云函数

产品简介

产品文档

腾讯云

文档目录

产品简介
产品概述

最近更新时间：2024-03-21 16:31:40

腾讯云云函数（Serverless Cloud Function，SCF）是腾讯云为企业和开发者们提供的无服务器执行环境，帮助您在无需购买和管理服务器的情况下运行代码， 是实时文件处理和数据处理等场景下理想的计算平台。 您只需使用 SCF 平台支持的语言编写核心代码并设置代码运行的条件，即可在腾讯云基础设施上弹性、安全地运行代码。

计算资源的变迁

随着云服务的发展，计算资源高度抽象化，腾讯云提供了从物理服务器到云函数和横跨各种抽象程度的计算资源供用户选择。
黑石物理服务器：以物理机为扩展单位。用户完全拥有整台实体计算资源，安全性最好。
云服务器：以云服务器为扩展单位，虚拟化硬件设备。用户和其他租户共享物理机资源，仍可自行配置 CVM 的各项指标，相对部署和迭代更加简单。
容器服务：以服务为扩展单位，虚拟化操作系统。测试和生产环境完全一致，测试和部署非常轻松。
云函数：以函数为扩展单位，虚拟化运行时环境（Runtime）。是现有计算资源的最小单位，具有完全自动、一键部署、高度可扩展等特点，是轻量级服务部署非常好的选择。

无服务器的概述

无服务器（Serverless）不是表示没有服务器，而表示当您在使用 Serverless 时，您无需关心底层资源，也无需登录服务器和优化服务器，只需关注最核心的代码片段，即可跳过复杂的、繁琐的基本工作。核心的代码片段完全由事件或者请求触发，平台根据请求自动平行调整服务资源。Serverless 拥有近乎无限的扩容能力，空闲时，不运行任何资源。代码运行无状态，可以轻易实现快速迭代、极速部署。

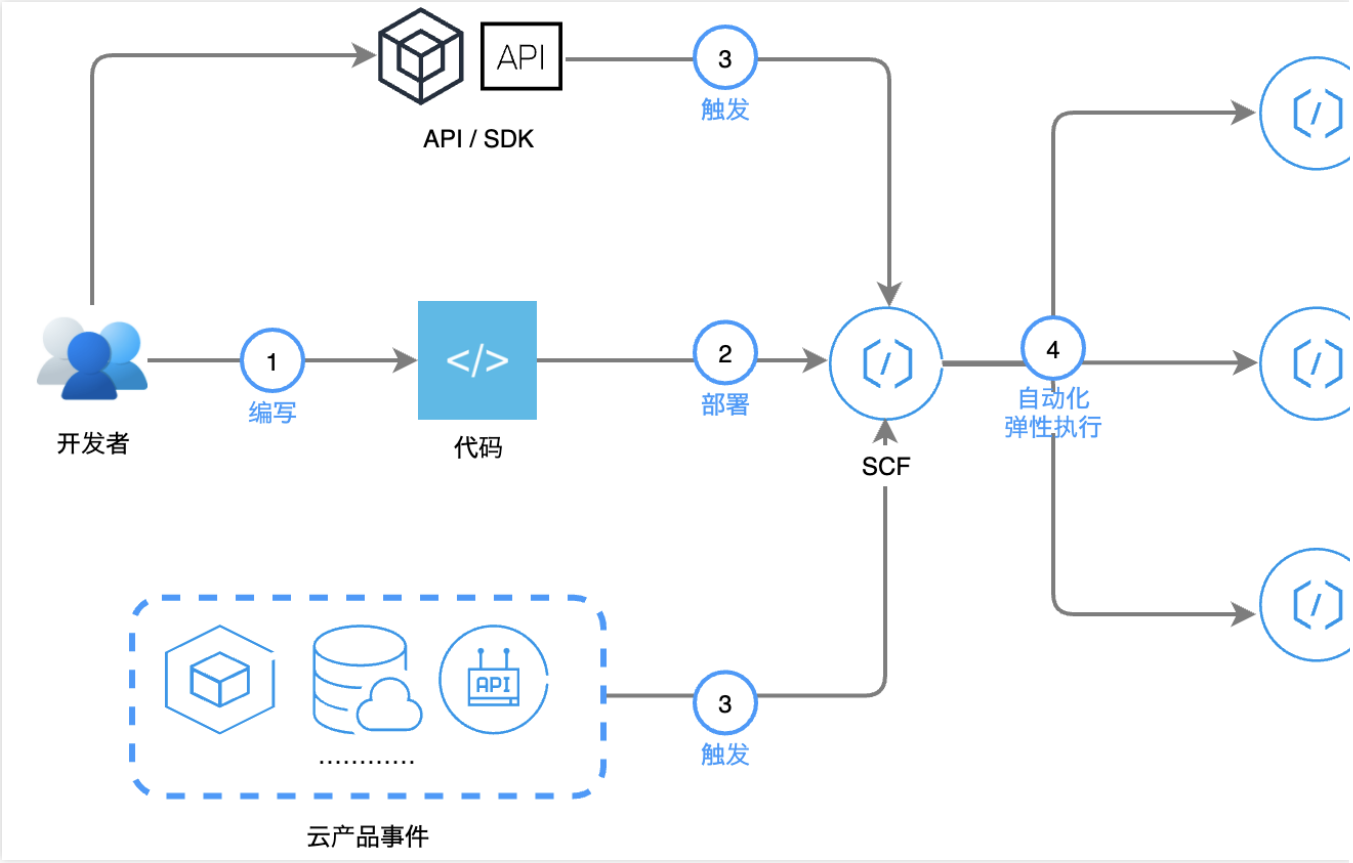
腾讯云云函数简介

腾讯云云函数是腾讯云提供的 Serverless 执行环境。您只需编写简单的、目的单一的云函数即可将它与您的腾讯云基础设施及其他云服务产生的事件关联。
使用云函数时，您只需使用平台支持的语言（Python、Node.js、PHP、Golang、Java 及 Custom Runtime）编写代码。腾讯云将完全管理底层计算资源，包括服务器 CPU、内存、网络和其他配置/资源维护、代码部署、弹性伸缩、负载均衡、安全升级、资源运行情况监控等。但这也意味着您无法登录或管理服务器、无法自定义系统和环境。

云函数自动地在同一地域内的多个可用区部署，同时提供极高的容错性。云函数在执行时将根据请求负载扩缩容，从每天几个请求到每秒数千个请求，都由云函数底层自行伸缩。您无需人工配置和介入，只需为运行中的云函数付费，即可满足不同情景下服务的可用性和稳定性。若云函数未运行，则不产生任何费用。
您可以自定义运行云函数的时机，例如，在 COS Bucket 上传时、删除文件时运行云函数、使用 Ckafka 中的消息时运行云函数、应用程序通过 SDK 调用时运行云函数，或指定云函数定期执行。您可以使用云函数作为 COS 服务的数据处理触发程序轻松实现 IFTTT 逻辑，您也可以通过构建灵活的定时自动化任务，用于覆盖手工完成的操作，轻松构建灵活可控的软件架构。

腾讯云云函数产品特性

Serverless 帮助用户脱离繁冗的开发配置工作，只需关注业务代码逻辑的编写，不用任何的基础设施建设、管理与运维开销。该服务模式降低了研发门槛，提升业务构建效率，获得了大量企业和开发者的支持。



支持多种开发工具和语言，极致的开发体验

腾讯云 Serverless 团队从多方面着手，提供可以满足多种开发场景的相关工具或能力。例如：
使用 Serverless Cloud Framework，在本地开发环境中进行项目创建、本地调试打包、一键部署上线。

通过 VS Code 插件，及 IDE 可视化操作，将函数的线上线下管理及代码编写调试，整合在一个界面中完成。VS Code IDE 及插件还提供了函数的本地管理、开发调试、上线发布功能。
支持 Web IDE，在控制台上实时的开发调试，与本地开发调试的体验相同，方便代码的调整或查看。
支持 Python、Node.js、Golang、PHP、Java 等语言，同时支持 Custom Runtime，您可以根据您的需要自定义运行环境。

