割成合适的尺寸。用户使用该应用上传照片，应用将这些用户照片存储到 COS 中并且创建每个用户照片的缩略图，并在用户页面上显示这些缩略图。本场景下，您需要选择 COS 作为事件源，在文件创建时将事件（Event）发布给 SCF 函数，事件数据提供关于存储桶和文件的所有信息。
  = 数据处理：半夜 12 点，分析一天来收集的数据（比如 clickstream）并生成报告。本场景下，您需要选择 定时器 作为事件源，在一个特定时间将事件（Event）发布给 SCF 函数。
- 自定义的应用程序：在您的某个应用程序中调用第一个图像处理 SCF 函数，作为应用程序的一个模块。本场景下，您需要该应用程序中自行调用 InvokeFunction 来发布事件（Event）。

这些事件源可以是以下任意之一：

- 内部事件源：这些是经过预配置可与 SCF 一起使用的腾讯云云服务。目前 SCF 支持 COS 和定时器两种内部事件源，当您配置了这些事件源触发函数时，函数将在出现事件时被自动调用。事件源和函数的关联关系（即事件源映射）将在事件源侧维护。例如，COS 提供 Bucket 通知配置 API。使用此 API，您可以将 Bucket 事件和函数绑定起来。
- 自定义应用程序：您可以让自定义应用程序发布事件和调用 SCF 函数。

示例 1：COS发布事件并调用函数

您可以配置 COS 的事件源映射，决定 COS 在发生何种行为时触发 SCF 函数（如 PUT、DELETE 对象等）。COS 的事件源映射存储在 COS 中，使用存储桶通知功能，引导 COS 在出现特定事件类型时调用函数：

- 创建 COS 触发器
- 用户在存储桶中创建/删除对象
- COS 检测到对象创建/删除事件。
- COS 自动调用函数，将根据存储在 COS 配置中的事件源映射明确应该调用哪个函数。将 Bucket 及 Object 信息作为事件数据传递给函数。

示例 2：定时器发布时间并调用函数

定时器的事件源映射将保存在 SCF 函数配置中，决定何时自动触发函数：

- 创建定时触发器
- 该定时器在配置时间时自动调用函数

示例 3：自定义应用程序调用函数

如果您需要在自定义应用程序中调用某个 SCF 函数，在这种情况下您不需要配置函数触发器，也不需设置事件源映射。此时，事件源使用 Invoke API。

- 自定义应用程序使用 InvokeFunction API 调用函数，自行传入事件数据。
- 函数接收到触发请求并执行
- 如果使用了同步调用方式，函数将向应用程序返回结果

> **注意：**
> 在此示例中，由于自定义应用程序和函数均为同一个用户生产的，可以指定用户凭证（APPID、SecretId 和 SecretKey）。

注意事项

1. 目前单个云函数支持 2 个 COS 触发器、2 个定时触发器、2 个 CMQ Topic 触发器。
2. 由于不同云服务的限制，事件源映射关系有着特定的限制。例如：对于 COS 触发器而言，同一个 COS Bucket 的相同事件（例如文件上传），不能触发多个不同的函数。

支持的触发器类型

API 网关触发器

最近更新时间：2018-07-12 16:51:49

用户可以通过编写 SCF 云函数来实现 Web 后端服务，并通过 API 网关对外提供服务。API 网关会将请求内容以参数形式传递给函数，并将函数返回作为响应返回给请求方。

API 网关触发器具有以下特点：

- **Push 模型**：API 网关在接受到 API 请求后，如果 API 在网关上的后端配置了对接云函数，函数将会被触发运行，同时 API 网关会将 API 请求的相关信息，例如具体接受到请求的服务和 API 规则，请求的实际路径，方法，请求的 path，header，query 等参数，全部封装为请求入参，以 event 入参的形式发送给触发的函数。
- **同步调用**：API 网关以同步调用的方式来调用函数，会在 API 网关中配置的超时时间未到前等待函数返回。有关调用类型的更多信息，请参阅 调用类型。

API 网关触发器配置

API 网关触发器配置 **不在云函数中进行配置**，而是在 API 网关中进行。在 API 网关中配置 API 规则时，后端配置可选 Cloud Function，在选择 Cloud Function 后，即可选择与 API 服务相同地域的云函数。

在 API 网关配置对接云函数时，同样会配置超时时间。API 网关中的请求超时时间，和云函数的运行超时时间，两者分别生效。超时规则如下：

- API 网关超时时间 > 云函数超时时间：云函数超时先生效，API 请求响应为 `200 http code` ，但返回内容为云函数超时报错内容。
- API 网关超时时间 < 云函数超时时间：API 网关超时先生效，API 请求返回 `5xx http code` ，标识请求超时。

