高性能计算平台

API 文档

腾讯云

文档目录

API 文档
产品版本

最近更新时间：2023-03-30 02:30:01

您正在浏览thpc产品文档的版本: 2023-03-21

最新版本

- 2023-03-21 (当前版本)

以往版本

- 2021-11-09
- 2022-04-01

更新历史

最近更新时间：2025-05-15 02:06:49

第 24 次发布

发布时间：2025-05-15 02:06:36

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- ModifyWorkspacesRenewFlag

第 23 次发布

发布时间：2025-03-28 02:07:07

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- ClusterOverview
  - 新增成员：ClusterType

第 22 次发布

发布时间：2025-03-26 02:12:01

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeClusters
  - 新增入参：Filters

第 21 次发布

发布时间：2025-03-25 02:09:57

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- NodeOverview
  - 新增成员：NodeAllocateState
- QueueOverview
  - **修改成员：** QueueName

第 20 次发布

发布时间：2025-03-24 02:09:58

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- GooseFSxOptionOverview
  - **修改成员：** Masters, LocalPath

第 19 次发布

发布时间：2025-02-14 02:03:55

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- QueueConfig
  - 新增成员：DesiredNodeCount
- QueueConfigOverview
  - 新增成员：DesiredNodeCount

第 18 次发布

发布时间：2024-09-05 02:14:49

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeWorkspaces
- ModifyWorkspacesAttribute
- TerminateWorkspaces

新增数据结构：

- SpaceInfo

第 17 次发布

发布时间：2024-09-04 02:14:20

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateWorkspaces

新增数据结构：

- SpaceChargePrepaid
- SpaceDataDisk
- SpaceInternetAccessible
- SpacePlacement
- SpaceSystemDisk
- SpaceVirtualPrivateCloud
- TagSpecification

第 16 次发布

发布时间：2024-07-24 02:16:49

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AttachNodes
- DetachNodes

修改接口：

- AddNodes
  - 新增入参：ResourceType

修改数据结构：

- ComputeNode
  - 新增成员：ResourceType
- NodeOverview
  - 新增成员：NodeId
  - **修改成员：** InstanceId, Zone, NodeState, ImageId, QueueName, NodeRole, NodeType

第 15 次发布

发布时间：2024-07-12 01:26:38

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- LoginSettings
  - 新增成员：KeyIds

第 14 次发布

发布时间：2024-06-27 01:21:12

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateCluster
  - 新增入参：SchedulerVersion

修改数据结构：

- ClusterOverview
  - 新增成员：SchedulerVersion

第 13 次发布

发布时间：2023-11-06 00:21:33

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- ManagerNode
  - 新增成员：EnhancedService

第 12 次发布

发布时间：2023-09-21 06:17:00

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- EnhancedService
- RunAutomationServiceEnabled
- RunMonitorServiceEnabled

- RunSecurityServiceEnabled

修改数据结构：

- QueueConfig
  - 新增成员：EnhancedService

第 11 次发布

发布时间：2023-09-14 02:31:21

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateCluster
  - 新增入参：HpcClusterId

第 10 次发布

发布时间：2023-09-07 02:37:25

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- QueueConfig
  - 新增成员：ScaleUpMemRatio
- QueueConfigOverview
  - 新增成员：ScaleUpMemRatio

第 9 次发布

发布时间：2023-09-06 02:32:29

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- QueueConfigOverview
  - **修改成员：** QueueName, MinSize, MaxSize, EnableAutoExpansion, EnableAutoShrink, ExpansionNodeConfigs, DesiredIdleNodeCapacity, ScaleOutRatio, ScaleOutNodeThreshold, MaxNodesPerCycle

第 8 次发布

发布时间：2023-06-15 01:38:28

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- CFSOption
  - 新增成员：MountOption
- CFSOptionOverview
  - 新增成员：MountOption

第 7 次发布

发布时间：2023-06-12 01:40:28

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- QueueConfig
  - 新增成员：MaxNodesPerCycle
- QueueConfigOverview
  - 新增成员：MaxNodesPerCycle

第 6 次发布

发布时间：2023-05-31 01:45:41

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- AddNodes
  - 新增入参：ProjectId

修改数据结构：

- ComputeNode
  - 新增成员：ProjectId
- ExpansionNodeConfig
  - 新增成员：ProjectId
- LoginNode
  - 新增成员：ProjectId
- ManagerNode
  - 新增成员：ProjectId

第 5 次发布

发布时间：2023-05-29 15:05:43

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- ClusterOverview
  - 新增成员：AutoScalingType

第 4 次发布

发布时间：2023-05-19 02:07:25

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- QueueConfig
  - 新增成员：ScaleOutRatio, ScaleOutNodeThreshold
- QueueConfigOverview
  - 新增成员：ScaleOutRatio, ScaleOutNodeThreshold

第 3 次发布

发布时间：2023-05-09 01:46:09

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeInitNodeScripts

第 2 次发布

发布时间：2023-05-01 01:41:10

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- ModifyInitNodeScripts

修改接口：

- CreateCluster
  - 新增入参：InitNodeScripts

新增数据结构：

- NodeScript

修改数据结构：

- QueueConfig
  - 新增成员：DesiredIdleNodeCapacity
- QueueConfigOverview
  - 新增成员：DesiredIdleNodeCapacity

第 1 次发布

发布时间：2023-03-29 12:01:10

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AddClusterStorageOption
- AddNodes
- AddQueue
- CreateCluster
- DeleteCluster
- DeleteClusterStorageOption
- DeleteNodes
- DeleteQueue
- DescribeAutoScalingConfiguration
- DescribeClusterActivities
- DescribeClusterStorageOption
- DescribeClusters
- DescribeNodes
- DescribeQueues
- SetAutoScalingConfiguration

新增数据结构：

- CFSOption
- CFSOptionOverview
- ClusterActivity
- ClusterOverview
- ComputeNode

- ComputeNodeOverview
- DataDisk
- ExpansionNodeConfig
- ExpansionNodeConfigOverview
- Filter
- GooseFSOption
- GooseFSOptionOverview
- GooseFSxOption
- GooseFSxOptionOverview
- InstanceChargePrepaid
- InternetAccessible
- LoginNode
- LoginNodeOverview
- LoginSettings
- ManagerNode
- ManagerNodeOverview
- NodeActivity
- NodeOverview
- Placement
- QueueConfig
- QueueConfigOverview
- QueueOverview
- StorageOption
- StorageOptionOverview
- SystemDisk
- Tag
- VirtualPrivateCloud

简介

最近更新时间：2025-04-25 02:01:40

高性能计算平台（TencentCloud High Performance Computing，THPC）是一款腾讯云自研的高性能计算资源管理服务，集成腾讯云上的计算、存储、网络等产品资源，并整合 HPC 专用作业管理调度、集群管理等软件，向用户提供弹性灵活、性能卓越、自助化的计算服务。可以帮助您高效地管理云上高性能计算资源，实现弹性使用云上高性能计算资源的需求。

API 概览

最近更新时间：2025-05-15 02:06:48

节点相关接口

接口名称	接口功能	频率限制（次/秒）
AddNodes	添加节点	20
AttachNodes	绑定计算资源到集群	20
DeleteNodes	删除节点	20
DescribeInitNodeScripts	查询节点初始化脚本列表	20
DescribeNodes	查询指定集群节点列表	20
DetachNodes	从集群解绑节点	20
ModifyInitNodeScripts	修改节点初始化脚本	20

存储选项相关接口

接口名称	接口功能	频率限制（次/秒）
AddClusterStorageOption	添加集群存储选项	20
DeleteClusterStorageOption	删除集群存储选项	20
DescribeClusterStorageOption	查询集群存储选项	20

集群相关接口

接口名称	接口功能	频率限制（次/秒）
CreateCluster	创建集群	20
DeleteCluster	删除集群	20
DescribeClusterActivities	查询集群活动历史记录	20
DescribeClusters	查询集群列表	20

弹性伸缩相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeAutoScalingConfiguration	查询弹性伸缩配置信息	20
SetAutoScalingConfiguration	设置弹性伸缩配置信息	20

队列相关接口

接口名称	接口功能	频率限制（次/秒）
AddQueue	添加队列	20
DeleteQueue	删除队列	20
DescribeQueues	查询队列列表	20

工作空间相关接口

接口名称	接口功能	频率限制（次/秒）
CreateWorkspaces	创建工作空间	20
DescribeWorkspaces	查询工作空间列表	20
ModifyWorkspacesAttribute	修改工作空间的属性	20
ModifyWorkspacesRenewFlag	修改工作空间的续费标识	20
TerminateWorkspaces	销毁工作空间	20

> **注意：**
> 以上给出的接口频率限制维度为 API + 接入地域 + 子账号，有关限频更多说明参考：API 频率限制说明

调用方式
请求结构

最近更新时间：2025-05-21 01:32:56

1. 服务地址

API 支持就近地域接入，本产品就近地域接入域名为 thpc.tencentcloudapi.com ，也支持指定地域域名访问，例如广州地域的域名为 thpc.ap-guangzhou.tencentcloudapi.com 。

推荐使用就近地域接入域名。根据调用接口时客户端所在位置，会自动解析到**最近的**某个具体地域的服务器。例如在广州发起请求，会自动解析到广州的服务器，效果和指定 thpc.ap-guangzhou.tencentcloudapi.com 是一致的。

注意：对时延敏感的业务，建议指定带地域的域名。

注意：域名是 API 的接入点，并不代表产品或者接口实际提供服务的地域。产品支持的地域列表请在调用方式/公共参数文档中查阅，接口支持的地域请在接口文档输入参数中查阅。

目前支持的域名列表为：

接入地域	域名
就近地域接入（推荐，只支持非金融区）	thpc.tencentcloudapi.com
华南地区（广州）	thpc.ap-guangzhou.tencentcloudapi.com
华东地区（上海）	thpc.ap-shanghai.tencentcloudapi.com
华东地区（南京）	thpc.ap-nanjing.tencentcloudapi.com
华北地区（北京）	thpc.ap-beijing.tencentcloudapi.com
西南地区（成都）	thpc.ap-chengdu.tencentcloudapi.com
西南地区（重庆）	thpc.ap-chongqing.tencentcloudapi.com
港澳台地区（中国香港）	thpc.ap-hongkong.tencentcloudapi.com
亚太东南（新加坡）	thpc.ap-singapore.tencentcloudapi.com
亚太东南（雅加达）	thpc.ap-jakarta.tencentcloudapi.com
亚太东南（曼谷）	thpc.ap-bangkok.tencentcloudapi.com
亚太东北（首尔）	thpc.ap-seoul.tencentcloudapi.com
亚太东北（东京）	thpc.ap-tokyo.tencentcloudapi.com
美国东部（弗吉尼亚）	thpc.na-ashburn.tencentcloudapi.com
美国西部（硅谷）	thpc.na-siliconvalley.tencentcloudapi.com
南美地区（圣保罗）	thpc.sa-saopaulo.tencentcloudapi.com
欧洲地区（法兰克福）	thpc.eu-frankfurt.tencentcloudapi.com

注意：由于金融区和非金融区是隔离不互通的，因此当访问金融区服务时（公共参数 Region 为金融区地域），需要同时指定带金融区地域的域名，最好和 Region 的地域保持一致。

金融区接入地域	金融区域名
华东地区（上海金融）	thpc.ap-shanghai-fsi.tencentcloudapi.com
华南地区（深圳金融）	thpc.ap-shenzhen-fsi.tencentcloudapi.com

2. 通信协议

腾讯云 API 的所有接口均通过 HTTPS 进行通信，提供高安全性的通信通道。

3. 请求方法

支持的 HTTP 请求方法:

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型:

- application/json（推荐），必须使用签名方法 v3（TC3-HMAC-SHA256）。
- application/x-www-form-urlencoded，必须使用签名方法 v1（HmacSHA1 或 HmacSHA256）。
- multipart/form-data（仅部分接口支持），必须使用签名方法 v3（TC3-HMAC-SHA256）。

GET 请求的请求包大小不得超过32KB。POST 请求使用签名方法 v1（HmacSHA1、HmacSHA256）时不得超过1MB。POST 请求使用签名方法 v3（TC3-HMAC-SHA256）时支持10MB。

4. 字符编码

均使用 `UTF-8` 编码。

公共参数

最近更新时间：2024-11-15 02:07:38

公共参数是用于标识用户和接口签名的参数，如非必要，在每个接口单独的文档中不再对这些参数进行说明，但每次请求均需要携带这些参数，才能正常发起请求。

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

签名方法 v3（有时也称作 TC3-HMAC-SHA256）相比签名方法 v1（有些文档可能会简称签名方法），更安全，支持更大的请求包，支持 POST JSON 格式，性能有一定提升，推荐使用该签名方法计算签名。完整介绍详见 签名方法 v3。

注意：出于简化的目的，部分接口文档中的示例使用的是签名方法 v1 GET 请求，而不是更安全的签名方法 v3。

使用签名方法 v3 时，公共参数需要统一放到 HTTP Header 请求头部中，如下表所示：

参数名称	类型	必选	描述
Action	String	是	HTTP 请求头：X-TC-Action。操作的接口名称。取值参考接口文档输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	HTTP 请求头：X-TC-Region。地域参数，用来标识希望操作哪个地域的数据。取值参考接口文档中输入参数章节关于公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	HTTP 请求头：X-TC-Timestamp。当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。**注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。**
Version	String	是	HTTP 请求头：X-TC-Version。操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKID***/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKID*** 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为具体产品名，通常为域名前缀。例如，域名 cvm.tencentcloudapi.com 意味着产品名是 cvm。本产品取值为 thpc；tc3_request 为固定字符串； - SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部； - Signature：签名摘要，计算过程详见 文档。
Token	String	否	HTTP 请求头：X-TC-Token。即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	HTTP 请求头：X-TC-Language。指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