多种部署方式，适应各种环境

支持控制台部署，命令行部署，SDK/API 部署，Web IDE 直接部署以及镜像部署。

多样化触发，支持更多业务场景

触发方式包括 API,SDK，以及其他多种云服务产品的事件例如 COS, API 网关等。多种触发选择支持更多使用场景。

自动化弹性执行，贴合调用曲线

在用户无感知的情况下根据调用量自动扩缩容，完美贴合调用曲线，最大程度节省资源和成本。

按需付费，毫秒级计费模式

腾讯云云函数资源使用量支持按1ms时间粒度计费，相较于按100ms粒度计费，可以为您大幅度节约成本。

相关概念

最近更新时间：2024-03-21 16:31:40

腾讯云云函数（Serverless Cloud Function，SCF）为函数即服务 （Function as a Service，FaaS）产品，提供无服务器（Serverless） 和 FaaS 的计算平台。运行方式依赖事件触发。因此在和触发事件源结合时，云函数就可以被触发源所产生的事件触发运行。

无服务器

无服务器架构说法的来源可以根据 Mike Roberts 在 Martin Fowler 的博客网站上发表的 无服务器架构 一文中得到解释。
无服务器并不是没有服务器就能够进行计算，而是对于开发者来说，无需了解底层的服务器情况，也能使用到相关资源，因此称为无服务器。
无服务器也可以从更广的角度来识别，针对无需配置和了解底层的服务器就可以直接使用的云服务，在一定程度上也可以称为无服务器。
在云函数 SCF 产品中，我们针对的是无服务器场景中的计算场景。云函数产品提供的是无服务器模式下的 FaaS 能力。

函数即服务

函数即服务提供了一种直接在云上运行无状态的、短暂的、由事件触发的代码的能力。
函数即服务和传统应用架构不同，函数服务提供的是事件触发式的运行方式，云函数不是始终运行的状态，而是在事件发生时由事件触发运行，并且在一次运行的过程中处理这一次事件。因此在云函数的代码中，仅需考虑针对一个事件的处理流程，而针对大量事件的高并发处理，由平台实现云函数的多实例并发来支持。
为了实现对高并发的支持，云函数平台提供了自动的弹性伸缩能力，会在有大量请求到来时启动更多实例来处理事件请求，也会在没有事件到来时缩减函数实例甚至到零实例。因此为了匹配自动扩缩能力，需要函数代码使用的是无状态开发方式，即不在云函数的运行内存中保留相关的状态数据并在多次运行时依赖这些状态数据。云函数的状态数据，可以依赖外部的持久存储能力例如云缓存、云数据库、云存储来进行。

触发器和触发源

任何可以产生事件，触发云函数执行的均可以被称为触发器或触发源。触发器在本身产生事件后，通过将事件传递给云函数来触发函数运行。
触发器在触发函数时，可以根据自身特点，使用同步或异步方式触发函数。同步方式触发函数时，触发器将等待函数执行完成并获取到函数执行结果；异步方式触发函数时，触发器将仅触发函数而忽略函数执行结果。

腾讯云云函数在和腾讯云的某些产品或服务对接时，也有自身实现的一些特殊方式，例如推（PUSH）模式和拉（PULL）模式。

推模式：触发器主动将事件推送至云函数平台并触发函数运行。

拉模式：云函数平台通过拉取模块，从触发器中拉取到事件并触发云函数运行。

触发事件

触发器在触发函数时会将事件传递给云函数。事件在传递时以一个特定的数据结构体现，数据结构格式在传递时均为 JSON 格式，并以函数 event 入参的方式传递给云函数。

触发事件的 JSON 数据内容，在不同的语言环境下将会转换为各自语言的数据结构或对象，无需在代码中自行进行从 JSON 结构到数据结构的转换。

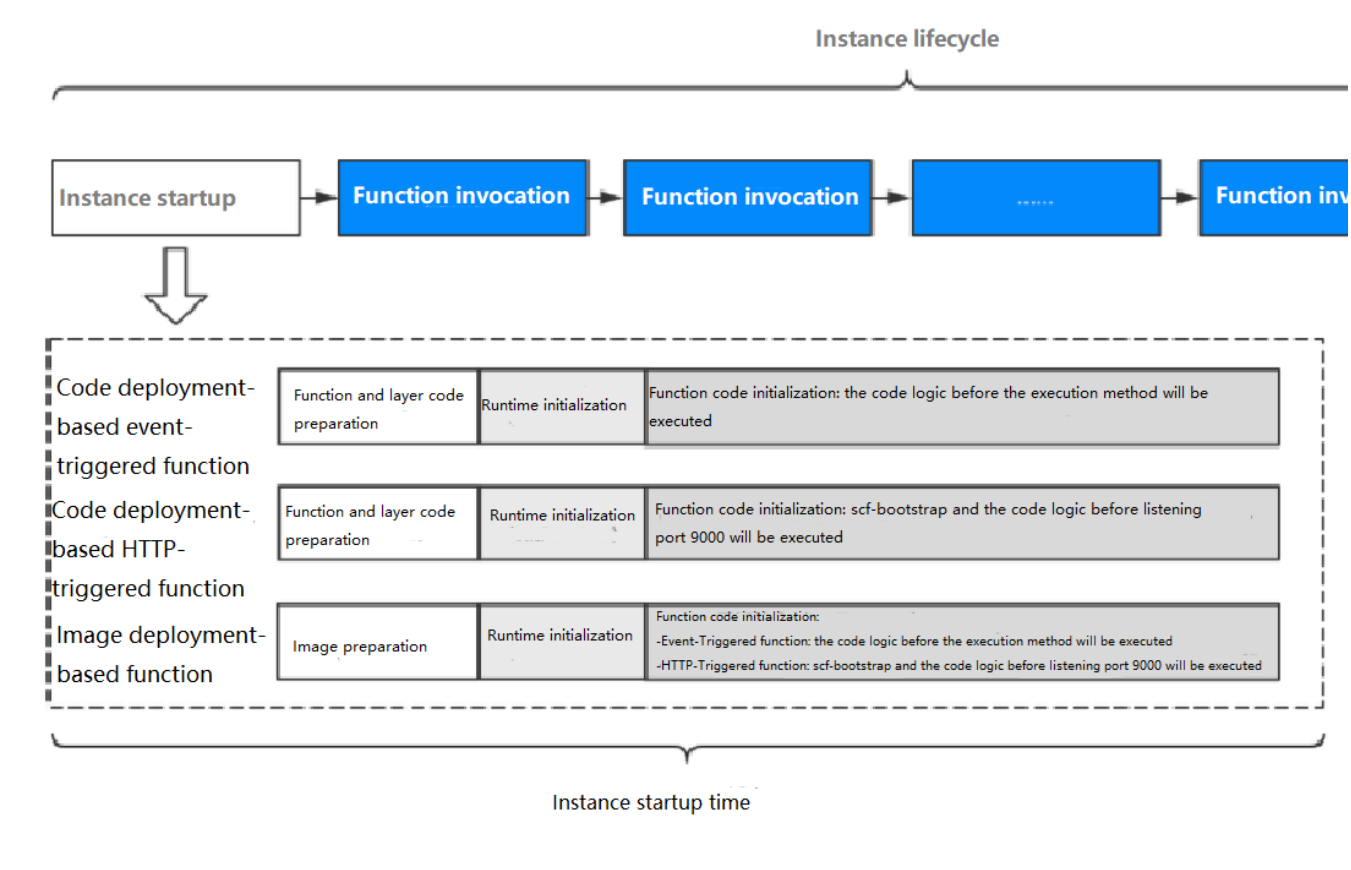
例如，在 Python 环境中，JSON 数据内容会转变为一个复杂 dict 对象，即函数的入参 event 就是一个 Python 的复杂 dict 对象。而在 Golang 或 Java 中，入参是一个需要和 event 数据结构可以匹配的对象。更具体的实现方式可以参见 开发语言说明。

工作原理

最近更新时间：2024-03-21 16:31:40

函数运行时的实例模型

云函数 SCF 将在函数接收到触发请求时为您执行函数。SCF 执行请求的资源为实例，根据函数的配置信息（如内存大小等）进行资源分配，并启动一个或多个实例处理函数请求。SCF 平台负责所有函数运行实例的创建、管理和删除清理操作，用户没有权限对其进行管理。
实例的生命周期如下图所示：