API 网关触发器绑定限制

API 网关中，一条 API 规则仅能绑定一个云函数，但一个云函数可以被多个 API 规则绑定为后端。同时，目前 API 网关触发器仅支持同地域云函数绑定，即广州区创建的云函数，仅支持被广州区创建的 API 服务中规则所绑定和触发。如果您想要使用特定地域的 API 网关配置来触发云函数，可以通过在对应地域下创建函数来实现。

请求与响应

针对 API 网关发送到云函数的请求处理方式，和云函数响应给 API 网关的返回值处理方式，称为请求方法和响应方法。请求方法和响应方法规划和实现的分别有透传方式和集成方式。

集成请求与透传请求

集成请求，是指 API 网关会将 HTTP 请求内容，转换为请求数据结构；请求数据结构作为函数的 event 入参，传递给函数并进行处理。具体请求数据结构说明见如下。

透传请求，是指 API 网关会将 HTTP 请求的 Body 内容作为函数的 event 入参传递给函数，透传请求目前还处于规划过程中。对于透传请求，要求 HTTP 请求的 body 内容为 JSON 数据结构内容。

> **注意：**
> 当前 API 网关触发器触发云函数时，全部使用的为集成请求。

API 网关触发器的集成请求事件消息结构

在 API 网关接收到请求时，会将类似以下 JSON 格式的事件数据发送给绑定的云函数。

```
{
  "requestContext": {
    "serviceName": "testsvc",
    "path": "/test/{path}",
    "httpMethod": "POST",
    "requestId": "c6af9ac6-7b61-11e6-9a41-93e8deadbeef",
    "identity": {
      "secretId": "abdcdxxxxxxxsdfs"
    },
    "sourceIp": "10.0.2.14",
    "stage": "prod"
  },
  "headers": {
    "Accept-Language": "en-US,en,cn",
    "Accept": "text/html,application/xml,application/json",
    "Host": "service-3ei3tii4-251000691.ap-guangzhou.apigateway.myqloud.com",
    "User-Agent": "User Agent String"
  },
  "body": "{\"test\":\"body\"}",
  "pathParameters": {
    "path": "value"
  },
  "queryStringParameters": {
    "foo": "bar"
  },
```

```
  "headerParameters":{
    "Refer": "10.0.2.14"
  },
  "stageVariables": {
    "stage": "test"
  },
  "path": "/test/value",
  "query": "foo=bar&bob=alice",
  "httpMethod": "POST"
}
```

数据结构内容详细说明如下：

结构名	内容
requestContext	请求来源的 API 网关的配置信息、请求标识、认证信息、来源信息。其中： • serviceName，path，httpMethod 指向 API 网关的服务、API 的路径和方法； • stage 指向请求来源 API 所在的环境； • requestId 标识当前这次请求的唯一 ID； • identity 标识用户的认证方法和认证的信息； • sourceIp 标识请求来源 IP
path	记录实际请求的完整 Path 信息
httpMethod	记录实际请求的 HTTP 方法
query	记录实际请求的完整 Query 内容
body	记录实际请求的完整 Body 内容
headers	记录实际请求的完整 Header 内容
pathParameters	记录在 API 网关中配置过的 Path 参数以及实际取值
queryStringParameters	记录在 API 网关中配置过的 Query 参数以及实际取值
headerParameters	记录在 API 网关中配置过的 Header 参数以及实际取值

注意：

- 在 API 网关迭代过程中， requestContext 内的内容可能会增加更多。目前会保证数据结构内容仅增加，不删除，不对已有结构进行破坏。
- 实际请求时的参数数据可能会在多个位置出现，可根据业务需求选择使用。

集成响应与透传响应

集成响应，是指 API 网关会将云函数的返回内容进行解析，并根据解析内容构造 HTTP 响应。通过使用集成响应，可以通过代码自主控制响应的状态码、headers、body 内容，可以实现非 JSON 格式的内容响应，例如响应 XML、HTML、甚至 JS 内容。在使用集成响应时，需要按如下说明的数据结构返回，才可以被 API 网关成功解析，否则会报错 `{"errno":403,"error":"Invalid scf response format. please check your scf response format."}` 。

透传响应，是指 API 网关将云函数的返回内容直接传递给 API 请求方。通常这种响应的数据格式直接确定为 JSON 格式，状态码根据函数执行的状态定义，函数执行成功即为 200 状态码。通过透传响应，用户可以自行获取到 JSON 格式后在调用位置解析结构，获取结构内的内容。