假设用户想要查询广州地域的云服务器实例列表中的前十个，接口参数设置为偏移量 Offset=0，返回数量 Limit=10，则其请求结构按照请求 URL、请求头部、请求体示例如下：

HTTP GET 请求结构示例：

```
https://cvm.tencentcloudapi.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: ap-guangzhou
```

HTTP POST（application/json）请求结构示例：

```
https://cvm.tencentcloudapi.com/

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: application/json
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

{"Offset":0,"Limit":10}
```

HTTP POST（multipart/form-data）请求结构示例（仅特定的接口支持）：

```
https://cvm.tencentcloudapi.com/

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: multipart/form-data; boundary=58731222010402
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

--58731222010402
Content-Disposition: form-data; name="Offset"

0
--58731222010402
Content-Disposition: form-data; name="Limit"

10
--58731222010402--
```

使用签名方法 v1 的公共参数

使用签名方法 v1（有时会称作 HmacSHA256 和 HmacSHA1），公共参数需要统一放到请求串中，完整介绍详见文档

参数名称	类型	必选	描述
Action	String	是	操作的接口名称。取值参考接口文档中输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。
Nonce	Integer	是	随机正整数，与 Timestamp 联合起来，用于防止重放攻击。
SecretId	String	是	在 云API密钥 上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey ，而 SecretKey 会用来生成请求签名 Signature。
Signature	String	是	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见 文档。

参数名称	类型	必选	描述
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

假设用户想要查询广州地域的云服务器实例列表，其请求结构按照请求 URL、请求头部、请求体示例如下：

HTTP GET 请求结构示例：

```
https://cvm.tencentcloudapi.com/?Action=DescribeInstances&Version=2017-03-12&SignatureMethod=HmacSHA256&Timestamp=1527672334&Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*******************************

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

```
https://cvm.tencentcloudapi.com/

Host: cvm.tencentcloudapi.com
Content-Type: application/x-www-form-urlencoded

Action=DescribeInstances&Version=2017-03-12&SignatureMethod=HmacSHA256&Timestamp=1527672334&Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*****************************
```

地域列表

本产品所有接口 Region 字段的可选值如下表所示。如果接口不支持该表中的所有地域，则会在接口文档中单独说明。

地域	取值
华北地区（北京）	ap-beijing
西南地区（成都）	ap-chengdu
西南地区（重庆）	ap-chongqing
华南地区（广州）	ap-guangzhou
华东地区（南京）	ap-nanjing
华东地区（上海）	ap-shanghai

签名方法 v3

最近更新时间：2024-12-25 02:03:52

以下文档说明了签名方法 v3 的签名过程，但仅在您编写自己的代码来调用腾讯云 API 时才有用。我们推荐您使用 腾讯云 API Explorer ，腾讯云 SDK 和 腾讯云命令行工具（TCCLI）等开发者工具，从而无需学习如何对 API 请求进行签名。

推荐使用 API Explorer　　点击调试

您可以通过 API Explorer 的【签名串生成】模块查看每个接口签名的生成过程。

腾讯云 API 会对每个请求进行身份验证，用户需要使用安全凭证，经过特定的步骤对请求进行签名（Signature），每个请求都需要在公共参数中指定该签名结果并以指定的方式和格式发送请求。

为什么要进行签名

签名通过以下方式帮助保护请求：

1. 验证请求者的身份
   签名确保请求是由持有有效访问密钥的人发送的。请参阅控制台 云 API 密钥 页面获取密钥相关信息。

2. 保护传输中的数据
   为了防止请求在传输过程中被篡改，腾讯云 API 会使用请求参数来计算请求的哈希值，并将生成的哈希值加密后作为请求的一部分，发送到腾讯云 API 服务器。服务器会使用收到的请求参数以同样的过程计算哈希值，并验证请求中的哈希值。如果请求被篡改，将导致哈希值不一致，腾讯云 API 将拒绝本次请求。

签名方法 v3（TC3-HMAC-SHA256）功能上覆盖了以前的签名方法 v1，而且更安全，支持更大的请求，支持 JSON 格式，POST 请求支持传空数组和空字符串，性能有一定提升，推荐使用该签名方法计算签名。

首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v3”，可以对生成签名过程进行验证，也可直接生成 SDK 代码。推荐使用腾讯云 API 配套的 8 种常见的编程语言 SDK，已经封装了签名和请求过程，均已开源，支持 Python、Java、PHP、Go、NodeJS、.NET、C++、Ruby。

申请安全凭证

本文使用的安全凭证为密钥，密钥包括 SecretId 和 SecretKey。每个用户最多可以拥有两对密钥。

- SecretId：用于标识 API 调用者身份，可以简单类比为用户名。
- SecretKey：用于验证 API 调用者的身份，可以简单类比为密码。
- **用户必须严格保管安全凭证，避免泄露，否则将危及财产安全。如已泄露，请立刻禁用该安全凭证。**

申请安全凭证的具体步骤如下：

1. 登录 腾讯云管理中心控制台 。
2. 前往 云API密钥 的控制台页面。
3. 在 云API密钥 页面，单击【新建密钥】创建一对密钥。

签名版本 v3 签名过程

云 API 支持 GET 和 POST 请求。对于GET方法，只支持 `Content-Type: application/x-www-form-urlencoded` 协议格式。对于POST方法，目前支持 `Content-Type: application/json` 以及 `Content-Type: multipart/form-data` 两种协议格式，json 格式绝大多数接口均支持，multipart 格式只有特定接口支持，此时该接口不能使用 json 格式调用，参考具体业务接口文档说明。推荐使用 POST 请求，因为两者的结果并无差异，但 GET 请求只支持 32 KB 以内的请求包。

下面以云服务器查询广州区实例列表作为例子，分步骤介绍签名的计算过程。我们选择该接口是因为：

1. 云服务器默认已开通，该接口很常用；
2. 该接口是只读的，不会改变现有资源的状态；
3. 接口覆盖的参数种类较全，可以演示包含数据结构的数组如何使用。

在示例中，不论公共参数或者接口的参数，我们尽量选择容易犯错的情况。在实际调用接口时，请根据实际情况来，每个接口的参数并不相同，不要照抄这个例子的参数和值。此外，这里只展示了部分公共参数和接口输入参数，用户可以根据实际需要添加其他参数，例如 Language 和 Token 公共参数（在 HTTP 头部设置，添加 X-TC-前缀）。

假设用户的 SecretId 和 SecretKey 分别是：AKID******************************** 和 ********************************。用户想查看广州区云服务器名为“未命名”的主机状态，只返回一条数据。则请求可能为：

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f" \
-H "Content-Type: application/json; charset=utf-8" \
-H "Host: cvm.tencentcloudapi.com" \
-H "X-TC-Action: DescribeInstances" \
-H "X-TC-Timestamp: 1551113065" \
-H "X-TC-Version: 2017-03-12" \
-H "X-TC-Region: ap-guangzhou" \
-d '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
```

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

```
CanonicalRequest =
HTTPRequestMethod + '\n' +
CanonicalURI + '\n' +
CanonicalQueryString + '\n' +
CanonicalHeaders + '\n' +
SignedHeaders + '\n' +
HashedRequestPayload
```

字段名称	解释
HTTPRequestMethod	HTTP 请求方法（GET、POST）。此示例取值为 POST。
CanonicalURI	URI 参数，API 3.0 固定为正斜杠（/）。
CanonicalQueryString	发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串""，对于 GET 请求，则为 URL 中问号（?）后面的字符串内容，例如：Limit=10&Offset=0。 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码（特殊字符编码后需大写字母），字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入其他头部参与签名以提高自身请求的唯一性和安全性，此示例额外增加了接口名头部。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接； 2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。 此示例计算结果是 content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n。 注意：content-type 必须和实际发送的相符合，有些编程语言网络库即使未指定也会自动添加 charset 值，如果签名时和发送时不一致，服务器会返回签名校验失败。
SignedHeaders	参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。 拼接规则： 1. 头部 key 统一转成小写； 2. 多个头部 key（小写）按照 ASCII 升序进行拼接，并且以分号（;）分隔。 此示例为 content-type;host;x-tc-action
HashedRequestPayload	请求正文（payload，即 body，此示例为 {"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}）的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，即对 HTTP 请求正文做 SHA256 哈希，然后十六进制编

字段名称	解释
	码，最后编码串转换成小写字母。对于 GET 请求，RequestPayload 固定为空字符串。此示例计算结果是 35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064。

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com
x-tc-action:describeinstances

content-type;host;x-tc-action
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

```
StringToSign =
Algorithm + "\n" +
RequestTimestamp + "\n" +
CredentialScope + "\n" +
HashedCanonicalRequest
```

字段名称	解释
Algorithm	签名算法，目前固定为 TC3-HMAC-SHA256。
RequestTimestamp	请求时间戳，即请求头部的公共参数 X-TC-Timestamp 取值，取当前时间 UNIX 时间戳，精确到秒。此示例取值为 1551113065。
CredentialScope	凭证范围，格式为 Date/service/tc3_request，包含日期、所请求的服务和终止字符串（tc3_request）。**Date 为 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致**；service 为产品名，必须与调用的产品域名一致。此示例计算结果是 2019-02-25/cvm/tc3_request。
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。此示例计算结果是 7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84。

注意：

1. **Date 必须从时间戳 X-TC-Timestamp 计算得到，且时区为 UTC+0。如果加入系统本地时区信息，例如东八区，将导致白天和晚上调用成功，但是凌晨时调用必定失败。假设时间戳为 1551113065，在东八区的时间是 2019-02-26 00:44:25，但是计算得到的 Date 取 UTC+0 的日期应为 2019-02-25，而不是 2019-02-26。**
2. **Timestamp 必须是当前系统时间，且需确保系统时间和标准时间是同步的，如果相差超过五分钟则必定失败。如果长时间不和标准时间同步，可能运行一段时间后，请求失败，返回签名过期错误。**

根据以上规则，示例中得到的待签名字符串如下：

```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84
```

3. 计算签名

1）计算派生签名密钥，伪代码如下：

```
SecretKey = "********************************"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

派生出的密钥 `SecretDate`、`SecretService` 和 `SecretSigning` 是二进制的数据，可能包含不可打印字符，将其转为十六进制字符串打印的输出分别为：
da98fb70dcf6b112dc21038d1eeeb3a95c74b4dcb12c1131f864f6066bd02be0，
8d70cbefb03939f929db64d32dc2ba89b1095620119fe3e050e2b18c5bd2752f，
b596b923aad85185e2d1f6659d2a062e0a86731226e021e61bfe06f7ed05f5af。

请注意，不同的编程语言，HMAC 库函数中参数顺序可能不一样，请以实际情况为准。此处的伪代码密钥参数 key 在前，消息参数 data 在后。通常标准库函数会提供二进制格式的返回值，也可能会提供打印友好的十六进制格式的返回值，此处使用的是二进制格式。

字段名称	解释
SecretKey	原始的 SecretKey，即 `********************************`。
Date	即 Credential 中的 Date 字段信息。此示例取值为 `2019-02-25`。
Service	即 Credential 中的 Service 字段信息。此示例取值为 `cvm`。

2）计算签名，伪代码如下：

```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

此示例计算结果是 `10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f`。

4. 拼接 Authorization

按如下格式拼接 Authorization：

```
Authorization =
Algorithm + ' ' +
'Credential=' + SecretId + '/' + CredentialScope + ', ' +
'SignedHeaders=' + SignedHeaders + ', ' +
'Signature=' + Signature
```

字段名称	解释
Algorithm	签名方法，固定为 `TC3-HMAC-SHA256`。
SecretId	密钥对中的 SecretId，即 `AKID********************************`。
CredentialScope	见上文，凭证范围。此示例计算结果是 `2019-02-25/cvm/tc3_request`。
SignedHeaders	见上文，参与签名的头部信息。此示例取值为 `content-type;host;x-tc-action`。
Signature	签名值。此示例计算结果是 `10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f`。

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
```

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}
```

> ⚠ 注意：
>
> 请求发送时的 HTTP 头部（Header）和请求体（Payload）必须和签名计算过程中的内容完全一致，否则会返回签名不一致错误。可以通过打印实际请求内容，网络抓包等方式对比排查。

签名演示

在实际调用 API 3.0 时，推荐使用配套的腾讯云 SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

下面提供了不同产品的生成签名 demo，您可以找到对应的产品参考签名的生成：

- Signature Demo

为了更清楚地解释签名过程，下面以实际编程语言为例，将上述的签名过程完整实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK。