实例启动

如果请求到来时没有正在运行的实例，请求会触发实例的启动。实例启动通常需要一些时间，这会使触发实例启动的调用增加一些耗时。通常首次调用函数、更新函数或长时间未调用时重新调用函数时才会触发实例启动。
实例启动耗时将体现在函数执行日志 `Init Report` 或 `Provisioned Report` 行中的 `Coldstart` 字段中。
您可以使用 预置并发 功能提前完成实例启动，避免函数请求到来时再触发实例拉起。

实例启动耗时受函数 初始化超时时间 限制，若实例启动所需时间大于函数配置的初始化超时时间，将会导致实例启动失败。请参考下文优化实例启动耗时或适当调大初始化超时时间配置。
实例启动三个子阶段及耗时优化方法：
代码准备：平台拉取用户上传的函数代码、层或镜像，为函数执行做准备。代码准备耗时与代码包、层、镜像大小成正相关，建议尽可能的精简代码包大小，只保留函数执行必须的代码文件及依赖，以获取最短的代码准备耗时。代码准备耗时将体现在函数执行日志 `Init Report` 或 `Provisioned Report` 行中的 `PullCode` 字段中。
运行时初始化：平台按照用户的函数配置准备函数执行依赖的运行环境。运行时初始化耗时将体现在函数执行日志 `Init Report` 或 `Provisioned Report` 行中的 `InitRuntime` 字段中。
函数代码初始化：代码初始化耗时与代码逻辑复杂程度成正相关，建议尽可能优化代码逻辑，以获取最短的代码初始化耗时。函数代码初始化耗时将体现在函数执行日志 `Init Report` 或 `Provisioned Report` 行中的 `InitFunction` 字段中。
代码部署的事件函数：实例启动阶段将会运行函数配置的 执行方法 之前的代码逻辑（例如当入口函数为 main_handler ，将运行 main_handler 之前所有的代码逻辑）。
代码部署的 Web 函数：实例启动阶段将会运行启动文件 `scf_bootstrap` 和监听 9000 端口之前的代码逻辑。
镜像部署的函数：事件函数将会运行函数配置的 执行方法 之前的代码逻辑；Web 函数将会运行启动文件 `scf_bootstrap` 和监听 9000 端口之前的代码逻辑。

实例复用

为尽量减少实例启动带来的耗时，平台会尝试对后续调用复用实例。在实例处理完函数请求后会根据平台的实际情况存留一段时间以用于下次调用，在此时间段内的调用会优先复用该实例。
实例复用的意义在于：
用户代码中位于 执行方法 外部的任何声明保持已初始化的状态，再次调用函数时可以直接重用。例如，如果您的函数代码中建立了数据库连接，容器重用时可以直接使用原始连接。您可以在代码中添加逻辑，在创建新连接之前检查是否已存在连接。
每个容器在 `/tmp` 目录中提供部分磁盘空间。容器存留时该目录内容会保留，提供可用于多次调用的暂时性缓存。再次调用函数时有可能可以直接使用该磁盘内容，您可以添加额外的代码来检查缓存中是否有您存储的数据。
注意：
请勿在函数代码中假定始终重用实例，因为是否重用和单次实际调用相关，无法保证是创建新实例还是重用现有实例。

实例回收

平台会对一段时间内没有处理请求的实例进行回收。

临时磁盘空间

SCF 函数在执行过程中，都拥有一块 512MB 的临时磁盘空间 `/tmp` ，用户可以在执行代码中对该空间进行一些读写操作，也可以创建子目录，但这部分数据可能**不会**在函数执行完成后保留。因此，如果您需要对执行过程中产生的数据进行持久化存储，请使用 对象存储 COS 或 Redis/Memcached 等外部持久化存储。

调用类型

SCF 平台支持同步和异步两种调用方式来调用云函数。

同步调用

同步调用函数将会在调用请求发出后持续等待函数的执行结果返回。

异步调用

异步调用将不会等待结果返回，只发出请求并获得当前请求的 Request ID。
当发生异步调用时，异步事件会先放置到云函数内置的异步队列，然后再消费异步队列内的事件执行函数。异步队列有如下限制：
异步队列是触发器维度的，一个函数触发器一个队列。
异步事件在队列中的保留时长最大 6 小时。
异步队列的长度上限为 10 万条消息。
异步调用的重试策略也有所不同，详情请参见 重试策略。

定义函数调用类型

调用类型与函数本身的配置无关，只有在调用函数时才能控制调用类型。
以下调用场景您可以自由定义函数的调用类型：
编写的应用程序调用 SCF 函数。如果您需要同步调用，请使用同步调用接口 InvokeFunction；如果您需要异步调用，请使用 Invoke 接口，并传入参数 `invokeType=Event` 。
手动调用 SCF 函数（使用 API 或 CLI）用于测试。调用时的参数区别同上。
当您使用腾讯云其他云服务作为事件源时，云服务的调用类型是预定义的：
同步调用：例如 API 网关触发器 、CLB 触发器、CKafka 触发器。
异步调用：例如 COS 触发器、定时触发器、CMQ Topic 触发器 等，详情请参见 触发器概述。