> **注意：**
> 当前 API 网关处理响应的方式默认为透传响应，如需开启集成响应，请在 API 配置中的后端配置位置，勾选 **启用集成响应**，并在代码中按如下说明的数据结构返回内容。

API 网关触发器的集成响应返回数据结构

在 API 网关设置为集成响应时，需要返回类似如下内容的数据结构。

```
{
    "isBase64": False,
    "statusCode": 200,
    "headers": {"Content-Type":"text/html"},
    "body": "<html><body><h1>Heading</h1><p>Paragraph.</p></body></html>"
}
```

数据结构内容详细说明如下：

结构名	内容
isBase64	指明 body 内的内容是否为 Base64 编码后的二进制内容，取值需要为 JSON 格式的 true 或 false
statusCode	HTTP 返回的状态码，取值需要为 int 值
headers	HTTP 返回的头部内容，取值需要为多个 key-value 对象
body	HTTP 返回的 body 内容

COS 触发器

最近更新时间：2018-07-03 10:15:15

用户可以编写 SCF 函数来处理 COS Bucket 中的对象创建和对象删除事件。COS 可将事件发布给 SCF 函数并将事件数据作为参数来调用该函数。用户可以在 COS Bucket 中添加存储桶通知配置，该配置可标识触发函数的事件类型和希望调用的函数名称等信息，更多内容请参考 PutBucketNotification 接口。

COS 触发器具有以下特点：

- **Push 模型**：COS 会监控指定的 Bucket 动作（事件类型）并调用相关函数，将事件数据推送给 SCF 函数。在推模型中使用 Bucket 通知来保存 COS 的事件源映射。
- **异步调用**：COS 始终使用异步调用类型来调用函数，结果不会返回给调用方。有关调用类型的更多信息，请参阅调用类型。

COS 触发器属性

- COS Bucket（必选）：配置的COS Bucket，仅支持选择同地域下的 COS 存储桶。
- 事件类型（必选）：目前支持“文件上传”和“文件删除”两种类型，决定了触发器何时触发函数。例如，选择“文件上传”时，会在该COS Bucket中有文件上传时触发该函数。

COS 触发器使用限制

为了避免 COS 的事件生产投递出现错误，COS 针对每个 Bucket 的每个事件（如文件上传/文件删除等），限制只能绑定一个可触发的函数。因此，在您创建 COS 触发器时，请不要将多个函数的触发器绑定至同一个 COS Bucket 的同一个事件上。

同时，目前 COS 触发器仅支持同地域 COS Bucket 事件触发，即广州区创建的 SCF 函数，在配置 COS 触发器时，仅支持选择广州区（华南）的 COS Bucket。如果您想要使用特定地域的 COS Bucket 事件来触发 SCF 函数，可以通过在对应地域下创建函数来实现。

COS 触发器的事件消息结构

在指定的 COS Bucket 发生对象创建或对象删除事件时，会将类似以下的 JSON 格式事件数据发送给绑定的 SCF 函数。

```
{
  "Records":[
```

```
    {
      "event": {
        "eventVersion":"1.0",
        "eventSource":"qcs::cos",
        "eventName":"cos:ObjectCreated:*,
        "eventTime":"1970-01-01T00:00:00.000Z",
        "eventQueue":"qcs:0:cos:gz:1251111111:cos",
        "requestParameters":{
          "requestSourceIP": "111.111.111.111",
          "requestHeaders":{
            "Authorization": "Example"
          }
        }
       },
        "cos":{
          "cosSchemaVersion":"1.0",
          "cosNotificationId":"设置的或返回的 ID",
          "cosBucket":{
            "name":"bucketname",
            "appid":"1251111111",
            "region":"gz",
          },
          "cosObject":{
            "key":"/test.jpg",
            "size":"1024",
            "meta":{
              "Content-Type": "text/plain",
              "x-cos-meta-test": "自定义的 meta",
              "x-image-test": "自定义的 meta"
            },
            "url": "访问文件的源站url"
          }
        }
      }
    ]
}
```

数据结构内容详细说明如下：

结构名	内容
Records	列表结构，可能有多条消息合并在列表中
event	记录事件信息，包括事件版本、事件源、事件名称、时间、队列信息
cos	记录事件对应的 COS 信息

结构名	内容
cosBucket	记录具体事件发生的 Bucket，包含 Bucket 名称，地域，所属用户 APPID
cosObject	记录具体事件发生的对象，包含对象文件路径，大小，自定义元数据，访问 URL
subscriptionName	记录云函数在 topic 处的订阅名称