Java

```java
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TencentCloudAPITC3Demo {
private final static Charset UTF8 = StandardCharsets.UTF_8;
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
private final static String SECRET_ID = System.getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
```

```
private final static String SECRET_KEY = System.getenv("TENCENTCLOUD_SECRET_KEY");
private final static String CT_JSON = "application/json; charset=utf-8";

public static byte[] hmac256(byte[] key, String msg) throws Exception {
Mac mac = Mac.getInstance("HmacSHA256");
SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
mac.init(secretKeySpec);
return mac.doFinal(msg.getBytes(UTF8));
}

public static String sha256Hex(String s) throws Exception {
MessageDigest md = MessageDigest.getInstance("SHA-256");
byte[] d = md.digest(s.getBytes(UTF8));
return DatatypeConverter.printHexBinary(d).toLowerCase();
}

public static void main(String[] args) throws Exception {
String service = "cvm";
String host = "cvm.tencentcloudapi.com";
String region = "ap-guangzhou";
String action = "DescribeInstances";
String version = "2017-03-12";
String algorithm = "TC3-HMAC-SHA256";
String timestamp = "1551113065";
//String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
// 注意时区，否则容易出错
sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

// ************ 步骤 1: 拼接规范请求串 ************
String httpRequestMethod = "POST";
String canonicalUri = "/";
String canonicalQueryString = "";
String canonicalHeaders = "content-type:application/json; charset=utf-8\n"
+ "host:" + host + "\n" + "x-tc-action:" + action.toLowerCase() + "\n";
String signedHeaders = "content-type;host;x-tc-action";

String payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}";
String hashedRequestPayload = sha256Hex(payload);
String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryString + "\n"
+ canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
System.out.println(canonicalRequest);

// ************ 步骤 2: 拼接待签名字符串 ************
String credentialScope = date + "/" + service + "/" + "tc3_request";
String hashedCanonicalRequest = sha256Hex(canonicalRequest);
String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
System.out.println(stringToSign);

// ************ 步骤 3: 计算签名 ************
byte[] secretDate = hmac256(("TC3" + SECRET_KEY).getBytes(UTF8), date);
byte[] secretService = hmac256(secretDate, service);
byte[] secretSigning = hmac256(secretService, "tc3_request");
String signature = DatatypeConverter.printHexBinary(hmac256(secretSigning, stringToSign)).toLowerCase();
System.out.println(signature);
```

```
// ************* 步骤 4：拼接 Authorization *************
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
System.out.println(authorization);

TreeMap<String, String> headers = new TreeMap<String, String>();
headers.put("Authorization", authorization);
headers.put("Content-Type", CT_JSON);
headers.put("Host", host);
headers.put("X-TC-Action", action);
headers.put("X-TC-Timestamp", timestamp);
headers.put("X-TC-Version", version);
headers.put("X-TC-Region", region);

StringBuilder sb = new StringBuilder();
sb.append("curl -X POST https://").append(host)
.append(" -H \"Authorization: ").append(authorization).append("\"")
.append(" -H \"Content-Type: application/json; charset=utf-8\"")
.append(" -H \"Host: ").append(host).append("\"")
.append(" -H \"X-TC-Action: ").append(action).append("\"")
.append(" -H \"X-TC-Timestamp: ").append(timestamp).append("\"")
.append(" -H \"X-TC-Version: ").append(version).append("\"")
.append(" -H \"X-TC-Region: ").append(region).append("\"")
.append(" -d '").append(payload).append("'");
System.out.println(sb.toString());
}
}
```

Python

```
# -*- coding: utf-8 -*-
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# 密钥参数
# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secret_key = os.environ.get("TENCENTCLOUD_SECRET_KEY")

service = "cvm"
host = "cvm.tencentcloudapi.com"
endpoint = "https://" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
#timestamp = int(time.time())
timestamp = 1551113065
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 1, "Filters": [{"Values": [u"未命名"], "Name": "instance-name"}]}

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = "POST"
canonical_uri = "/"
canonical_querystring = ""
```

```
ct = "application/json; charset=utf-8"
payload = json.dumps(params)
canonical_headers = "content-type:%s\nhost:%s\nx-tc-action:%s\n" % (ct, host, action.lower())
signed_headers = "content-type;host;x-tc-action"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
canonical_uri + "\n" +
canonical_querystring + "\n" +
canonical_headers + "\n" +
signed_headers + "\n" +
hashed_request_payload)
print(canonical_request)

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
str(timestamp) + "\n" +
credential_scope + "\n" +
hashed_canonical_request)
print(string_to_sign)


# ************* 步骤 3：计算签名 *************
# 计算签名摘要函数
def sign(key, msg):
return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
print(signature)

# ************* 步骤 4：拼接 Authorization *************
authorization = (algorithm + " " +
"Credential=" + secret_id + "/" + credential_scope + ", " +
"SignedHeaders=" + signed_headers + ", " +
"Signature=" + signature)
print(authorization)

print('curl -X POST ' + endpoint
+ ' -H "Authorization: ' + authorization + '"'
+ ' -H "Content-Type: application/json; charset=utf-8"'
+ ' -H "Host: ' + host + '"'
+ ' -H "X-TC-Action: ' + action + '"'
+ ' -H "X-TC-Timestamp: ' + str(timestamp) + '"'
+ ' -H "X-TC-Version: ' + version + '"'
+ ' -H "X-TC-Region: ' + region + '"'
+ " -d '" + payload + "'")
```

Golang

```
package main

import (
"crypto/hmac"
```

```
"crypto/sha256"
"encoding/hex"
"fmt"
"os"
"strings"
"time"
)

func sha256hex(s string) string {
b := sha256.Sum256([]byte(s))
return hex.EncodeToString(b[:])
}

func hmacsha256(s, key string) string {
hashed := hmac.New(sha256.New, []byte(key))
hashed.Write([]byte(s))
return string(hashed.Sum(nil))
}

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
host := "cvm.tencentcloudapi.com"
algorithm := "TC3-HMAC-SHA256"
service := "cvm"
version := "2017-03-12"
action := "DescribeInstances"
region := "ap-guangzhou"
//var timestamp int64 = time.Now().Unix()
var timestamp int64 = 1551113065

// step 1: build canonical request string
httpRequestMethod := "POST"
canonicalURI := "/"
canonicalQueryString := ""
canonicalHeaders := fmt.Sprintf("content-type:%s\nhost:%s\nx-tc-action:%s\n",
"application/json; charset=utf-8", host, strings.ToLower(action))
signedHeaders := "content-type;host;x-tc-action"
payload := `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`
hashedRequestPayload := sha256hex(payload)
canonicalRequest := fmt.Sprintf("%s\n%s\n%s\n%s\n%s\n%s",
httpRequestMethod,
canonicalURI,
canonicalQueryString,
canonicalHeaders,
signedHeaders,
hashedRequestPayload)
fmt.Println(canonicalRequest)

// step 2: build string to sign
date := time.Unix(timestamp, 0).UTC().Format("2006-01-02")
credentialScope := fmt.Sprintf("%s/%s/tc3_request", date, service)
hashedCanonicalRequest := sha256hex(canonicalRequest)
string2sign := fmt.Sprintf("%s\n%d\n%s\n%s",
algorithm,
```

```
timestamp,
credentialScope,
hashedCanonicalRequest)
fmt.Println(string2sign)

// step 3: sign string
secretDate := hmacsha256(date, "TC3"+secretKey)
secretService := hmacsha256(service, secretDate)
secretSigning := hmacsha256("tc3_request", secretService)
signature := hex.EncodeToString([]byte(hmacsha256(string2sign, secretSigning)))
fmt.Println(signature)

// step 4: build authorization
authorization := fmt.Sprintf("%s Credential=%s/%s, SignedHeaders=%s, Signature=%s",
algorithm,
secretId,
credentialScope,
signedHeaders,
signature)
fmt.Println(authorization)

curl := fmt.Sprintf(`curl -X POST https://%s\
-H "Authorization: %s"\
-H "Content-Type: application/json; charset=utf-8"\
-H "Host: %s" -H "X-TC-Action: %s"\
-H "X-TC-Timestamp: %d"\
-H "X-TC-Version: %s"\
-H "X-TC-Region: %s"\
-d '%s'`, host, authorization, host, action, timestamp, version, region, payload)
fmt.Println(curl)
}
```

PHP

```
<?php
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$host = "cvm.tencentcloudapi.com";
$service = "cvm";
$version = "2017-03-12";
$action = "DescribeInstances";
$region = "ap-guangzhou";
// $timestamp = time();
$timestamp = 1551113065;
$algorithm = "TC3-HMAC-SHA256";

// step 1: build canonical request string
$httpRequestMethod = "POST";
$canonicalUri = "/";
$canonicalQueryString = "";
$canonicalHeaders = implode("\n", [
"content-type:application/json; charset=utf-8",
"host:".$host,
"x-tc-action:".strtolower($action),
```

```
""
]);
$signedHeaders = implode(";", [
"content-type",
"host",
"x-tc-action",
]);
$payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}';
$hashedRequestPayload = hash("SHA256", $payload);
$canonicalRequest = $httpRequestMethod."\n"
.$canonicalUri."\n"
.$canonicalQueryString."\n"
.$canonicalHeaders."\n"
.$signedHeaders."\n"
.$hashedRequestPayload;
echo $canonicalRequest.PHP_EOL;

// step 2: build string to sign
$date = gmdate("Y-m-d", $timestamp);
$credentialScope = $date."/".$service."/tc3_request";
$hashedCanonicalRequest = hash("SHA256", $canonicalRequest);
$stringToSign = $algorithm."\n"
.$timestamp."\n"
.$credentialScope."\n"
.$hashedCanonicalRequest;
echo $stringToSign.PHP_EOL;

// step 3: sign string
$secretDate = hash_hmac("SHA256", $date, "TC3".$secretKey, true);
$secretService = hash_hmac("SHA256", $service, $secretDate, true);
$secretSigning = hash_hmac("SHA256", "tc3_request", $secretService, true);
$signature = hash_hmac("SHA256", $stringToSign, $secretSigning);
echo $signature.PHP_EOL;

// step 4: build authorization
$authorization = $algorithm
." Credential=".$secretId."/".$credentialScope
.", SignedHeaders=".$signedHeaders.", Signature=".$signature;
echo $authorization.PHP_EOL;

$curl = "curl -X POST https://".$host
.' -H "Authorization: '.$authorization.'"'
.' -H "Content-Type: application/json; charset=utf-8"'
.' -H "Host: '.$host.'"'
.' -H "X-TC-Action: '.$action.'"'
.' -H "X-TC-Timestamp: '.$timestamp.'"'
.' -H "X-TC-Version: '.$version.'"'
.' -H "X-TC-Region: '.$region.'"'
." -d '".$payload."'";
echo $curl.PHP_EOL;
```

Ruby

```
# -*- coding: UTF-8 -*-
# require ruby>=2.3.0
require 'digest'
```

```ruby
require 'json'
require 'time'
require 'openssl'

# 密钥参数
# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

service = 'cvm'
host = 'cvm.tencentcloudapi.com'
endpoint = 'https://' + host
region = 'ap-guangzhou'
action = 'DescribeInstances'
version = '2017-03-12'
algorithm = 'TC3-HMAC-SHA256'
# timestamp = Time.now.to_i
timestamp = 1551113065
date = Time.at(timestamp).utc.strftime('%Y-%m-%d')

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = 'POST'
canonical_uri = '/'
canonical_querystring = ''
canonical_headers = "content-type:application/json; charset=utf-8\nhost:#{host}\nx-tc-action:#{action.downcase}\n"
signed_headers = 'content-type;host;x-tc-action'
# params = { 'Limit' => 1, 'Filters' => [{ 'Name' => 'instance-name', 'Values' => ['未命名'] }] }
# payload = JSON.generate(params, { 'ascii_only' => true, 'space' => ' ' })
# json will generate in random order, to get specified result in example, we hard-code it here.
payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
hashed_request_payload = Digest::SHA256.hexdigest(payload)
canonical_request = [
http_request_method,
canonical_uri,
canonical_querystring,
canonical_headers,
signed_headers,
hashed_request_payload,
].join("\n")

puts canonical_request

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + '/' + service + '/' + 'tc3_request'
hashed_request_payload = Digest::SHA256.hexdigest(canonical_request)
string_to_sign = [
algorithm,
timestamp.to_s,
credential_scope,
hashed_request_payload,
].join("\n")
puts string_to_sign

# ************* 步骤 3：计算签名 *************
digest = OpenSSL::Digest.new('sha256')
secret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)
```

```
secret_service = OpenSSL::HMAC.digest(digest, secret_date, service)
secret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')
signature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)
puts signature

# ************* 步骤 4：拼接 Authorization *************
authorization = "#{algorithm} Credential=#{secret_id}/#{credential_scope}, SignedHeaders=#{signed_headers}, Signature=#{si
gnature}"
puts authorization

puts 'curl -X POST ' + endpoint \
+ ' -H "Authorization: ' + authorization + '"' \
+ ' -H "Content-Type: application/json; charset=utf-8"' \
+ ' -H "Host: ' + host + '"' \
+ ' -H "X-TC-Action: ' + action + '"' \
+ ' -H "X-TC-Timestamp: ' + timestamp.to_s + '"' \
+ ' -H "X-TC-Version: ' + version + '"' \
+ ' -H "X-TC-Region: ' + region + '"' \
+ " -d '" + payload + "'"
```