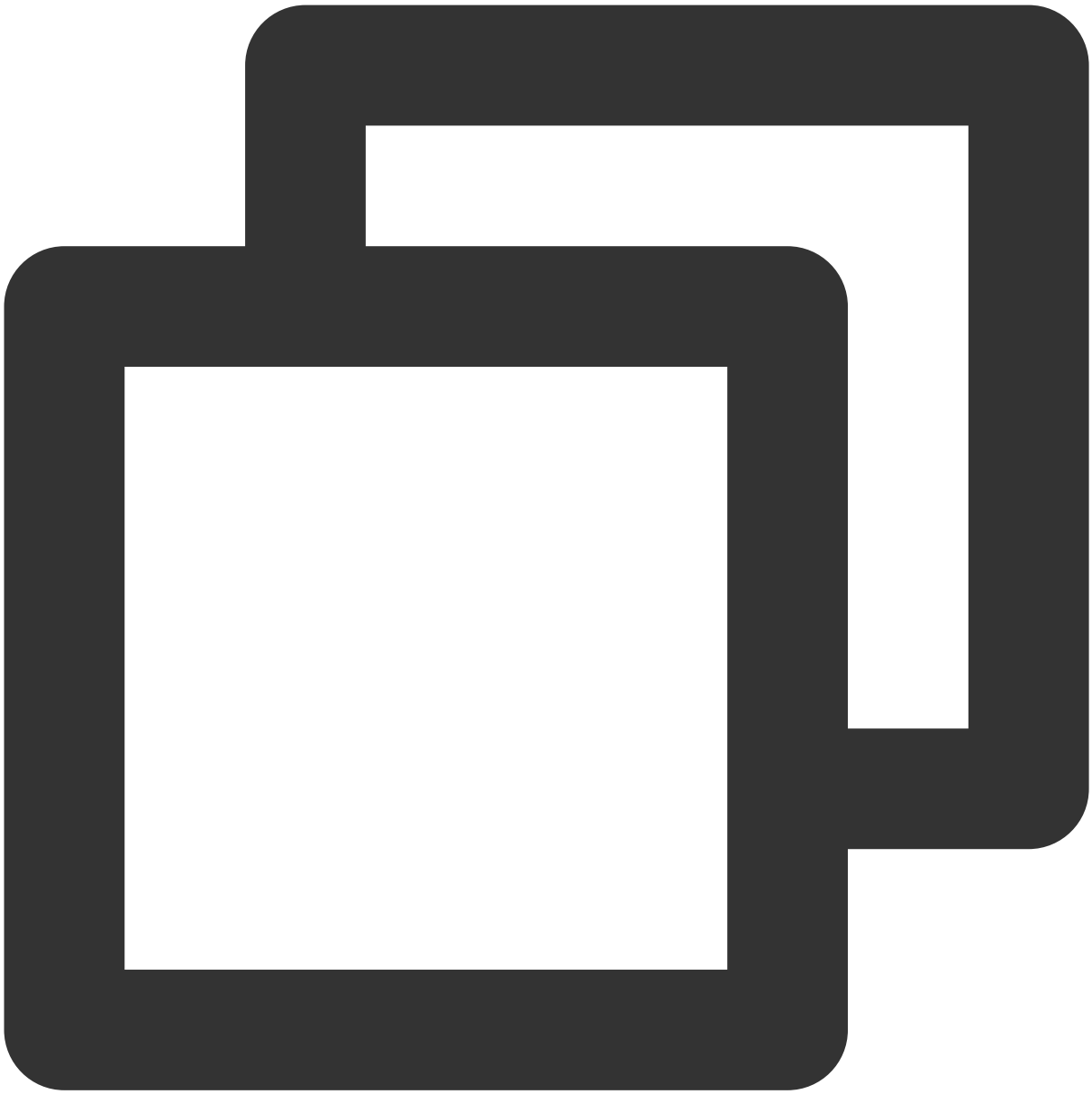
用户限制

函数相关的使用配额及相关环境限制，可参见 配额及限制。

函数并发量

函数的并发数量是指在任意指定时间对函数代码的执行数量。对于当前的 SCF 函数来说，每个发布的事件请求就会执行一次。因此，这些触发器发布的事件数（即请求量）会影响函数的并发数。您可以使用以下公式来估算并发的

函数实例总数目。



```
每秒请求量 * 函数执行时间（按秒）
```

例如，考虑一个处理 COS 事件的函数，假定函数平均用时 0.2 秒（即 200 毫秒），COS 每秒发布 300 个请求至函数。这样将同时生产 300 * 0.2 = 60 个函数实例。

并发限制

当前默认情况下，SCF 对每个函数的并发量有一定限制，您可以通过查看 并发管理 了解当前函数的并发量限制。如果调用导致函数的并发数目超过了默认限制，则该调用会被阻塞，SCF 将不会执行这次调用。根据函数的调用方

式，受限制的调用的处理方式会有所不同：
同步调用：如果函数被同步调用时受到限制，将会直接返回 432 错误码。
异步调用：如果函数被异步调用时受到限制，SCF 将以一定的策略 重试 受限制的事件。

执行环境和可用库

当前 SCF 的执行环境建立在以下基础上：
标准 CentOS 7.2
如果需要在代码中包含可执行的二进制文件、动态库或静态库，请都确保兼容此执行环境。
基于不同语言环境，在 SCF 执行环境下有相关语言的基础库及安装的附加库，您可以在各个语言说明中查看环境中已安装的附加库：
Python
Node.js
Golang
PHP
Java

部署方式

云函数（Serverless Cloud Function，SCF）提供代码部署、镜像部署两种部署方式，支持事件函数和 Web 函数两种函数类型。不同的部署方式以及函数类型在代码开发时需要采用不同的规范。本文主要介绍代码部署的事件函数的编写规范及相关概念，镜像部署 和 Web 函数 详情请参考对应文档。

SCF 事件函数

SCF 事件函数有三个基本概念：执行方法、函数入参和函数返回。
上述概念在通常的项目开发中分别对应：
执行方法：对应项目的主函数，是程序执行的起点。
函数入参：即通常理解的函数入参，但在云函数环境下，入口函数的入参为平台固定值，详情请参见 函数入参。
函数返回：对应项目中主函数的返回值，函数返回后，代码执行结束。

执行方法

SCF 平台在调用云函数时，首先会寻找执行方法作为入口，执行用户的代码。此时，用户需以**文件名.执行方法名**的形式进行设置。
例如，用户设置的执行方法为 `index.handler`，则 SCF 平台会首先寻找代码程序包中的 `index` 文件，并找到该文件中的 `handler` 方法开始执行。

在执行方法中，用户可对入口函数入参进行处理，也可任意调用代码中的其他方法。SCF 的某个函数以入口函数执行完成或函数执行异常作为执行结束。

函数入参

函数入参，是指函数在被触发调用时所传递给函数的内容。通常情况下，函数入参包括 **event** 和 **context** 两部分，但根据开发语言和环境的不同，入参个数可能有所不同，详情请参见 开发语言说明。

event 入参

context 入参

作用

event 参数类型为 dict，event 中包含了触发函数执行的基本信息，可以是平台定义的格式，也可以自定义格式。函数被触发开始执行后，可以在代码内部对 event 进行处理。

使用说明

有两种方法可以触发云函数 SCF 执行：

1. 通过调用 云 API 触发函数执行。
2. 通过绑定 触发器 触发函数执行。

SCF 两种触发方式对应两种 event 格式：

云 API 触发函数执行：

可以在调用方和函数代码之间自定义一个 dict 类型的参数。调用方按照定义好的格式传入数据，函数代码按格式获取数据。

示例：

定义一个 dict 类型的数据结构 {"key":"XXX"}，当调用方传入数据 {"key":"abctest"} 时，函数代码可以通过 event [key] 来获得值 abctest。

触发器触发函数执行：

SCF 和 API 网关、对象存储 COS、消息队列 Ckafka 等多种云服务打通，可以通过给函数绑定对应的云服务触发器触发函数执行。触发器触发函数时，event 会以一种平台预定义的、不可更改的格式作为 event 参数传给函数，您可以根据此格式编写代码并从 event 参数中获取信息。

示例：

通过对象存储 COS 触发函数时会将对象存储桶及文件的具体信息以 JSON 格式 传递给 event 参数，在函数代码中通过解析 event 信息即可完成对触发事件的处理。

作用

context 为 SCF 平台提供的入参，将 context 入参传递给执行方法，代码可通过解析 context 入参对象，获取到运行环境及当前请求的相关信息。

使用说明

SCF 提供的入参 context 包含的字段及含义如下：

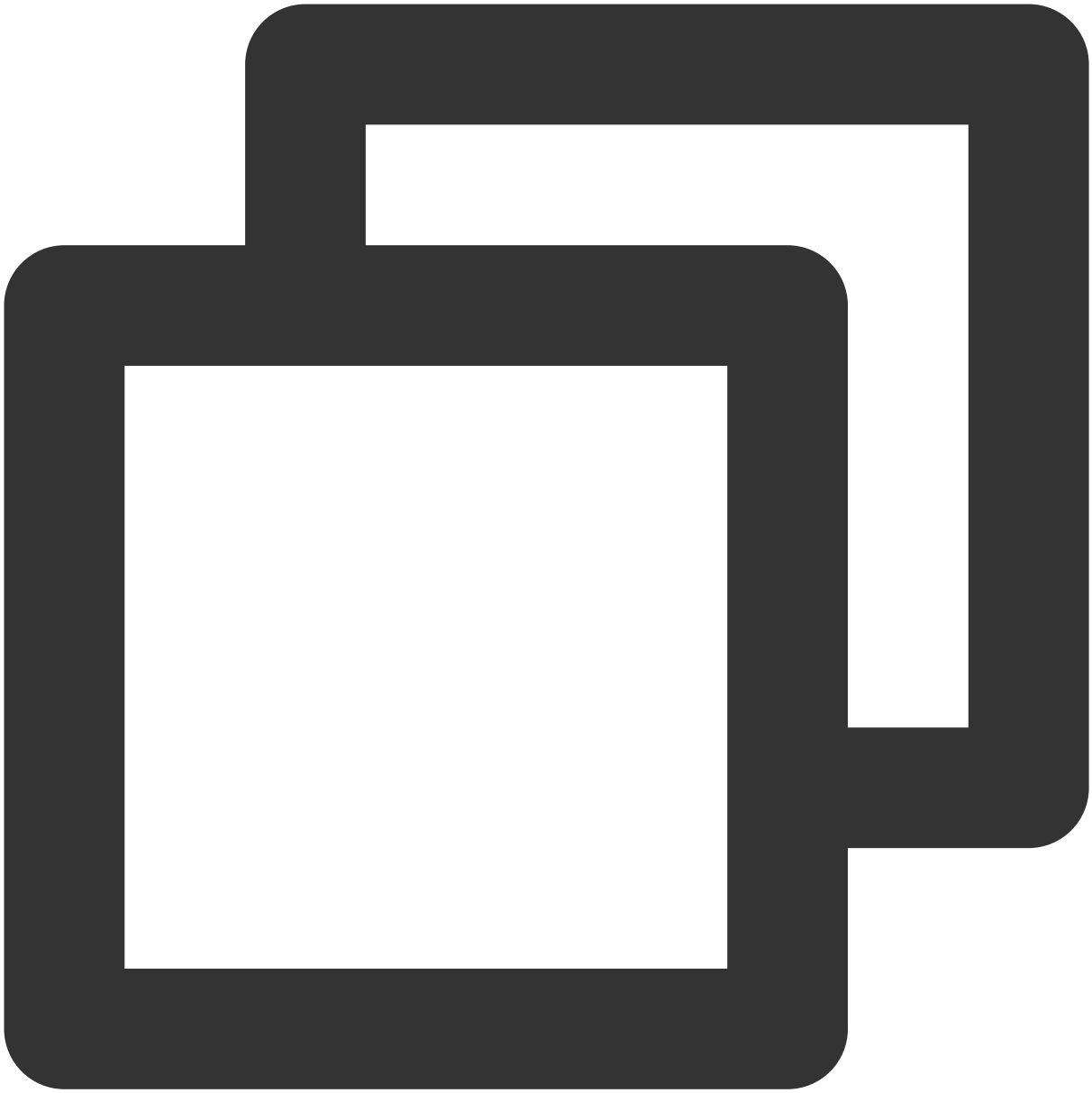
字段名称	描述
memory_limit_in_mb	函数配置内存
time_limit_in_ms	函数执行超时时间
request_id	函数执行请求 ID
environment	函数命名空间信息
environ	函数命名空间信息
function_version	函数版本信息
function_name	函数名称
namespace	函数命名空间信息
tencentcloud_region	函数所在地域
tencentcloud_appid	函数所属腾讯云账号 APPID
tencentcloud_uin	函数所属腾讯云账号 ID