定时触发器

最近更新时间：2018-07-31 11:25:08

用户可以编写 SCF 函数来处理定时任务。定时器会在指定时间自动触发 SCF 函数。定时触发器具有以下特点：

- **Push 模型**：定时器指定时间到达时直接调用相关函数的 Invoke 接口来触发函数。该事件源映射关系保存在 SCF 函数中。
- **异步调用**：定时器始终使用异步调用类型来调用函数，结果不会返回给调用方。有关调用类型的更多信息，请参阅调用类型。

定时触发器属性

- 定时器名称（必选）：最大支持 60 个字符，支持 `a-z`，`A-Z`，`0-9`，`-` 和 `_`。必须以字母开头，且一个函数下不支持同名的多个定时触发器。
- 触发周期（必选）：指定的函数触发时间。用户可以使用控制台上的默认值，或选择自定义标准的 CRON 表达式来决定何时触发函数。有关 CRON 表达式的更多信息，请参考下面的内容。

Cron 表达式

创建定时触发器时，用户能够使用标准的 cron 表达式的形式自定义何时触发。

Cron 表达式语法

Cron 表达式有五个必需字段，按空格分隔。

第一位	第二位	第三位	第四位	第五位
分钟	小时	日	月	星期

其中，每个字段都有相应的取值范围：

字段	值	通配符
分钟	0-59	, - * /
小时	0-23	, - * /
日	1-31	, - * /
月	1-12 或 JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC	, - * /

字段	值	通配符
星期	0-6 或 MON,TUE,WED,THU,FRI,SAT,SUN；其中 0 指星期一，1 指星期二，依次类推	, - * /

通配符分别代表了以下意义：

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

注意事项

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

示例

下面展示了一些 Cron 表达式和相关含义的示例：

- `15 23 * * *` 每天晚上 23:15 运行
- `0 18 * * MON-FRI` 每个工作日下午 6:00 运行
- `0 8 1 * *` 每个月 1 号上午 8:00 运行
- `0/15 * * * *` 每 15 分钟运行一次

定时触发器入参说明

定时触发器在触发函数时，暂时无入参，即 event 内容为空。可通过此特性，判断触发来源是否为定时触发器。

CMQ Topic 触发器

最近更新时间：2018-07-03 10:16:11

用户可以编写 SCF 函数来处理 CMQ Topic 主题队列中收到的消息。CMQ Topic 可将消息传递给 SCF 函数并将消息内容和相关信息作为参数来调用该函数。

CMQ Topic 主题队列触发器具有以下特点：

- **Push 模型**：CMQ Topic 主题队列在接受到消息后，会将消息推送给所有订阅该主题的订阅者，配置了触发 SCF 函数的情况下，函数也会被作为订阅者接收到队列的推送。在推模型中，CMQ Topic 主题队列会保存对 SCF 云函数的事件源映射。
- **异步调用**：CMQ Topoic 主题队列始终使用异步调用类型来调用函数，结果不会返回给调用方。有关调用类型的更多信息，请参阅调用类型。

CMQ Topic 主题队列触发器属性

- CMQ Topic（必选）：配置的 CMQ Topic 主题队列，仅支持选择同地域下的 CMQ 队列。

CMQ Topic 触发器绑定限制

由于 CMQ Topic 主题队列，在单个 Topic 下最多支持 100 个订阅者。因此在 SCF 云函数触发器绑定时，如果达到此限制，可能绑定失败。在未达到此限制前，一个 Topic 下可以绑定多个 SCF 云函数。

同时，目前 CMQ Topic 触发器仅支持同地域 CMQ Topic 消息发，即广州区创建的 SCF 函数，在配置 CMQ Topic 触发器时，仅支持选择广州区（华南）的 CMQ Topic。如果您想要使用特定地域的 CMQ Topic 消息来触发 SCF 函数，可以通过在对应地域下创建函数来实现。

CMQ Topic 触发器的事件消息结构

在指定的 CMQ Topic 主题队列接受到消息时，会将类似以下的 JSON 格式事件数据发送给绑定的 SCF 函数。

```
{
  "Records": [
    {
      "CMQ": {
        "type": "topic",
        "topicOwner":120xxxxx,
        "topicName": "testtopic",
```

```
        "subscriptionName":"xxxxxx",
        "publishTime": "1970-01-01T00:00:00.000Z",
        "msgId": "123345346",
        "requestId":"123345346",
        "msgBody": "Hello from CMQ!",
        "msgTag": ["tag1","tag2"]
      }
    }
  ]
}
```