DotNet

```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

public class Application
{
public static string SHA256Hex(string s)
{
using (SHA256 algo = SHA256.Create())
{
byte[] hashbytes = algo.ComputeHash(Encoding.UTF8.GetBytes(s));
StringBuilder builder = new StringBuilder();
for (int i = 0; i < hashbytes.Length; ++i)
{
builder.Append(hashbytes[i].ToString("x2"));
}
return builder.ToString();
}
}

public static byte[] HmacSHA256(byte[] key, byte[] msg)
{
using (HMACSHA256 mac = new HMACSHA256(key))
{
return mac.ComputeHash(msg);
}
}

public static Dictionary<String, String> BuildHeaders(string secretid,
string secretkey, string service, string endpoint, string region,
string action, string version, DateTime date, string requestPayload)
{
string datestr = date.ToString("yyyy-MM-dd");
```

```
DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, MidpointRounding.AwayFromZero) / 1000;
// ************* 步骤 1: 拼接规范请求串 *************
string algorithm = "TC3-HMAC-SHA256";
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string contentType = "application/json";
string canonicalHeaders = "content-type:" + contentType + "; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.ToLower() + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string hashedRequestPayload = SHA256Hex(requestPayload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
Console.WriteLine(canonicalRequest);

// ************* 步骤 2: 拼接待签名字符串 *************
string credentialScope = datestr + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = SHA256Hex(canonicalRequest);
string stringToSign = algorithm + "\n"
+ requestTimestamp.ToString() + "\n"
+ credentialScope + "\n"
+ hashedCanonicalRequest;
Console.WriteLine(stringToSign);

// ************* 步骤 3: 计算签名 *************
byte[] tc3SecretKey = Encoding.UTF8.GetBytes("TC3" + secretkey);
byte[] secretDate = HmacSHA256(tc3SecretKey, Encoding.UTF8.GetBytes(datestr));
byte[] secretService = HmacSHA256(secretDate, Encoding.UTF8.GetBytes(service));
byte[] secretSigning = HmacSHA256(secretService, Encoding.UTF8.GetBytes("tc3_request"));
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringToSign));
string signature = BitConverter.ToString(signatureBytes).Replace("-", "").ToLower();
Console.WriteLine(signature);

// ************* 步骤 4: 拼接 Authorization *************
string authorization = algorithm + " "
+ "Credential=" + secretid + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", "
+ "Signature=" + signature;
Console.WriteLine(authorization);

Dictionary<string, string> headers = new Dictionary<string, string>();
headers.Add("Authorization", authorization);
headers.Add("Host", endpoint);
headers.Add("Content-Type", contentType + "; charset=utf-8");
headers.Add("X-TC-Timestamp", requestTimestamp.ToString());
headers.Add("X-TC-Version", version);
headers.Add("X-TC-Action", action);
headers.Add("X-TC-Region", region);
return headers;
}
public static void Main(string[] args)
```

```
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID*******************************
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

string service = "cvm";
string endpoint = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";

// 此处由于示例规范的原因，采用时间戳2019-02-26 00:44:25，此参数作为示例，如果在项目中，您应当使用：
// DateTime date = DateTime.UtcNow;
// 注意时区，建议此时间统一采用UTC时间戳，否则容易出错
DateTime date = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc).AddSeconds(1551113065);
string requestPayload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name
\"}]}";

Dictionary<string, string> headers = BuildHeaders(SECRET_ID, SECRET_KEY, service
, endpoint, region, action, version, date, requestPayload);

Console.WriteLine("POST https://cvm.tencentcloudapi.com");
foreach (KeyValuePair<string, string> kv in headers)
{
Console.WriteLine(kv.Key + ": " + kv.Value);
}
Console.WriteLine();
Console.WriteLine(requestPayload);
}
}
```

NodeJS

```
const crypto = require('crypto');

function sha256(message, secret = '', encoding) {
const hmac = crypto.createHmac('sha256', secret)
return hmac.update(message).digest(encoding)
}

function getHash(message, encoding = 'hex') {
const hash = crypto.createHash('sha256')
return hash.update(message).digest(encoding)
}

function getDate(timestamp) {
const date = new Date(timestamp * 1000)
const year = date.getUTCFullYear()
const month = ('0' + (date.getUTCMonth() + 1)).slice(-2)
const day = ('0' + date.getUTCDate()).slice(-2)
return `${year}-${month}-${day}`
}

function main(){
```

```
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

const endpoint = "cvm.tencentcloudapi.com"
const service = "cvm"
const region = "ap-guangzhou"
const action = "DescribeInstances"
const version = "2017-03-12"
//const timestamp = getTime()
const timestamp = 1551113065
//时间处理，获取世界时间日期
const date = getDate(timestamp)

// ************* 步骤 1：拼接规范请求串 *************
const payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}"

const hashedRequestPayload = getHash(payload);
const httpRequestMethod = "POST"
const canonicalUri = "/"
const canonicalQueryString = ""
const canonicalHeaders = "content-type:application/json; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.toLowerCase() + "\n"
const signedHeaders = "content-type;host;x-tc-action"

const canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload
console.log(canonicalRequest)

// ************* 步骤 2：拼接待签名字符串 *************
const algorithm = "TC3-HMAC-SHA256"
const hashedCanonicalRequest = getHash(canonicalRequest);
const credentialScope = date + "/" + service + "/" + "tc3_request"
const stringToSign = algorithm + "\n" +
timestamp + "\n" +
credentialScope + "\n" +
hashedCanonicalRequest
console.log(stringToSign)

// ************* 步骤 3：计算签名 *************
const kDate = sha256(date, 'TC3' + SECRET_KEY)
const kService = sha256(service, kDate)
const kSigning = sha256('tc3_request', kService)
const signature = sha256(stringToSign, kSigning, 'hex')
console.log(signature)

// ************* 步骤 4：拼接 Authorization *************
const authorization = algorithm + " " +
"Credential=" + SECRET_ID + "/" + credentialScope + ", " +
"SignedHeaders=" + signedHeaders + ", " +
```

```
"Signature=" + signature
console.log(authorization)

const curlcmd = 'curl -X POST ' + "https://" + endpoint
+ ' -H "Authorization: ' + authorization + '"'
+ ' -H "Content-Type: application/json; charset=utf-8"'
+ ' -H "Host: ' + endpoint + '"'
+ ' -H "X-TC-Action: ' + action + '"'
+ ' -H "X-TC-Timestamp: ' + timestamp.toString() + '"'
+ ' -H "X-TC-Version: ' + version + '"'
+ ' -H "X-TC-Region: ' + region + '"'
+ " -d '" + payload + "'"
console.log(curlcmd)
}
main()
```

C++

```
#include <algorithm>
#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdio.h>
#include <time.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>

using namespace std;

string get_data(int64_t &timestamp)
{
string utcDate;
char buff[20] = {0};
// time_t timenow;
struct tm sttime;
sttime = *gmtime(&timestamp);
strftime(buff, sizeof(buff), "%Y-%m-%d", &sttime);
utcDate = string(buff);
return utcDate;
}

string int2str(int64_t n)
{
std::stringstream ss;
ss << n;
return ss.str();
}

string sha256Hex(const string &str)
{
char buf[3];
unsigned char hash[SHA256_DIGEST_LENGTH];
SHA256_CTX sha256;
SHA256_Init(&sha256);
```

```
SHA256_Update(&sha256, str.c_str(), str.size());
SHA256_Final(hash, &sha256);
std::string NewString = "";
for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
{
snprintf(buf, sizeof(buf), "%02x", hash[i]);
NewString = NewString + buf;
}
return NewString;
}

string HmacSha256(const string &key, const string &input)
{
unsigned char hash[32];

HMAC_CTX *h;
#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif

HMAC_Init_ex(h, &key[0], key.length(), EVP_sha256(), NULL);
HMAC_Update(h, ( unsigned char* )&input[0], input.length());
unsigned int len = 32;
HMAC_Final(h, hash, &len);

#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

std::stringstream ss;
ss << std::setfill('0');
for (int i = 0; i < len; i++)
{
ss << hash[i];
}

return (ss.str());
}

string HexEncode(const string &input)
{
static const char* const lut = "0123456789abcdef";
size_t len = input.length();

string output;
output.reserve(2 * len);
for (size_t i = 0; i < len; ++i)
{
const unsigned char c = input[i];
output.push_back(lut[c >> 4]);
output.push_back(lut[c & 15]);
```

```
}
return output;
}

int main()
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
string SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");

string service = "cvm";
string host = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";
int64_t timestamp = 1551113065;
string date = get_data(timestamp);

// ************* 步骤 1: 拼接规范请求串 *************
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string lower = action;
std::transform(action.begin(), action.end(), lower.begin(), ::tolower);
string canonicalHeaders = string("content-type:application/json; charset=utf-8\n")
+ "host:" + host + "\n"
+ "x-tc-action:" + lower + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}";
string hashedRequestPayload = sha256Hex(payload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
cout << canonicalRequest << endl;

// ************* 步骤 2: 拼接待签名字符串 *************
string algorithm = "TC3-HMAC-SHA256";
string RequestTimestamp = int2str(timestamp);
string credentialScope = date + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = sha256Hex(canonicalRequest);
string stringToSign = algorithm + "\n" + RequestTimestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
cout << stringToSign << endl;

// ************* 步骤 3: 计算签名 ***************
string kKey = "TC3" + SECRET_KEY;
string kDate = HmacSha256(kKey, date);
string kService = HmacSha256(kDate, service);
string kSigning = HmacSha256(kService, "tc3_request");
string signature = HexEncode(HmacSha256(kSigning, stringToSign));
cout << signature << endl;

// ************* 步骤 4: 拼接 Authorization *************
```

```
string authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
cout << authorization << endl;

string curlcmd = "curl -X POST https://" + host + "\n"
+ " -H \"Authorization: " + authorization + "\"\n"
+ " -H \"Content-Type: application/json; charset=utf-8\"" + "\n"
+ " -H \"Host: " + host + "\"\n"
+ " -H \"X-TC-Action: " + action + "\"\n"
+ " -H \"X-TC-Timestamp: " + RequestTimestamp + "\"\n"
+ " -H \"X-TC-Version: " + version + "\"\n"
+ " -H \"X-TC-Region: " + region + "\"\n"
+ " -d '" + payload + "\'";
cout << curlcmd << endl;
return 0;
};
```

C

```
#include <ctype.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>

void get_utc_date(int64_t timestamp, char* utc, int len)
{
// time_t timenow;
struct tm sttime;
sttime = *gmtime(&timestamp);
strftime(utc, len, "%Y-%m-%d", &sttime);
}

void sha256_hex(const char* str, char* result)
{
char buf[3];
unsigned char hash[SHA256_DIGEST_LENGTH];
SHA256_CTX sha256;
SHA256_Init(&sha256);
SHA256_Update(&sha256, str, strlen(str));
SHA256_Final(hash, &sha256);
for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
{
snprintf(buf, sizeof(buf), "%02x", hash[i]);
strcat(result, buf);
}
}

void hmac_sha256(const char* key, int key_len,
const char* input, int input_len,
unsigned char* output, unsigned int* output_len)
{
HMAC_CTX *h;
```

```
#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif

HMAC_Init_ex(h, key, key_len, EVP_sha256(), NULL);
HMAC_Update(h, ( unsigned char* )input, input_len);
HMAC_Final(h, output, output_len);


#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

}

void hex_encode(const char* input, int input_len, char* output)
{
static const char* const lut = "0123456789abcdef";

char add_out[128] = {0};
char temp[2] = {0};
for (size_t i = 0; i < input_len; ++i)
{
const unsigned char c = input[i];
temp[0] = lut[c >> 4];
strcat(add_out, temp);
temp[0] = lut[c & 15];
strcat(add_out, temp);
}
strncpy(output, add_out, 128);
}

void lowercase(const char * src, char * dst)
{
for (int i = 0; src[i]; i++)
{
dst[i] = tolower(src[i]);
}
}

int main()
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
const char* SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const char* SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");
const char* service = "cvm";
const char* host = "cvm.tencentcloudapi.com";
const char* region = "ap-guangzhou";
const char* action = "DescribeInstances";
```

```
const char* version = "2017-03-12";
int64_t timestamp = 1551113065;
char date[20] = {0};
get_utc_date(timestamp, date, sizeof(date));

// ************* 步骤 1: 拼接规范请求串 *************
const char* http_request_method = "POST";
const char* canonical_uri = "/";
const char* canonical_query_string = "";
char canonical_headers[100] = {"content-type:application/json; charset=utf-8\nhost:"};
strcat(canonical_headers, host);
strcat(canonical_headers, "\nx-tc-action:");
char value[100] = {0};
lowercase(action, value);
strcat(canonical_headers, value);
strcat(canonical_headers, "\n");
const char* signed_headers = "content-type;host;x-tc-action";
const char* payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name
\"}]}";
char hashed_request_payload[100] = {0};
sha256_hex(payload, hashed_request_payload);

char canonical_request[256] = {0};
sprintf(canonical_request, "%s\n%s\n%s\n%s\n%s\n%s", http_request_method,
canonical_uri, canonical_query_string, canonical_headers,
signed_headers, hashed_request_payload);
printf("%s\n", canonical_request);


// ************* 步骤 2: 拼接待签名字符串 *************
const char* algorithm = "TC3-HMAC-SHA256";
char request_timestamp[16] = {0};
sprintf(request_timestamp, "%d", timestamp);
char credential_scope[64] = {0};
strcat(credential_scope, date);
sprintf(credential_scope, "%s/%s/tc3_request", date, service);
char hashed_canonical_request[100] = {0};
sha256_hex(canonical_request, hashed_canonical_request);
char string_to_sign[256] = {0};
sprintf(string_to_sign, "%s\n%s\n%s\n%s", algorithm, request_timestamp,
credential_scope, hashed_canonical_request);
printf("%s\n", string_to_sign);

// ************* 步骤 3: 计算签名 ***************
char k_key[64] = {0};
sprintf(k_key, "%s%s", "TC3", SECRET_KEY);
unsigned char k_date[64] = {0};
unsigned int output_len = 0;
hmac_sha256(k_key, strlen(k_key), date, strlen(date), k_date, &output_len);
unsigned char k_service[64] = {0};
hmac_sha256(k_date, output_len, service, strlen(service), k_service, &output_len);
unsigned char k_signing[64] = {0};
hmac_sha256(k_service, output_len, "tc3_request", strlen("tc3_request"), k_signing, &output_len);
unsigned char k_hmac_sha_sign[64] = {0};
hmac_sha256(k_signing, output_len, string_to_sign, strlen(string_to_sign), k_hmac_sha_sign, &output_len);

char signature[128] = {0};
```

```
hex_encode(k_hmac_sha_sign, output_len, signature);
printf("%s\n", signature);

// ************* 步骤 4: 拼接 Authorization *************
char authorization[512] = {0};
sprintf(authorization, "%s Credential=%s/%s, SignedHeaders=%s, Signature=%s",
algorithm, SECRET_ID, credential_scope, signed_headers, signature);
printf("%s\n", authorization);

char curlcmd[10240] = {0};
sprintf(curlcmd, "curl -X POST https://%s\n \
-H \"Authorization: %s\"\n \
-H \"Content-Type: application/json; charset=utf-8\"\n \
-H \"Host: %s\"\n \
-H \"X-TC-Action: %s\"\n \
-H \"X-TC-Timestamp: %s\"\n \
-H \"X-TC-Version: %s\"\n \
-H \"X-TC-Region: %s\"\n \
-d \'%s\'",
host, authorization, host, action, request_timestamp, version, region, payload);
printf("%s\n", curlcmd);
return 0;
}
```