注意：

为保证兼容性，context 中保留了 SCF 不同阶段对命名空间的描述方式。

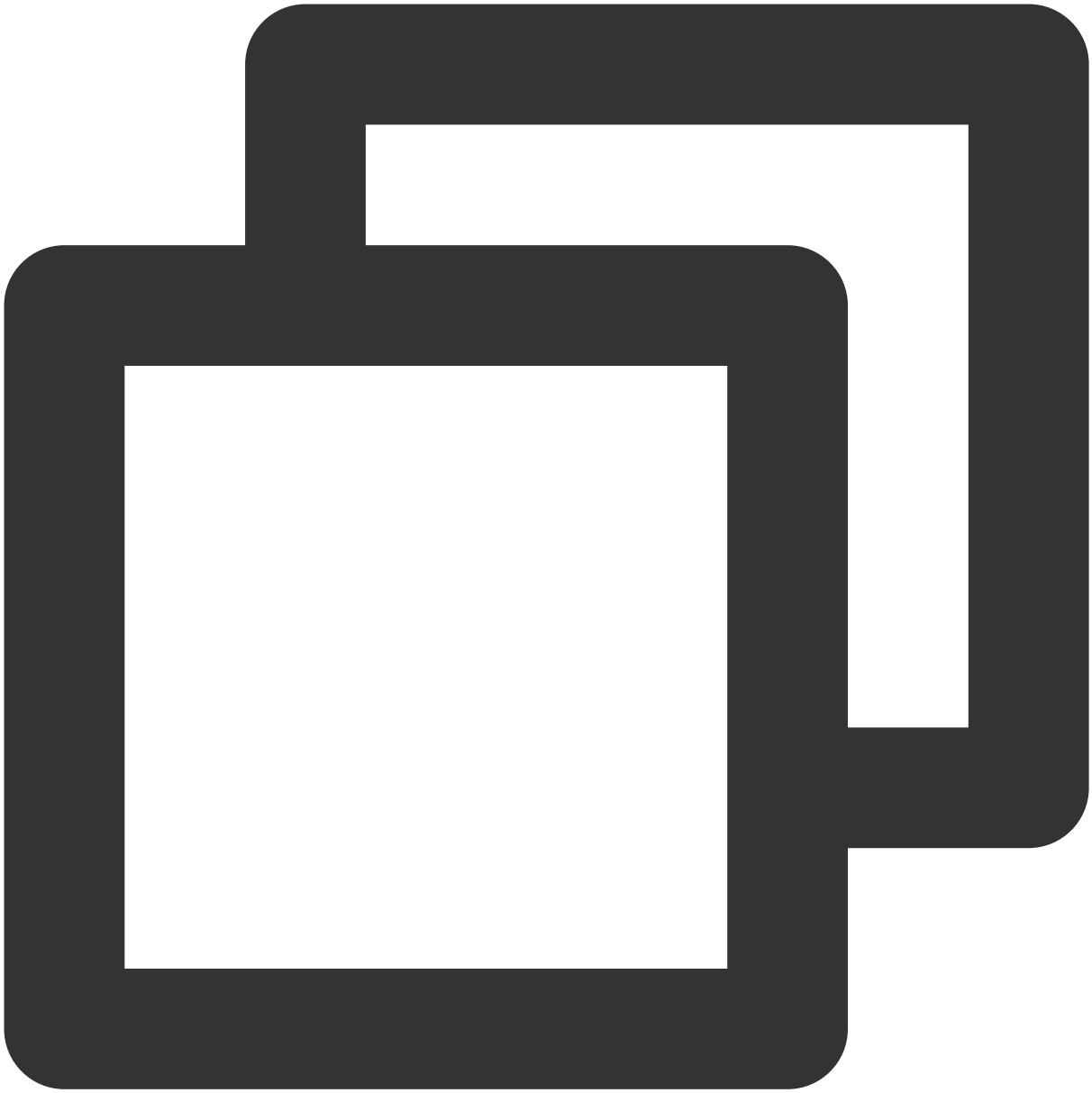
context 结构内容将会随着 SCF 平台的开发迭代而增加。

您可以在函数代码中通过标准输出语句打印 context 信息，以 `python` 运行环境为例：



```
# -*- coding: utf8 -*-
import json
def main_handler (event, context):
    print (context)
    return ("Hello World")
```

可得到以下 context 信息：



```
{"memory_limit_in_mb": 128, "time_limit_in_ms": 3000, "request_id": "f03dc946-3df4-
```

了解 event 入参和 context 入参的基本用法后，在编写函数代码时您还需注意以下几点：
为保证针对各开发语言和环境的统一性，event 入参和 context 入参均使用 JSON 数据格式统一封装。
不同触发器在触发函数时，所传递的数据结构均有所不同。详情请参见 函数触发器说明。
当云函数不需要任何输入时，您可以在代码中忽略 event 和 context 参数。

函数返回

SCF 平台会获取到云函数执行完成后的返回值，并根据下表中不同的触发方式进行处理。

触发方式	处理方式
同步触发	通过 API 网关、云 API 同步 invoke 触发函数的方式为同步触发。 使用同步方式触发的函数在执行期间，SCF 平台不会返回触发结果。 在函数执行完成后，SCF 平台会将函数返回值封装为 JSON 格式并返回给调用方。
异步触发	使用异步方式触发的云函数，SCF 平台接收触发事件后，会返回触发请求 ID 。 在函数执行完成后，函数的返回值会封装为 JSON 格式并存储在日志中。 用户可在函数执行完成后，通过返回的请求 ID 查询日志获取该异步触发函数的返回值。

当函数中的代码返回具体值时，通常返回特定的数据结构。例如 ：

运行环境	返回数据结构类型
Python	简单数据结构或 dict 数据结构
Node.js	JSON Object
PHP	Array 结构
GO	简单的数据结构或带有 JSON 描述的 struct

为保证针对各开发语言和环境的统一性，函数返回会使用**JSON 数据格式统一封装**。SCF 平台在获取到例如以上运行环境函数的返回值后，将会对返回的数据结构进行 JSON 化，并返回 JSON 内容到调用方。