数据结构内容详细说明如下：

结构名	内容
Records	列表结构，可能有多条消息合并在列表中
CMQ	标识数据结构来源为 CMQ 消息队列
type	使用 type 区分消息来源为 topic 或 queue
topicOwner	记录 topic 所有者账号 ID
topicName	记录 topic 名称
subscriptionName	记录云函数在 topic 处的订阅名称
publishTime	记录消息的发布时间
msgId	记录消息的唯一 ID
requestId	记录消息推送的请求 ID
msgBody	记录消息内容
msgTag	通过列表结构记录消息标签

CKafka 触发器

最近更新时间：2018-07-03 10:17:42

用户可以编写 SCF 函数来处理 CKafka 中收取到的消息。SCF 后台模块可以作为消费者消费 CKafka 中的消息，并将消息传递给 SCF 函数。

CKafka 触发器具有以下特点：

- **Pull 模型**：SCF 的后台模块作为消费者，连接 CKafka 实例并消费消息。在后台模块获取到消息后，会将消息封装到数据结构中并调用指定的函数，将消息数据传递给云函数。
- **异步调用**： CKafka 触发器始终使用异步调用类型来调用函数，结果不会返回给调用方。有关调用类型的更多信息，请参阅 调用类型。

CKafka 触发器属性

- CKafka 实例：配置连接的 CKafka 实例，仅支持选择同地域下的实例。
- Topic：支持在 CKafka 实例中已经创建的 Topic。
- 最大批量消息数：在拉取并批量投递给当前云函数时的最大批量数。根据消息大小，写入速度，每次触发云函数并投递时的消息，数量不一定均能达到最大消息数，而是一个变动值，消息个数在 1 ~ 最大消息数之间。

CKafka 消费及消息传递方法

由于 CKafka 消息无主动推送能力，需要消费方通过拉取方式，拉取到消息并进行消费。因此，在配置 CKafka 触发器后，SCF 云函数后台会通过启动 CKafka 消费模块，作为消费者，在 CKafka 中创立独立的消费组进行消费。

SCF 云函数后台的消费模块，在消费到消息后，会根据一定的超时时间、累积消息数量及大小、最大批量消息数等信息，组合为事件结构并传递给云函数。在这个过程中，每次组合的消息数量不一定相同，每个事件结构内的消息个数在 **1 ~ 配置的最大消息个数** 之间。如果配置的最大消息数过大，有可能出现事件结构内的消息个数始终不会达到最大消息数的情况。

在云函数中获取到事件内容后，可以通过循环处理的方式，确保每一条消息都得到处理，而不应假定每次传递的消息个数均是恒定的。

CKafka 触发器的事件消息结构

在指定的 CKafka Topic 接收到消息时，云函数的后台消费者模块会消费到消息，并将消息组装为类似以下的 JSON 格式事件，触发绑定的函数并将数据内容作为入参传递给函数。

```
{
  "Records": [
    {
      "Ckafka": {
        "topic": "test-topic",
        "partition":1,
        "offset":36,
        "msgKey": "None",
        "msgBody": "Hello from Ckafka!"
      }
    },
    {
      "Ckafka": {
        "topic": "test-topic",
        "partition":1,
        "offset":37,
        "msgKey": "None",
        "msgBody": "Hello from Ckafka again!"
      }
    }
  ]
}
```

数据结构内容详细说明如下：

结构名	内容
Records	列表结构，可能有多条消息合并在列表中
Ckafka	标识事件来源为 CKafka
topic	消息来源 Topic
partition	消息来源的分区 ID
offset	消费偏移编号
msgKey	消息 key
msgBody	消息内容

云 API 触发器

最近更新时间：2018-07-09 11:38:27

概述

用户可以编写 SCF 函数来处理自身业务逻辑，并通过 SCF 暴露的管理接口来触发云函数。管理接口在腾讯云中统一称为云 API。通过使用 SCF 云 API 中的 Invoke 接口，用户可以按需触发调用云函数。
详细的云 API 接口调用方法可见 运行函数 API 文档。云 API 触发器具有以下特点：

- **调用模式可选**：可根据 InvocationType 参数，自行定义同步或异步触发方法。
- **自定义事件**：可根据 ClientContext 参数，自行定义触发云函数的事件或数据内容，内容需要以 JSON 格式编码。

云 API 调用

通过云 API 触发云函数，需要：

1. 按 API 接口进行鉴权；
2. 填写公共参数；
3. 根据 返回结果 解析返回结果。

另外，为了避免自行构造请求内容及解析内容，用户也可以直接使用云 API SDK 触发云函数。 云 API SDK 提供了 Python，PHP，JAVA，GO，.NET，Node.js 语言的实现，各语言 SDK 的具体使用方式可见 腾讯云 SDK 中心。