其他语言

- Lua: GitHub
- Swift: GitHub
- Dart: GitHub
- Shell(Bash): GitHub

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

错误码	错误描述
AuthFailure.SignatureExpire	签名过期。Timestamp 与服务器接收到请求的时间相差不得超过五分钟。
AuthFailure.SecretIdNotFound	密钥不存在。请到控制台查看密钥是否被禁用，是否少复制了字符或者多了字符。
AuthFailure.SignatureFailure	签名错误。可能是签名计算错误，或者签名与实际发送的内容不相符合，也有可能是密钥 SecretKey 错误导致的。
AuthFailure.TokenFailure	临时证书 Token 错误。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。

签名方法

最近更新时间：2024-12-25 02:03:52

签名方法 v1 简单易用，但是功能和安全性都不如签名方法 v3，推荐使用签名方法 v3。

首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v1”，可以生成签名过程进行验证，并提供了部分编程语言的签名示例，也可直接生成 SDK 代码。推荐使用腾讯云 API 配套的 8 种常见的编程语言 SDK，已经封装了签名和请求过程，均已开源，支持 Python、Java、PHP、Go、NodeJS、.NET、C++、Ruby。

推荐使用 API Explorer　　点击调试

您可以通过 API Explorer 的【签名串生成】模块查看每个接口签名的生成过程。

腾讯云 API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。
签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往 云API密钥页面 申请，否则无法调用云 API 接口。

1. 申请安全凭证

在第一次使用云 API 之前，请前往 云 API 密钥页面 申请安全凭证。
安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- **用户必须严格保管安全凭证，避免泄露。**

申请安全凭证的具体步骤如下：

1. 登录 腾讯云管理中心控制台 。
2. 前往 云 API 密钥 的控制台页面
3. 在 云 API 密钥 页面，单击【新建密钥】即可以创建一对 SecretId/SecretKey。

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

有了安全凭证 SecretId 和 SecretKey 后，就可以生成签名串了。以下是使用签名方法 v1 生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKID********************************
- SecretKey: ********************************

注意：这里只是示例，请根据用户实际申请的 SecretId 和 SecretKey 进行后续操作！

以云服务器查看实例列表（DescribeInstances）请求为例，当用户调用这一接口时，其请求参数可能如下：

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥 ID	AKID********************************
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例 ID	ins-09dx96dg
Offset	偏移量	0
Limit	最大允许输出	20
Version	接口版本号	2017-03-12

这里只展示了部分公共参数和接口输入参数，用户可以根据实际需要添加其他参数，例如 Language 和 Token 公共参数。

2.1. 对参数排序

首先对所有请求参数按参数名的字典序（ ASCII 码）升序排序。注意：1）只按参数名进行排序，参数值保持对应即可，不参与比大小；2）按 ASCII 码比大小，如 InstanceIds.2 要排在 InstanceIds.12 后面，不是按字母表，也不是按数值。用户可以借助编程语言中的相关排序函数来实现这一功能，如 PHP 中的 ksort 函数。上述示例参数的排序结果如下:

```
{
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : 'AKID********************************',
'Timestamp' : 1465185768,
'Version': '2017-03-12',
}
```

使用其它程序设计语言开发时，可对上面示例中的参数进行排序，得到的结果一致即可。

2.2. 拼接请求字符串

此步骤生成请求字符串。
将把上一步排序好的请求参数格式化成“参数名称=参数值”的形式，如对 Action 参数，其参数名称为 "Action" ，参数值为 "DescribeInstances" ，因此格式化后就为 Action=DescribeInstances 。
注意：“参数值”为原始值而非 url 编码后的值。

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID*******
************************&Timestamp=1465185768&Version=2017-03-12
```

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机:查看实例列表(DescribeInstances)的请求域名为：cvm.tencentcloudapi.com。实际的请求域名根据接口所属模块的不同而不同，详见各接口说明。
3. 请求路径: 当前版本云API的请求路径固定为 / 。
4. 请求字符串: 即上一步生成的请求字符串。

签名原文串的拼接规则为： 请求方法 + 请求主机 +请求路径 + ? + 请求字符串 。

示例的拼接结果为:

```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-gu
angzhou&SecretId=AKID********************************&Timestamp=1465185768&Version=2017-03-12
```

2.4. 生成签名串

此步骤生成签名串。
首先使用 HMAC-SHA1 算法对上一步中获得的**签名原文字符串**进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例:

```
$secretKey = '********************************';
$srcStr = 'GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&R
egion=ap-guangzhou&SecretId=AKID********************************&Timestamp=1465185768&Version=2017-03-12';
```

```
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

使用其它程序设计语言开发时，可用上面示例中的原文进行签名验证，得到的签名串与例子中的一致即可。

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 7RAM2xfNMO9EiVTNmPg06MRnCvQ= ，最终得到的签名串请求参数（Signature）为：7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D，它将用于生成最终的请求 URL。

注意：如果用户的请求方法是 GET，或者请求方法为 POST 同时 Content-Type 为 application/x-www-form-urlencoded，则发送请求时所有请求参数的值均需要做 URL 编码，参数键和=符号不需要编码。非 ASCII 字符在 URL 编码前需要先以 UTF-8 进行编码。

注意：有些编程语言的网络库会自动为所有参数进行 urlencode，在这种情况下，就不需要对签名串进行 URL 编码了，否则两次 URL 编码会导致签名失败。

注意：其他参数值也需要进行编码，编码采用 RFC 3986。使用 %XY 对特殊字符例如汉字进行百分比编码，其中“X”和“Y”为十六进制字符（0-9 和大写字母 A-F），使用小写将引发错误。

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

错误代码	错误描述
AuthFailure.SignatureExpire	签名过期
AuthFailure.SecretIdNotFound	密钥不存在
AuthFailure.SignatureFailure	签名错误
AuthFailure.TokenFailure	token 错误
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）

5. 签名演示

在实际调用 API 3.0 时，推荐使用配套的腾讯云 SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

下面提供了不同产品的生成签名 demo，您可以找到对应的产品参考签名的生成：

- Signature Demo

为了更清楚的解释签名过程，下面以实际编程语言为例，将上述的签名过程具体实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK 。

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-`

guangzhou&SecretId=AKID********************************&Signature=7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D&Timestamp=1465185768&Version=2017-03-12 。

注意：由于示例中的密钥是虚构的，时间戳也不是系统当前时间，因此如果将此 url 在浏览器中打开或者用 curl 等命令调用时会返回鉴权错误：签名过期。为了得到一个可以正常返回的 url ，需要修改示例中的 SecretId 和 SecretKey 为真实的密钥，并使用系统当前时间戳作为 Timestamp 。

注意：在下面的示例中，不同编程语言，甚至同一语言每次执行得到的 url 可能都有所不同，表现为参数的顺序不同，但这并不影响正确性。只要所有参数都在，且签名计算正确即可。

注意：以下代码仅适用于 API 3.0，不能直接用于其他的签名流程，请以对应的实际文档为准。

Java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TencentCloudAPIDemo {
private final static String CHARSET = "UTF-8";

public static String sign(String s, String key, String method) throws Exception {
Mac mac = Mac.getInstance(method);
SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAlgorithm());
mac.init(secretKeySpec);
byte[] hash = mac.doFinal(s.getBytes(CHARSET));
return DatatypeConverter.printBase64Binary(hash);
}

public static String getStringToSign(TreeMap<String, Object> params) {
StringBuilder s2s = new StringBuilder("GETcvm.tencentcloudapi.com/?");
// 签名时要求对参数进行字典排序，此处用TreeMap保证顺序
for (String k : params.keySet()) {
s2s.append(k).append("=").append(params.get(k).toString()).append("&");
}
return s2s.toString().substring(0, s2s.length() - 1);
}

public static String getUrl(TreeMap<String, Object> params) throws UnsupportedEncodingException {
StringBuilder url = new StringBuilder("https://cvm.tencentcloudapi.com/?");
// 实际请求的url中对参数顺序没有要求
for (String k : params.keySet()) {
// 需要对请求串进行urlencode，由于key都是英文字母，故此处仅对其value进行urlencode
url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).append("&");
}
return url.toString().substring(0, url.length() - 1);
}

public static void main(String[] args) throws Exception {
TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
// 实际调用时应当使用随机数，例如: params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
params.put("Nonce", 11886); // 公共参数
// 实际调用时应当使用系统当前时间，例如: params.put("Timestamp", System.currentTimeMillis() / 1000);
params.put("Timestamp", 1465185768); // 公共参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
params.put("SecretId", System.getenv("TENCENTCLOUD_SECRET_ID")); // 公共参数
```

```
params.put("Action", "DescribeInstances"); // 公共参数
params.put("Version", "2017-03-12"); // 公共参数
params.put("Region", "ap-guangzhou"); // 公共参数
params.put("Limit", 20); // 业务参数
params.put("Offset", 0); // 业务参数
params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
params.put("Signature", sign(getStringToSign(params), System.getenv("TENCENTCLOUD_SECRET_KEY"), "HmacSHA1")); // 公共参数
System.out.println(getUrl(params));
}
}
```

Python

注意：如果是在 Python 2 环境中运行，需要先安装 requests 依赖包： `pip install requests` 。

```
# -*- coding: utf8 -*-
import base64
import hashlib
import hmac
import os
import time

import requests

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secret_key = os.environ.get("TENCENTCLOUD_SECRET_KEY")

def get_string_to_sign(method, endpoint, params):
s = method + endpoint + "/?"
query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
return s + query_str

def sign_str(key, s, method):
hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
return base64.b64encode(hmac_str)

if __name__ == '__main__':
endpoint = "cvm.tencentcloudapi.com"
data = {
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : secret_id,
'Timestamp' : 1465185768, # int(time.time())
'Version': '2017-03-12'
}
s = get_string_to_sign("GET", endpoint, data)
data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
print(data["Signature"])
# 此处会实际调用，成功后可能产生计费
```

```
# resp = requests.get("https://" + endpoint, params=data)
# print(resp.url)
```

Golang

```
package main

import (
"bytes"
"crypto/hmac"
"crypto/sha1"
"encoding/base64"
"fmt"
"os"
"sort"
"strconv"
)

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
params := map[string]string{
"Nonce": "11886",
"Timestamp": strconv.Itoa(1465185768),
"Region": "ap-guangzhou",
"SecretId": secretId,
"Version": "2017-03-12",
"Action": "DescribeInstances",
"InstanceIds.0": "ins-09dx96dg",
"Limit": strconv.Itoa(20),
"Offset": strconv.Itoa(0),
}

var buf bytes.Buffer
buf.WriteString("GET")
buf.WriteString("cvm.tencentcloudapi.com")
buf.WriteString("/")
buf.WriteString("?")

// sort keys by ascii asc order
keys := make([]string, 0, len(params))
for k, _ := range params {
keys = append(keys, k)
}
sort.Strings(keys)

for i := range keys {
k := keys[i]
buf.WriteString(k)
buf.WriteString("=")
buf.WriteString(params[k])
buf.WriteString("&")
}
buf.Truncate(buf.Len() - 1)
```

```
hashed := hmac.New(sha1.New, []byte(secretKey))
hashed.Write(buf.Bytes())

fmt.Println(base64.StdEncoding.EncodeToString(hashed.Sum(nil)))
}
```

PHP

```
<?php
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$param["Nonce"] = 11886;//rand();
$param["Timestamp"] = 1465185768;//time();
$param["Region"] = "ap-guangzhou";
$param["SecretId"] = $secretId;
$param["Version"] = "2017-03-12";
$param["Action"] = "DescribeInstances";
$param["InstanceIds.0"] = "ins-09dx96dg";
$param["Limit"] = 20;
$param["Offset"] = 0;

ksort($param);

$signStr = "GETcvm.tencentcloudapi.com/?";
foreach ( $param as $key => $value ) {
$signStr = $signStr . $key . "=" . $value . "&";
}
$signStr = substr($signStr, 0, -1);

$signature = base64_encode(hash_hmac("sha1", $signStr, $secretKey, true));
echo $signature.PHP_EOL;
// need to install and enable curl extension in php.ini
// $param["Signature"] = $signature;
// $url = "https://cvm.tencentcloudapi.com/?".http_build_query($param);
// echo $url.PHP_EOL;
// $ch = curl_init();
// curl_setopt($ch, CURLOPT_URL, $url);
// $output = curl_exec($ch);
// curl_close($ch);
// echo json_decode($output);
```

Ruby

```
# -*- coding: UTF-8 -*-
# require ruby>=2.3.0
require 'time'
require 'openssl'
require 'base64'

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
```

```
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

method = 'GET'
endpoint = 'cvm.tencentcloudapi.com'
data = {
'Action' => 'DescribeInstances',
'InstanceIds.0' => 'ins-09dx96dg',
'Limit' => 20,
'Nonce' => 11886,
'Offset' => 0,
'Region' => 'ap-guangzhou',
'SecretId' => secret_id,
'Timestamp' => 1465185768, # Time.now.to_i
'Version' => '2017-03-12',
}
sign = method + endpoint + '/?'
params = []
data.sort.each do |item|
params << "#{item[0]}=#{item[1]}"
end
sign += params.join('&')
digest = OpenSSL::Digest.new('sha1')
data['Signature'] = Base64.encode64(OpenSSL::HMAC.digest(digest, secret_key, sign))
puts data['Signature']