注意：

需确保函数的返回值均可被 JSON 化，若直接返回对象且不具备 JSON 化方法，将会导致 SCF 平台在 JSON 化操作时失败并报错。

例如以上运行环境的返回值，无需在 return 前自行 JSON 化，否则会导致输出的字符串会进行再次 JSON 化。

异常处理

若函数在调试和运行过程中出现异常，SCF 平台会尽最大可能捕获异常并将异常信息写入日志中。函数运行产生的异常包括捕获的异常（Handled error）和未捕获的异常（Unhandled Error）。

处理方式

您可以前往 SCF 控制台 按照以下步骤进行异常处理测试：

1. 新建函数并复制以下函数代码，不添加任何触发器。
2. 单击控制台**测试**，选择 “Hello World” 测试示例进行测试。

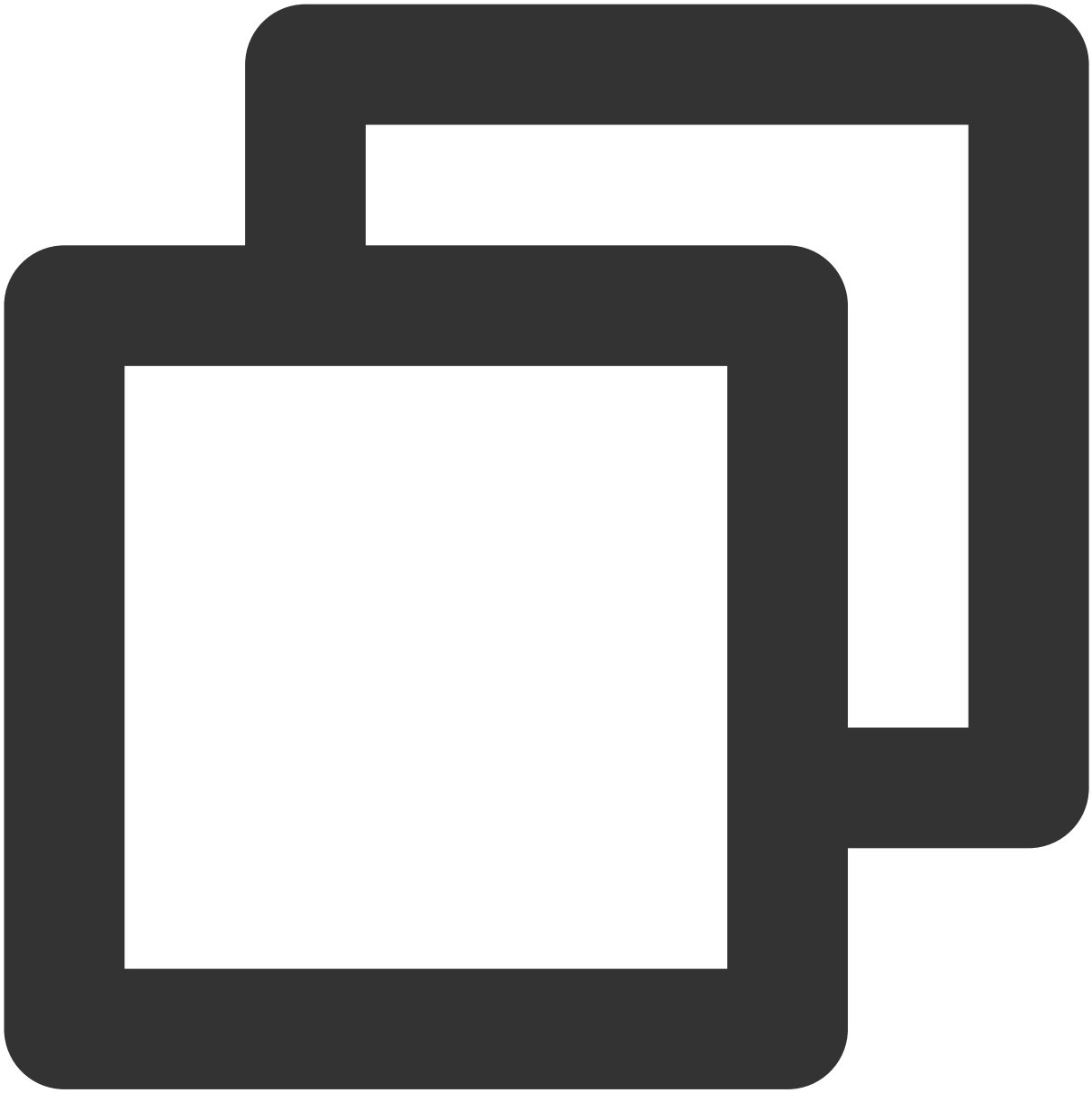
本文提供以下三种抛出异常方式，您可根据实际需求选择在代码中进行异常处理。

显式抛出异常

继承 Exception 类

使用 Try 语句捕获错误

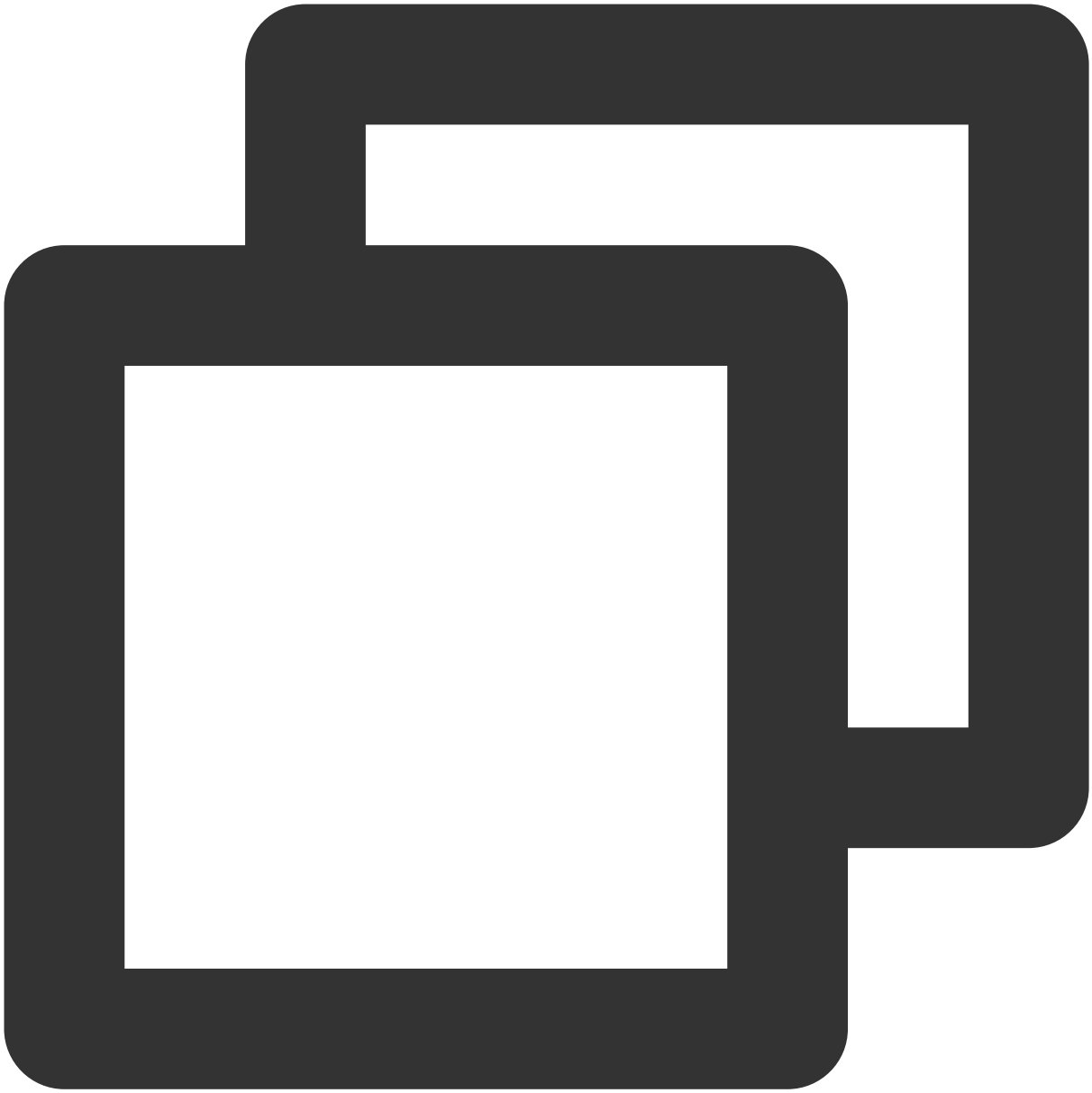
示例



```
def always_failed_handler (event,context):
    raise Exception ('I failed!')
```

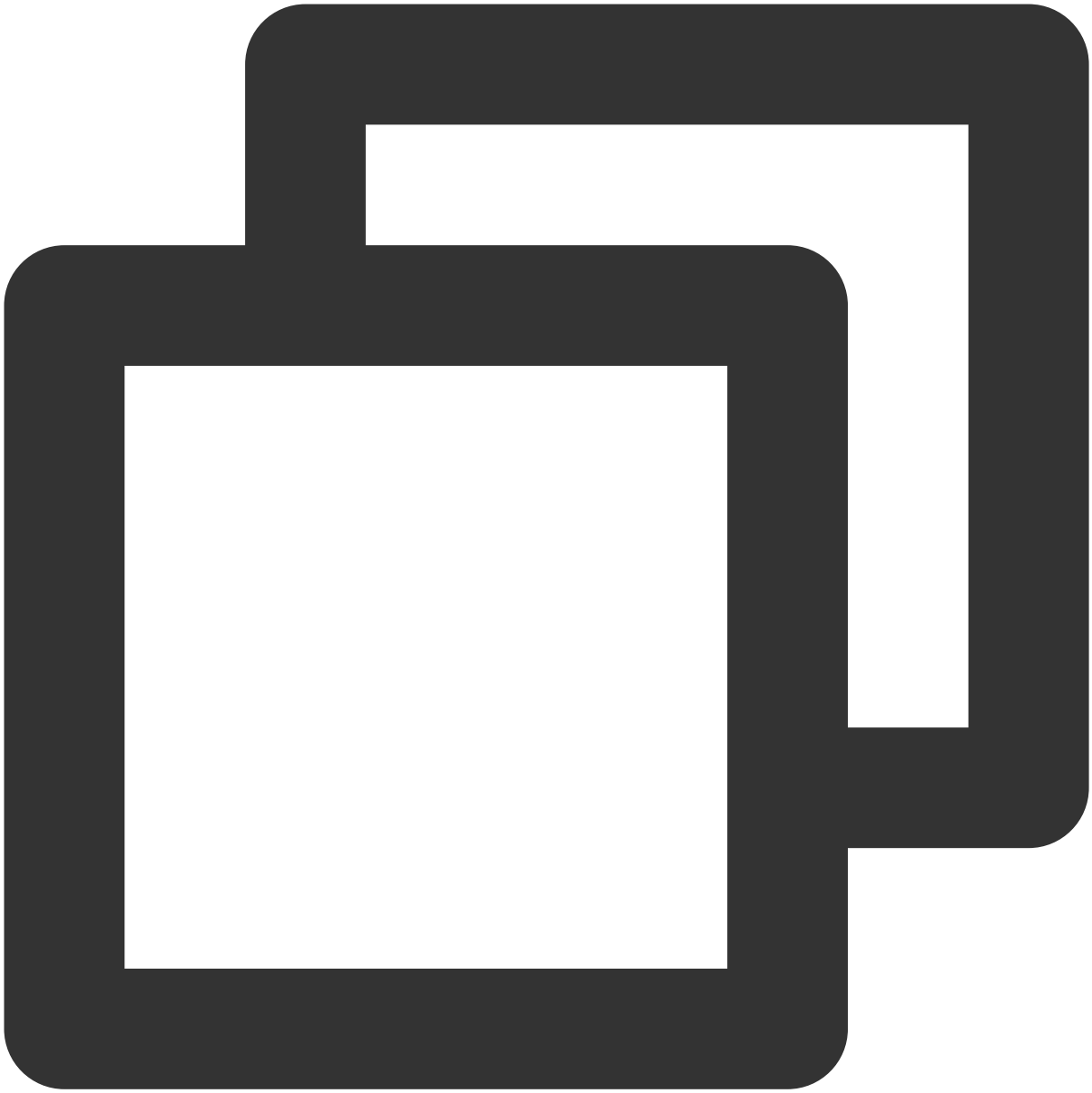
说明

此函数在运行过程中将引发异常，返回以下错误信息，SCF 平台会将此错误信息记录到函数日志中。



```
File "/var/user/index.py", line 2, in always_failed_handler
raise Exception ('I failed!')
Exception: I failed!
```

示例

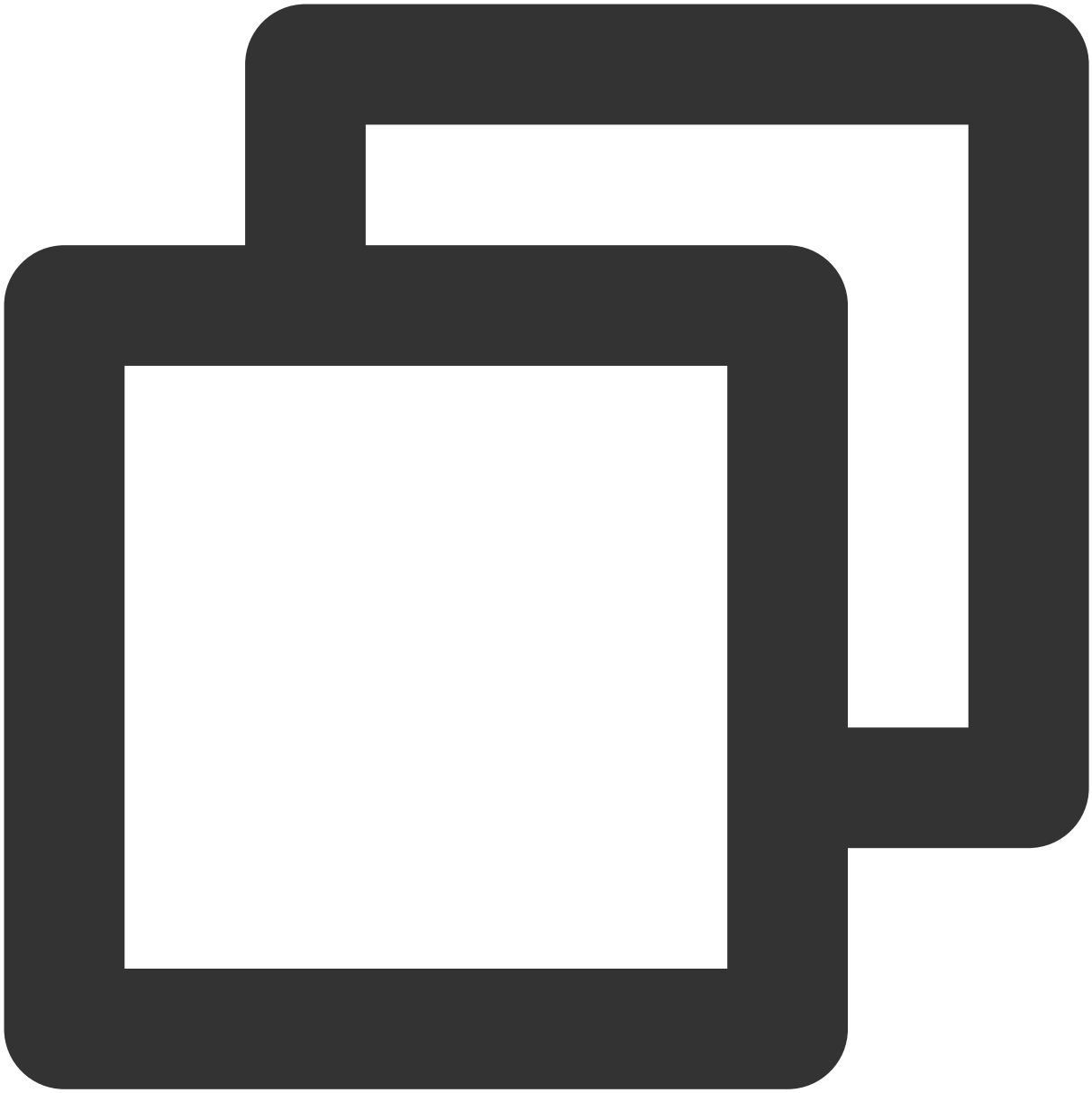


```
class UserNameAlreadyExistsException (Exception): pass
def create_user (event):
    raise UserNameAlreadyExistsException ('
        The username already exists,
        please change a name!')
```