# require 'net/http'
# uri = URI('https://' + endpoint)
# uri.query = URI.encode_www_form(data)
# p uri
# res = Net::HTTP.get_response(uri)
# puts res.body
```

DotNet

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Security.Cryptography;
using System.Text;

public class Application {
public static string Sign(string signKey, string secret)
{
string signRet = string.Empty;
using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
{
byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
signRet = Convert.ToBase64String(hash);
}
return signRet;
}
public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, string reques
tHost, string requestPath)
{
string retStr = "";
retStr += requestMethod;
```

```
retStr += requestHost;
retStr += requestPath;
retStr += "?";
string v = "";
foreach (string key in requestParams.Keys)
{
v += string.Format("{0}={1}&", key, requestParams[key]);
}
retStr += v.TrimEnd('&');
return retStr;
}

public static void Main(string[] args)
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

string endpoint = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";
double RequestTimestamp = 1465185768; // 时间戳 2019-02-26 00:44:25,此参数作为示例，以实际为准
// long timestamp = ToTimestamp() / 1000;
// string requestTimestamp = timestamp.ToString();
Dictionary<string, string> param = new Dictionary<string, string>();
param.Add("Limit", "20");
param.Add("Offset", "0");
param.Add("InstanceIds.0", "ins-09dx96dg");
param.Add("Action", action);
param.Add("Nonce", "11886");
// param.Add("Nonce", Math.Abs(new Random().Next()).ToString());

param.Add("Timestamp", RequestTimestamp.ToString());
param.Add("Version", version);

param.Add("SecretId", SECRET_ID);
param.Add("Region", region);
SortedDictionary<string, string> headers = new SortedDictionary<string, string>(param, StringComparer.Ordinal);
string sigInParam = MakeSignPlainText(headers, "GET", endpoint, "/");
string sigOutParam = Sign(SECRET_KEY, sigInParam);
Console.WriteLine(sigOutParam);
}
}
```

NodeJS

```
const crypto = require('crypto');

function get_req_url(params, endpoint){
params['Signature'] = encodeURIComponent(params['Signature']);
const url_strParam = sort_params(params)
return "https://" + endpoint + "/?" + url_strParam.slice(1);
}
```

```
function formatSignString(reqMethod, endpoint, path, strParam){
let strSign = reqMethod + endpoint + path + "?" + strParam.slice(1);
return strSign;
}
function sha1(secretKey, strsign){
let signMethodMap = {'HmacSHA1': "sha1"};
let hmac = crypto.createHmac(signMethodMap['HmacSHA1'], secretKey || "");
return hmac.update(Buffer.from(strsign, 'utf8')).digest('base64')
}

function sort_params(params){
let strParam = "";
let keys = Object.keys(params);
keys.sort();
for (let k in keys) {
//k = k.replace(/_/g, '.');
strParam += ("&" + keys[k] + "=" + params[keys[k]]);
}
return strParam
}

function main(){
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

const endpoint = "cvm.tencentcloudapi.com"
const Region = "ap-guangzhou"
const Version = "2017-03-12"
const Action = "DescribeInstances"
const Timestamp = 1465185768 // 时间戳 2016-06-06 12:02:48，此参数作为示例，以实际为准
// const Timestamp = Math.round(Date.now() / 1000)
const Nonce = 11886 // 随机正整数
//const nonce = Math.round(Math.random() * 65535)

let params = {};
params['Action'] = Action;
params['InstanceIds.0'] = 'ins-09dx96dg';
params['Limit'] = 20;
params['Offset'] = 0;
params['Nonce'] = Nonce;
params['Region'] = Region;
params['SecretId'] = SECRET_ID;
params['Timestamp'] = Timestamp;
params['Version'] = Version;

// 1. 对参数排序，并拼接请求字符串
strParam = sort_params(params)

// 2. 拼接签名原文字符串
const reqMethod = "GET";
const path = "/";
strSign = formatSignString(reqMethod, endpoint, path, strParam)
// console.log(strSign)
```

```
// 3. 生成签名串
params['Signature'] = sha1(SECRET_KEY, strSign)
console.log(params['Signature'])

// 4. 进行url编码并拼接请求url
// const req_url = get_req_url(params, endpoint)
// console.log(params['Signature'])
// console.log(req_url)
}
main()
```

返回结果

最近更新时间：2024-03-12 01:55:27

云 API 3.0 接口默认返回 JSON 数据，返回非 JSON 格式的接口会在文档中做出说明。返回 JSON 数据时最大限制为 50 MB，如果返回的数据超过最大限制，请求会失败并返回内部错误。请根据接口文档中给出的过滤功能（例如时间范围）或者分页功能，控制返回数据不要过大。

注意：目前只要请求被服务端正常处理了，响应的 HTTP 状态码均为200。例如返回的消息体里的错误码是签名失败，但 HTTP 状态码是200，而不是401。

正确返回结果

以云服务器的接口查看实例状态列表 (DescribeInstancesStatus) 2017-03-12 版本为例，若调用成功，其可能的返回如下为：

```
{
"Response": {
"TotalCount": 0,
"InstanceStatusSet": [],
"RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
}
}
```

- Response 及其内部的 RequestId 是固定的字段，无论请求成功与否，只要 API 处理了，则必定会返回。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系 腾讯云客服 或 提交工单，并提供该 ID 来解决问题。
- 除了固定的字段外，其余均为具体接口定义的字段，不同的接口所返回的字段参见接口文档中的定义。此例中的 TotalCount 和 InstanceStatusSet 均为 DescribeInstancesStatus 接口定义的字段，由于调用请求的用户暂时还没有云服务器实例，因此 TotalCount 在此情况下的返回值为 0， InstanceStatusSet 列表为空。

错误返回结果

若调用失败，其返回值示例如下为：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

- Error 的出现代表着该请求调用失败。Error 字段连同其内部的 Code 和 Message 字段在调用失败时是必定返回的。
- Code 表示具体出错的错误码，当请求出错时可以先根据该错误码在公共错误码和当前接口对应的错误码列表里面查找对应原因和解决方案。
- Message 显示出了这个错误发生的具体原因，随着业务发展或体验优化，此文本可能会经常保持变更或更新，用户不应依赖这个返回值。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系 腾讯云客服 或 提交工单，并提供该 ID 来解决问题。

公共错误码

返回结果中如果存在 Error 字段，则表示调用 API 接口失败。 Error 中的 Code 字段表示错误码，所有业务都可能出现的错误码为公共错误码。完整的错误码列表请参考本产品“API 文档”目录下的“错误码”页面。

参数类型

最近更新时间：2023-03-29 12:02:12

目前腾讯云 API 3.0 输入参数和输出参数支持如下几种数据格式：

- String: 字符串。
- Integer：整型，上限为无符号64位整数。SDK 3.0 不同编程语言支持的类型有所差异，建议以所使用编程语言的最大整型定义，例如 `Golang` 的 `uint64`。
- Boolean：布尔型。
- Float：浮点型。
- Double：双精度浮点型。
- Date：字符串，日期格式。例如：2022-01-01。
- Timestamp：字符串，时间格式。例如：2022-01-01 00:00:00。
- Timestamp ISO8601：ISO 8601 是由国际标准化组织（International Organization for Standardization，ISO）发布的关于日期和时间格式的国际标准，对应国标《GB/T 7408-2005数据元和交换格式信息交换日期和时间表示法》。建议以所使用编程语言的标准库进行格式解析。例如：2022-01-01T00:00:00+08:00。
- Binary：二进制内容，需要以特定协议请求和解析。

节点相关接口
查询指定集群节点列表

最近更新时间：2025-04-25 02:01:39

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (DescribeNodes) 用于查询指定集群节点概览信息列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeNodes。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
Filters.N	否	Array of Filter	• **queue-name** 按照【**队列名称**】进行过滤。队列名称形如：compute。 类型：String 必选：否 • **node-role** 按照【**节点角色**】进行过滤。节点角色形如：Manager。（Manager：管控节点。Compute：计算节点。Login：登录节点。ManagerBackup：备用管控节点。） 类型：String 必选：否 • **node-type** 按照【**节点类型**】进行过滤。节点类型形如：STATIC。(STATIC：静态节点。DYNAMIC：弹性节点。) 类型：String 必选：否

参数名称	必选	类型	描述
			每次请求的 Filters 的上限为10， Filter.Values 的上限为5。
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：20

3. 输出参数

参数名称	类型	描述
NodeSet	Array of NodeOverview	节点概览信息列表。
TotalCount	Integer	符合条件的节点数量。 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询指定集群节点概览信息。

查询指定集群节点概览信息，集群ID为hpc-ggaqjyax。

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DescribeNodes
&ClusterId=hpc-ggaqjyax
&Offset=0
&Limit=1
&<公共请求参数>
```

输出示例


```
{
"Response": {
"NodeSet": [
{
"NodeId": "node-2sh754ah",
"InstanceId": "ins-j3n6kiae",
"Zone": "ap-chongqing-1",
"NodeState": "RUNNING",
"ImageId": "img-18og963d",
"QueueName": "compute",
"NodeRole": "Compute",
"NodeType": "DYNAMIC"
}
],
"TotalCount": 3,
"RequestId": "6793673e-3bce-4fbb-adea-923a82267da2"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidFilterNotSupportedName	不支持指定过滤器的键。
InvalidParameterValue.LimitExceeded	参数值数量超过限制。
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooSmall	参数值过小。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
RequestLimitExceeded	请求的次数超过了频率限制。
UnknownParameter	未知参数错误。
UnsupportedOperation.ParameterTooLarge	参数值过大，不支持此操作。
UnsupportedOperation.ParameterTooSmall	参数值过小，不支持此操作。

删除节点

最近更新时间：2025-04-25 02:01:39

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(DeleteNodes)用于删除指定集群中一个或者多个计算节点或者登录节点。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteNodes。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
NodeIds.N	是	Array of String	节点ID。 示例值：["node-xxx"]

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除节点

批量删除节点。

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DeleteNodes
&ClusterId=hpc-5lyv94lq
&NodeIds.0=ins-jhf9c1gi
&NodeIds.1=ins-qmzlqk70
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "f4d90874-f565-463d-ba1d-c707517eeaa1"
```

```
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.NodeId	无法找到ID对应节点。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。
UnsupportedOperation.InvalidNodeRole	类型节点不支持当前操作。
UnsupportedOperation.NodeStatusNotSupport	节点状态不支持此操作。

添加节点

最近更新时间：2025-05-14 02:04:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(AddNodes)用于添加一个或者多个计算节点或者登录节点到指定集群。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：AddNodes。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
Placement	是	Placement	集群中实例所在的位置。 示例值：ap-guangzhou-1
ClusterId	是	String	集群ID。 示例值：hpc-xxx
VirtualPrivateCloud	是	VirtualPrivateCloud	私有网络相关信息配置。 示例值：vpc-xxx
Count	是	Integer	添加节点数量。 示例值：1
ImageId	否	String	指定有效的镜像ID，格式形如img-xxx。目前支持部分公有镜像和自定义镜像。公共镜像请参考镜像限制 示例值：img-xxx
InstanceChargeType	否	String	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：PREPAID
InstanceChargePrepaid	否	InstanceChargePrepaid	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{ "Period": 36, "RenewFlag": "NOTIFY_AND_AUTO_RENEW" }
InstanceType	否	String	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S5.LARGE4
SystemDisk	否	SystemDisk	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值：{ "DiskSize": 50 }
DataDisks.N	否	Array of DataDisk	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值：[{ "DiskSize": 50, "DiskType": "CLOUD_PREMIUM" }]

参数名称	必选	类型	描述
InternetAccessible	否	InternetAccessible	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{ "PublicIpAssigned": "false" }
InstanceName	否	String	节点显示名称。 不指定节点显示名称则默认显示‘未命名’。 最多支持60个字符。 示例值：server_{R:3}
LoginSettings	否	LoginSettings	集群登录设置。 示例值：{ "KeyIds": ["skey-9tzxxb4j"] }
SecurityGroupIds.N	否	Array of String	集群中实例所属安全组。该参数可以通过调用 DescribeSecurityGroups 的返回值中的sgId字段来获取。若不指定该参数，则绑定默认安全组。 示例值：["sg-ajhn9qtq"]
ClientToken	否	String	用于保证请求幂等性的字符串。该字符串由客户生成，需保证不同请求之间唯一，最大值不超过64个ASCII字符。若不指定该参数，则无法保证请求的幂等性。 示例值：system-f3827db9-c58a-49cc-bf10-33fc1923a34a
QueueName	否	String	队列名称。不指定则为默认队列。 • SLURM默认队列为：compute。 示例值：compute
NodeRole	否	String	添加节点角色。默认值：Compute • Compute：计算节点。 • Login：登录节点。 示例值：Compute
DryRun	否	Boolean	是否只预检此次请求。 true：发送检查请求，不会创建实例。检查项包括是否填写了必需参数，请求格式，业务限制和云服务器库存。 如果检查不通过，则返回对应错误码； 如果检查通过，则返回RequestId. false（默认）：发送正常请求，通过检查后直接创建实例 示例值：False
NodeType	否	String	添加节点类型。默认取值：STATIC。 • STATIC：静态节点，不会参与弹性伸缩流程。 • DYNAMIC：弹性节点，会被弹性缩容的节点。管控节点和登录节点不支持此参数。 示例值：STATIC
ProjectId	否	Integer	实例所属项目ID。该参数可以通过调用 DescribeProject 的返回值中的 projectId 字段来获取。不填为默认项目。 示例值：0
ResourceType	否	String	要新增节点的资源类型。 • CVM：CVM实例类型资源 • WORKSPACE：工作空间类型实例资源 默认值：CVM。 示例值：CVM

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 添加节点

往集群ID为hpc-52nkfau6的集群的compute队列增加一个计算节点。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: AddNodes
<公共请求参数>

{
"Count": "1",
"VirtualPrivateCloud": {
"SubnetId": "subnet-r0zpktaa",
"VpcId": "vpc-r9jw2jzr"
},
"Placement": {
"Zone": "ap-guangzhou-2"
},
"ClusterId": "hpc-52nkfau6",
"ImageId": "img-3la7wgnt",
"InstanceChargeType": "SPOTPAID",
"InstanceType": "S2.SMALL2",
"NodeRole": "Compute",
"QueueName": "compute"
}
```

输出示例

```
{
"Response": {
"RequestId": "b2ac2379-6453-4eab-8f63-7ade00cb67b0"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
InternalError.CallCvm	cvm调用失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.ParametersNotSupported	字段不支持此值。
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooSmall	参数值过小。
MissingParameter	缺少参数错误。
ResourceInsufficient	资源不足。
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.ImageId	无法找到镜像ID。
ResourceNotFound.Queue	无法找到指定队列。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。
UnsupportedOperation.VpcIdConflict	vpc冲突，不支持当前操作。

修改节点初始化脚本

最近更新时间：2025-04-25 02:01:38

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (ModifyInitNodeScripts) 用于修改节点初始化脚本。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ModifyInitNodeScripts。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
InitNodeScripts.N	否	Array of NodeScript	节点初始化脚本信息列表。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 修改节点初始化脚本

修改节点初始化脚本。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: ModifyInitNodeScripts
<公共请求参数>

{
"ClusterId": "hpc-5lyv94lq",
"InitNodeScripts": [
{
"ScriptPath": "cos://test-xxxxxxxx.cos.ap-guangzhou.myqcloud.com/test/echo.sh",
"Timeout": 30
}
```

```
]
}
```

输出示例

```
{
"Response": {
"RequestId": "b40ac3ec-f0ac-4eaaa-8af8-ef2d93f37982"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameterValue.TooLarge	参数值过大。

查询节点初始化脚本列表

最近更新时间：2025-04-25 02:01:39

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (DescribeInitNodeScripts) 用于查询节点初始化脚本列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeInitNodeScripts。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq

3. 输出参数

参数名称	类型	描述
InitNodeScriptSet	Array of NodeScript	节点初始化脚本列表。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询节点初始化脚本信息

查询节点初始化脚本信息。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeInitNodeScripts
<公共请求参数>

{
"ClusterId": "hpc-5lyv94lq"
}
```

输出示例

```
{
"Response": {
"InitNodeScriptSet": [
{
"ScriptPath": "cos://test-xxxxxxxx.cos.ap-guangzhou.myqcloud.com/test/echo.sh",
"Timeout": 30
}
],
"RequestId": "a40ac3ec-f0ac-4eee-8af8-ef2d93f37982"
}
}
```


5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

从集群解绑节点

最近更新时间：2025-04-25 02:01:38

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (DetachNodes) 用于将一个或者多个计算节点从集群中移除，但是不销毁指定计算资源。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DetachNodes。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群id 示例值：hpc-2j8ntf9l
NodeIds.N	是	Array of String	集群中的节点id 示例值：["node-9workmoi"]

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 从集群解绑计算节点

集群ID为hpc-2j8ntf9l的集群解绑计算节点node-8workmoi

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: AttachNodes
<公共请求参数>

{
"ClusterId": "hpc-2j8ntf9l",
"NodeIds": [
"node-8workmoi"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "40c2e07e-8500-4460-8db8-666d0e791ee5"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

绑定计算资源到集群

最近更新时间：2025-05-14 02:04:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (AttachNodes) 用于绑定一个或者多个计算节点指定资源到指定集群中。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：AttachNodes。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群id 示例值：hpc-cy4pnyp9
ResourceSet.N	是	Array of String	节点的实例id列表 示例值：["ins-a437qz3d"]
QueueName	否	String	队列名称。不指定则为默认队列： SLURM默认队列为：compute。 示例值：compute
ImageId	否	String	指定有效的镜像ID，格式形如img-xxx。目前仅支持公有镜像和特定自定义镜像。如不指定，则该字段是默认镜像。 示例值：img-n7nyt2d7
ResourceType	否	String	要新增节点的资源类型。 • CVM：CVM实例类型资源 • WORKSPACE：工作空间类型实例资源 默认值：CVM。 示例值：CVM

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 绑定计算节点到集群

往集群ID为hpc-2j8ntf9l的集群的compute队列绑定一个计算节点。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: AttachNodes
<公共请求参数>

{
"ClusterId": "hpc-2j8ntf91",
"QueueName": "compute",
"ImageId": "img-39ywauzd",
"ResourceType": "CVM",
"ResourceSet": [
"ins-7i4yut4b"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "e81d404b-4fa2-49de-bce7-6499a3529f04"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

存储选项相关接口
查询集群存储选项

最近更新时间：2025-04-25 02:01:40

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (DescribeClusterStorageOption) 用于查询集群存储选项信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeClusterStorageOption。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq

3. 输出参数

参数名称	类型	描述
StorageOption	StorageOptionOverview	集群存储选项信息概览。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询集群存储选项

查询集群ID为hpc-5lyv94lq的集群存储选项信息。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeClusterStorageOption
<公共请求参数>

{
"ClusterId": "hpc-5lyv94lq"
}
```

输出示例

```
{
"Response": {
"StorageOption": {
"CFSOptions": [
{
"LocalPath": "/opt/",
"RemotePath": "172.30.3.90:/j25ey5tz/hpc-kjddsnfa/opt/",
"Protocol": "NFS 3.0",
"StorageType": "SD",
"MountOption": "vers=3,nolock,proto=tcp,noresvport"
}
],
"GooseFSOptions": [
{
"LocalPath": "/data/",
"RemotePath": "/xxx/cfs/",
"Masters": [
"172.30.4.63:9202",
"172.30.0.128:9202"
]
}
],
"GooseFSxOptions": [
{
"LocalPath": "/data/",
"Masters": [
"172.30.4.63:9202",
"172.30.0.128:9202"
]
}
]
},
"RequestId": "2aeaea7e-9fc4-4c17-8a9b-50343b712993"
}
}
```


5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
ResourceNotFound.ClusterId	集群不存在。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

删除集群存储选项

最近更新时间：2025-04-25 02:01:40

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (DeleteClusterStorageOption) 用于删除集群存储选项信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteClusterStorageOption。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
LocalPath	是	String	本地挂载路径。 示例值：/data/

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除集群存储选项信息。

删除本地挂载点为/data/的存储选项信息。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DeleteClusterStorageOption
<公共请求参数>

{
"ClusterId": "hpc-5lyv94lq",
"LocalPath": "/data/"
}
```

输出示例

```
{
"Response": {
"RequestId": "2aeaea7e-9fc4-4c17-8a9b-50343b711003"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.LocalPath	无法找到本地挂载路径。

添加集群存储选项

最近更新时间：2025-04-25 02:01:40

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（AddClusterStorageOption）用于添加集群存储选项信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：AddClusterStorageOption。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
StorageOption	是	StorageOption	集群存储选项；集群已存在的节点和新增节点都会挂载此存储。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 添加集群存储选项

集群添加CFS挂载选项。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: AddClusterStorageOption
<公共请求参数>

{
"ClusterId": "hpc-eq97tf4u",
"StorageOption": {
"CFSOptions": [
{
"LocalPath": "/data/",
"RemotePath": "172.30.3.90:/",
"Protocol": "NFS 4.0",
```

```
"StorageType": "SD"
},
{
"LocalPath": "/tmp/",
"RemotePath": "172.30.2.180@tcp0:/4fe1839b/cfs",
"Protocol": "TURBO",
"StorageType": "TB"
}
]
}
}
```

输出示例

```
{
"Response": {
"RequestId": "b2ac2379-6453-4eab-8f63-7ade00cb67b0"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.ValueDuplicated	参数值重复。不支持此操作。
ResourceNotFound.ClusterId	集群不存在。

集群相关接口
查询集群列表

最近更新时间：2025-04-25 02:01:37

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（DescribeClusters）用于查询集群列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeClusters。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterIds.N	否	Array of String	集群ID列表。通过该参数可以指定需要查询信息的集群列表。 如果您不指定该参数，则返回Limit数量以内的集群信息。 示例值：["hpc-xxx"]
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：20
Filters.N	否	Array of Filter	• cluster-type 按照【集群类型】进行过滤 类型：String 必选：否 每次请求的 Filters 的上限为10， Filter.Values 的上限为5。

3. 输出参数

参数名称	类型	描述
ClusterSet	Array of ClusterOverview	集群概览信息列表。 示例值：[{"ClusterId":"hpc-qrb7ybn0","ClusterName":"unnamed","ClusterStatus":"RUNNING"}]
TotalCount	Integer	集群数量。 示例值：1

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询集群列表

查询集群列表。

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DescribeClusters
&Offset=0
&Limit=1
&<公共请求参数>
```

输出示例


```
{
"Response": {
"ClusterSet": [
{
"ClusterId": "hpc-qrb7ybn0",
"ClusterName": "unnamed",
"ClusterStatus": "RUNNING",
"Placement": {
"Zone": "ap-guangzhou-2"
},
"CreateTime": "2021-12-07T03:29:09Z",
"SchedulerType": "SGE",
"VpcId": "vpc-xxxxxxxx",
"ComputeNodeCount": 1,
"LoginNodeSet": [
{
"NodeId": "ins-jehc407m"
}
],
"LoginNodeCount": 1,
"ComputeNodeSet": [
{
"NodeId": "ins-jfhc307q"
}
],
"ManagerNodeCount": 1,
"ManagerNodeSet": [
{
"NodeId": "ins-cv6ygpkq"
}
],
"AutoScalingType": "THPC_AS"
}
],
"TotalCount": 1,
"RequestId": "8a39c8e3-129b-4628-b763-ef96caaa2f4d"
```

```
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。

查询集群活动历史记录

最近更新时间：2025-04-25 02:01:38

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（DescribeClusterActivities）用于查询集群活动历史记录列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeClusterActivities。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。通过该参数指定需要查询活动历史记录的集群。 示例值：hpc-myd8fgsc
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：20

3. 输出参数

参数名称	类型	描述
ClusterActivitySet	Array of ClusterActivity	集群活动历史记录列表。
TotalCount	Integer	集群活动历史记录数量。 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询集群活动历史记录

根据集群ID查询集群活动历史记录。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeClusterActivities
<公共请求参数>
```

```
{
"ClusterId": "hpc-0yd8fqsc",
"Limit": "1",
"Offset": "0"
}
```

输出示例

```
{
"Response": {
"ClusterActivitySet": [
{
"ClusterId": "hpc-myd8fgsc",
"ActivityId": "cha-gvzj0zbd",
"ActivityType": "TerminateNodes",
"ActivityStatus": "SUCCESSFUL",
"ActivityStatusCode": "ActivitySuccess",
"ResultDetail": "Activity success.",
"Cause": "DeleteCluster",
"Description": "删除指定集群，销毁实例，销毁所有节点。",
"RelatedNodeActivitySet": [
{
"NodeInstanceId": "ins-1zll2of0",
"NodeActivityStatus": "SUCCESSFUL",
"NodeActivityStatusCode": "ActivitySuccess",
"NodeActivityStatusReason": "Activity success."
},
{
"NodeInstanceId": "ins-ig2bew40",
"NodeActivityStatus": "SUCCESSFUL",
"NodeActivityStatusCode": "ActivitySuccess",
"NodeActivityStatusReason": "Activity success."
}
],
"StartTime": "2021-11-01T02:17:20Z",
"EndTime": "2021-11-01T02:17:38Z"
}
],
"TotalCount": 1,
"RequestId": "7fa864e6-cf1a-4962-8aa9-f68abfa31a00"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee

- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.TooLarge	参数值过大。

删除集群

最近更新时间：2025-04-25 02:01:38

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（DeleteCluster）用于删除一个指定的集群。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteCluster。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除集群

删除指定集群。

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DeleteCluster
&ClusterId=hpc-5lyv94lq
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "8b71050a-55c1-4f3b-bf66-5e4f8510a5a0"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
ResourceNotFound.ClusterId	集群不存在。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

创建集群

最近更新时间：2025-05-14 02:04:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (CreateCluster) 用于创建并启动集群。

- 本接口为异步接口，当创建集群请求下发成功后会返回一个集群ID和一个RequestId，此时创建集群操作并未立即完成。在此期间集群的状态将会处于“PENDING”或者“INITING”，集群创建结果可以通过调用 DescribeClusters 接口查询，如果集群状态(ClusterStatus)变为“RUNNING(运行中)”，则代表集群创建成功，“INIT_FAILED”代表集群创建失败。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateCluster。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
Placement	是	Placement	集群中实例所在的位置。 示例值：{ "Zone": "ap-guangzhou-6" }
ManagerNode	否	ManagerNode	指定管理节点。 示例值：{"InstanceType": "S2.SMALL2"}
ManagerNodeCount	否	Integer	指定管理节点的数量。默认取值：1。取值范围：1~2。 示例值：1
ComputeNode	否	ComputeNode	指定计算节点。 示例值：{"InstanceType": "S2.SMALL2"}
ComputeNodeCount	否	Integer	指定计算节点的数量。默认取值：0。 示例值：1
SchedulerType	否	String	调度器类型。默认取值：SLURM。 • SLURM：SLURM调度器。 示例值：SLURM
SchedulerVersion	否	String	创建调度器的版本号，可填写版本号为“latest” 和 各调度器支持的版本号；如果是"latest", 则代表创该类型调度器最新版本。如果不填写，默认创建的是“latest”版本调度器 各调度器支持的集群版本： • SLURM：21.08.8、23.11.7 示例值：23.11.7
ImageId	否	String	指定有效的镜像ID，格式形如img-xxx。目前支持部分公有镜像和自定义镜像。公共镜像请参考镜像限制 示例值：img-xxx
VirtualPrivateCloud	否	VirtualPrivateCloud	私有网络相关信息配置。 示例值：{ "VpcId": "vpc-rhfaxx31", "SubnetId": "subnet-x7vxgqe" }
LoginSettings	否	LoginSettings	集群登录设置。 示例值：{ "KeyIds": ["skey-9tzxxb4j"] }