说明

您可以在代码中自行定义错误的处理方式，保障应用程序的健壮性和可扩展型。

示例



```
def create_user (event):
    try:
        createUser (event [username],event [pwd])
    except UserNameAlreadyExistsException,e: //catch error and do something
```

说明

您可以在代码中自行定义错误的处理方式，保障应用程序的健壮性和可扩展型。

返回错误信息

当用户的代码逻辑中未进行异常处理及错误捕获时，SCF 平台会尽可能的捕获错误。例如，用户函数在运行过程中突然崩溃退出，当出现此类平台也无法捕获错误的情况时，系统将会返回一个通用的错误信息。
下表展示了代码运行中常见的一些错误：

错误场景	返回消息
使用 raise 抛出异常	{File "/var/user/index.py", line 2, in always_failed_handler raise Exception ('xxx') Exception: xxx}
处理方法不存在	{'module' object has no attribute 'xxx'}
依赖模块并不存在	{global name 'xxx' is not defined}
超时	{"time out"}

日志

SCF 平台会将函数调用的所有记录及函数代码中的全部输出存储在日志中，请使用编程语言中的打印输出语句或日志语句生成输出日志，方便调试及故障排除时使用。详情请参见 日志管理。

注意事项

由于 SCF 的特点，您须以**无状态**的风格编写您的函数代码。本地文件存储等函数生命周期内的状态特征，在函数调用结束后将随之销毁。
因此，持久状态建议存储在关系型数据库 TencentDB、对象存储 COS、云数据库 Memcached 或其他云存储服务中。

开发流程

了解更多云函数开发流程，请参见 使用流程。

产品优势

最近更新时间：2024-03-21 16:31:40

简单易用

减少组件开销

使用云函数时，用户只需编写最重要的 “核心代码”，不再需要关心负载均衡、自动伸缩、网关等周边组件，极大地降低了服务架构搭建的复杂性。

自动扩缩容

无需任何手动配置，云函数即可根据请求量自动横向扩缩。不管您的应用每天只有几个请求（如日志统计等定期事务），还是每秒有几千上万个请求（如移动应用的后端），云函数均可自动安排合理的计算资源满足业务需求。

高效又创造性地开发

加速开发

云函数不要求特定框架或依赖，开发者可以专注于核心代码的开发。同时开发人员可以组成多个小团队，单个模块的开发无需了解其他团队的代码细节。独立开发和迭代的速度变得前所未有的快，帮助用户把握住产品上线的黄金时间。

复用第三方服务

您可以使用云函数编写一些目的单一、逻辑独立的业务模块，因而可以完全复用已经成熟的第三方代码实现，例如使用 OAuth 实现登录模块。

简化运维

每个函数都是单独运行、单独部署、单独伸缩的，用户上传代码后即可自动部署，免除单体式应用部署升级难的问题。

稳定可靠

高可用部署

云函数可以自动在每个地域中随机地选择可用区来运行。如果某个可用区因灾害或电力故障等导致瘫痪，云函数会自动地选择其他可用区的基础设施来运行，免除单可用区运行的故障风险。

与其他计算服务相辅相成

常驻的工作负载可以通过云服务器 CVM ，容器服务 TKE 来承载，而由事件触发的工作负载可以使用云函数。不同云服务满足不同的业务场景和业务需求，使得您的服务架构更加健壮。

简化管理

简化安全配置

用户不再需要对 OS 入侵、登录风险、文件系统安全、网络安全和端口监听做复杂的配置和管理，一切交由平台处理，平台通过定制化的容器保证每个用户的隔离性。

可视化管理

用户可直接在控制台管理函数代码及函数何时运行（即函数触发器），无需复杂的配置文件即可一键部署和测试函数

大幅度降低开销

永远不为空闲时间付费

函数在未执行时不产生任何费用，对一些并非常驻的业务进程来说开销将大大降低。函数执行时按请求数和计算资源的运行时间收费，价格优势明显，对初创期的开发者十分友好。

应用场景

最近更新时间：2024-03-21 16:31:40

腾讯云云函数（Serverless Cloud Function , SCF）目前持续迭代发展，随着产品能力、对接产品的持续增长，云函数的适配应用场景也会越来越多。

文件处理及通知

使用对象存储 COS 作为函数触发器，在 COS Bucket 中有文件发生变更时可获得事件通知。因此针对事件，可以进行变更文件的及时处理和业务通知。
例如，在 COS Bucket 上传图片，云函数可以立刻得到通知，并可以立刻获取图片进行相应的图片剪裁、缩略、水印等操作，实现图片的自动化处理，还可以在处理完成后写入数据库，便于后续选择使用已处理好的图片。

数据 ETL 处理

一些数据处理系统中，经常需要周期性、计划性处理庞大的数据量。
例如，证券公司每12小时统计一次该时段的交易情况，并整理出该时段的交易量的前五名。例如，秒杀网站每天处理一遍交易流日志，获取因售罄导致的错误，借此分析网站的热度和趋势等。云函数近乎无限的扩容能力，可以使您轻松进行大容量数据的计算。使用云函数可以对源数据并发执行多个 mapper 和 reducer 函数，并在短时间内完成工作。相比传统的工作方式，使用云函数更能避免闲置和浪费，从而节省资金。

移动及 Web 应用

云函数可以作为移动应用及 Web 应用的后端，实现服务端应用逻辑，并通过 API 对外提供服务。通过与云缓存、云数据库、对象存储等产品的紧密结合，开发者能够构建可弹性扩展的移动或 Web 应用程序，轻松创建丰富的无服务器后端，并且这些程序可在多个数据中心高可用运行，无需在可扩展性、备份冗余方面执行任何管理工作。

AI 推理预测

在 AI 模型完成训练并开始对外提供推理服务时，可以使用无服务器云函数将数据模型包装在调用函数中。在实际用户请求到达时再运行代码，无需准备服务器，不仅可以按实际调用量计费，节省 GPU 服务器的费用，还可以获得高并发请求下的自动扩容伸缩能力。

消息转存

使用消息队列或 Ckafka 作为函数触发器，在消息队列中接收到消息时将触发云函数的运行，并会将消息作为事件内容传递给云函数。
例如，在 Ckafka 中接收到业务系统的日志时，云函数可以将日志内容作为文件写入到对象存储 COS 中，实现日志的归档存储。

业务流转

消息队列 CMQ 作为业务事件流转的中间通道，连接多个云函数，可以实现业务的状态流转及分派。云函数中的业务逻辑判断与处理，可以根据业务消息内容，进行不同的通道分派、状态流转、事件分发，实现复杂的业务流程连接。

相关产品

最近更新时间：2024-03-21 16:31:40

腾讯云云函数（Serverless Cloud Function，SCF）在使用过程中可能关联和使用到的产品如下：

产品名称	与云函数的关系
私有网络 VPC	可以通过将云函数配置到 VPC，实现访问 VPC 内的资源。
对象存储 COS	可以通过配置对象存储触发器，在对应的 Bucket 产生事件时触发云函数。
日志服务 CLS	可以通过配置对接日志服务，将云函数的运行日志写入日志服务中。
消息队列 CMQ	可以通过配置消息队列触发器，在对应的队列收到消息时触发云函数。
消息队列 Ckafka	可以通过配置 Ckafka 触发器，在对应的 Kafka topic 收到消息时触发云函数。
API 网关 API Gateway	可以通过配置 API 网关触发器，在 API URL 上接收 HTTP 请求时触发云函数。
访问管理 CAM	可以通过配置访问管理的角色，在云函数授予代码执行时访问授权资源的权限。