参数名称	必选	类型	描述
SecurityGroupIds.N	否	Array of String	集群中实例所属安全组。该参数可以通过调用 DescribeSecurityGroups 的返回值中的sgId字段来获 则绑定默认安全组。 示例值：["sg-ajhn9qtq"]
ClientToken	否	String	用于保证请求幂等性的字符串。该字符串由客户生成，需保证不同请求之间唯一，最大值不超过64个AS 数，则无法保证请求的幂等性。 示例值：system-f3827db9-c58a-49cc-bf10-33fc1923a34a
DryRun	否	Boolean	是否只预检此次请求。 true：发送检查请求，不会创建实例。检查项包括是否填写了必需参数，请求格式，业务限制和云服务器 如果检查不通过，则返回对应错误码； 如果检查通过，则返回RequestId. false（默认）：发送正常请求，通过检查后直接创建实例 示例值：False
AccountType	否	String	域名字服务类型。默认取值：NIS。 • NIS：NIS域名字服务。 示例值：NIS
ClusterName	否	String	集群显示名称。 示例值：cluster-test
StorageOption	否	StorageOption	集群存储选项 示例值：{"CFSOptions": [{"LocalPath":"/mnt/","RemotePath":"127.0.0.1@tcp0:/test/cfs/","Protocol":"TURBO","S
LoginNode	否	LoginNode	指定登录节点。 示例值：{"InstanceType": "S2.SMALL2"}
LoginNodeCount	否	Integer	指定登录节点的数量。默认取值：0。取值范围：0~10。 示例值：1
Tags.N	否	Array of Tag	创建集群时同时绑定的标签对说明。 示例值：[{"Key": "test", "Value": "test"}]
AutoScalingType	否	String	弹性伸缩类型。默认值：THPC_AS 示例值：THPC_AS
InitNodeScripts.N	否	Array of NodeScript	节点初始化脚本信息列表。 示例值：{"ScriptPath": "cos://demo-xxxxxxxx.cos.ap-guangzhou.myqcloud.com/test/e "Timeout": "30"}
HpcClusterId	否	String	高性能计算集群ID。若创建的实例为高性能计算实例，需指定实例放置的集群，否则不可指定。 示例值：hpc-xxx

3. 输出参数

参数名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-5lyv94lq
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建集群

创建一个管控节点和两个计算节点的集群。调度器为：SLURM。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
```

```
Content-Type: application/json
X-TC-Action: CreateCluster
<公共请求参数>

{
"ManagerNodeCount": "1",
"Placement": {
"Zone": "ap-guangzhou-2"
},
"SchedulerType": "SLURM",
"ImageId": "img-l8og963d",
"ComputeNode": {
"InstanceChargeType": "SPOTPAID",
"InstanceType": "S2.SMALL2"
},
"ComputeNodeCount": "2",
"ManagerNode": {
"InstanceType": "S2.SMALL2"
}
}
```

输出示例

```
{
"Response": {
"ClusterId": "hpc-5lyv94lq",
"RequestId": "b2ac2379-6453-4eab-8f63-7ade00cb67b0"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.CallCAM	CAM服务调用失败。
InternalError.CallCvm	cvm调用失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.ParametersNotSupported	字段不支持此值。
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooShort	无效参数值。参数值太短。
InvalidParameterValue.TooSmall	参数值过小。
ResourceNotFound.ImageId	无法找到镜像ID。

弹性伸缩相关接口
设置弹性伸缩配置信息

最近更新时间：2025-04-25 02:01:39

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(SetAutoScalingConfiguration)用于为集群设置集群弹性伸缩配置信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：SetAutoScalingConfiguration。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
ExpansionBusyTime	否	Integer	任务连续等待时间，队列的任务处于连续等待的时间。单位秒。默认值120。 示例值：120
ShrinkIdleTime	否	Integer	节点连续空闲（未运行作业）时间，一个节点连续处于空闲状态时间。单位秒。默认值300。 示例值：300
QueueConfigs.N	否	Array of QueueConfig	扩容队列配置列表。 示例值：{"QueueName":"compute"}
DryRun	否	Boolean	是否只预检此次请求。 true：发送检查请求，不会绑定弹性伸缩组。检查项包括是否填写了必需参数，请求格式，业务限制。 如果检查不通过，则返回对应错误码； 如果检查通过，则返回RequestId。 false（默认）：发送正常请求，通过检查后直接绑定弹性伸缩组。 示例值：False

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 设置弹性伸缩配置

集群配置弹性伸缩策略信息。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: SetAutoScalingConfiguration
<公共请求参数>

{
"DryRun": "false",
"ShrinkIdleTime": "300",
"ExpansionBusyTime": "120",
"ClusterId": "hpc-5lyv94lq"
}
```

输出示例

```
{
"Response": {
"RequestId": "d1d90874-f565-463d-ba1d-c707517eece1"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.CallCvm	cvm调用失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.LimitExceeded	参数值数量超过限制。
InvalidParameterValue.TooLarge	参数值过大。

错误码	描述
InvalidParameterValue.TooSmall	参数值过小。
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.ImageId	无法找到镜像ID。
ResourceNotFound.Queue	无法找到指定队列。
UnsupportedOperation.AutoScalingType	弹性伸缩类型不支持此操作。
UnsupportedOperation.ClusterAcceptOtherRequest	集群资源正在处理其他请求。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。
UnsupportedOperation.ParameterTooLarge	参数值过大，不支持此操作。
UnsupportedOperation.VpcIdConflict	vpc冲突，不支持当前操作。

查询弹性伸缩配置信息

最近更新时间：2025-04-25 02:01:39

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(DescribeAutoScalingConfiguration)用于查询集群弹性伸缩配置信息。本接口仅适用于弹性伸缩类型为THPC_AS的集群。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeAutoScalingConfiguration。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq

3. 输出参数

参数名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-rv7hahw2
ExpansionBusyTime	Integer	任务连续等待时间，队列的任务处于连续等待的时间。单位秒。 示例值：120
ShrinkIdleTime	Integer	节点连续空闲（未运行作业）时间，一个节点连续处于空闲状态时间。 示例值：300
QueueConfigs	Array of QueueConfigOverview	扩容队列配置概览列表。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询扩缩容策略配置

查询集群ID为hpc-rv7hahw2的集群弹性伸缩配置信息。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeAutoScalingConfiguration
<公共请求参数>
```

```
{
"ClusterId": "hpc-rv7hahw2"
}
```

输出示例

```
{
"Response": {
"ClusterId": "hpc-rv7hahw2",
"ExpansionBusyTime": 120,
"ShrinkIdleTime": 300,
"QueueConfigs": [
{
"QueueName": "compute",
"MinSize": 0,
"MaxSize": 10,
"EnableAutoExpansion": false,
"EnableAutoShrink": false,
"ExpansionNodeConfigs": [
{
"InstanceType": "M5.SMALL8",
"Placement": {
"Zone": "ap-chongqing-1"
},
"InstanceChargeType": "POSTPAID_BY_HOUR",
"InstanceChargePrepaid": null,
"VirtualPrivateCloud": {
"VpcId": "vpc-r9jw2jzv",
"SubnetId": "subnet-r0zpktae"
},
"ImageId": "img-l8og963d",
"InternetAccessible": {
"InternetChargeType": "TRAFFIC_POSTPAID_BY_HOUR",
"InternetMaxBandwidthOut": 0
},
"SystemDisk": {
"DiskType": "CLOUD_BSSD",
"DiskSize": 50
},
"DataDisks": null
}
]
}
],
"RequestId": "935760b6-2a63-480d-9124-c5f5b9d4218b"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
ResourceNotFound.ClusterId	集群不存在。
UnsupportedOperation.AutoScalingType	弹性伸缩类型不支持此操作。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

队列相关接口
查询队列列表

最近更新时间：2025-04-25 02:01:38

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(DescribeQueues)用于查询指定集群队列概览信息列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeQueues。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：20

3. 输出参数

参数名称	类型	描述
QueueSet	Array of QueueOverview	队列概览信息列表。
TotalCount	Integer	符合条件的队列数量。 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询集群队列概览信息列表

查询集群队列概览信息列表。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeQueues
```

```
<公共请求参数>

{
"ClusterId": "hpc-prkxbd7c",
"Offset": 0,
"Limit": 20
}
```

输出示例

```
{
"Response": {
"QueueSet": [
{
"QueueName": "compute"
}
],
"TotalCount": 1,
"RequestId": "13ee84a5-7846-4682-8877-1c8f94962dd5"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.Malformed	参数格式有误。

错误码	描述
InvalidParameterValue	参数取值错误。
InvalidParameterValue.ParametersNotSupported	字段不支持此值。
InvalidParameterValue.TooShort	无效参数值。参数值太短。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceNotFound	资源不存在。
UnknownParameter	未知参数错误。
UnsupportedOperation.ParameterTooLarge	参数值过大，不支持此操作。
UnsupportedOperation.ParameterTooSmall	参数值过小，不支持此操作。

删除队列

最近更新时间：2025-04-25 02:01:38

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(DeleteQueue)用于从指定集群删除队列。

- 本接口为目前只支持SchedulerType为SLURM的集群。
- 删除队列时，需要保证队列内不存在节点。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteQueue。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-prkxbd7c
QueueName	是	String	队列名称。 最多支持32个字符。 示例值：compute

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除队列

从集群删除指定队列。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DeleteQueue
<公共请求参数>

{
"ClusterId": "hpc-prkxbd7c",
```

```
"QueueName": "compute"
}
```

输出示例

```
{
"Response": {
"RequestId": "13ee84a5-7846-4682-8877-1c8f94962d10"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.LimitExceeded	参数值数量超过限制。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.ParametersNotSupported	字段不支持此值。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooShort	无效参数值。参数值太短。
InvalidParameterValue.TooSmall	参数值过小。

错误码	描述
InvalidParameterValue.ValueDuplicated	参数值重复。不支持此操作。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
ResourceInUse	资源被占用。
ResourceNotFound	资源不存在。
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.Queue	无法找到指定队列。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。
UnsupportedOperation.QueueNotEmpty	队列内存在节点，不支持此操作。

添加队列

最近更新时间：2025-05-12 02:04:00

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(AddQueue)用于添加队列到指定集群。

- 本接口为目前只支持SchedulerType为SLURM的集群。
- 单个集群中队列数量上限为10个。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：AddQueue。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-prkxbd7c
QueueName	是	String	队列名称。 • 最多支持32个字符。 示例值：compute

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 添加队列

往指定集群添加队列。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: AddQueue
<公共请求参数>

{
"ClusterId": "hpc-prkxbd7c",
```

```
"QueueName": "gpu"
}
```

输出示例

```
{
"Response": {
"RequestId": "13ee84a5-7846-4682-8877-1c8f94962dd6"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
FailedOperation	操作失败。
InternalError.AgentRunScriptFail	agent执行脚本失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooSmall	参数值过小。
InvalidParameterValue.ValueDuplicated	参数值重复。不支持此操作。
LimitExceeded	超过配额限制。
LimitExceeded.QueueNumLimit	队列数量达到上限。
OperationDenied	操作被拒绝。

错误码	描述
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceNotFound	资源不存在。
ResourceNotFound.ClusterId	集群不存在。
UnsupportedOperation	操作不支持。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

工作空间相关接口
创建工作空间

最近更新时间：2025-05-21 01:32:56

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (CreateWorkspaces) 用于创建工作空间。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateWorkspaces。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表，本接口仅支持其中的: ap-beijing, ap-guangzhou, ap-nanjing, ap-shanghai 。
ClientToken	否	String	用于保证请求幂等性的字符串。该字符串由客户生成，需保证不同请求之间唯一，最大值不超过64个ASCII字符。若不指定该参数，则无法保证请求的幂等性。 示例值：system-f3827db9-c58a-49cc-bf10-33fc1923a34a
Placement	否	SpacePlacement	实例所在的位置。通过该参数可以指定实例所属可用区，所属项目，所属宿主机（在专用宿主机上创建子机时指定）等属性。 **注：如果您不指定LaunchTemplate参数，则Placement为必选参数。若同时传递Placement和LaunchTemplate，则默认覆盖LaunchTemplate中对应的Placement的值。** 示例值：{ "Zone": "ap-guangzhou-6" }
SpaceChargePrepaid	否	SpaceChargePrepaid	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月实例的购买时长、是否设置自动续费等属性。若指定实例的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
SpaceChargeType	否	String	工作空间计费类型，包括：PREPAID，UNDERWRITE。工作空间计费类型，包括：PREPAID，UNDERWRITE。 示例值：PREPAID
SpaceType	否	String	工作空间规格 示例值：96A.96XLARGE2304
ImageId	否	String	镜像ID 示例值：img-irmer45l
SystemDisk	否	SpaceSystemDisk	工作空间系统盘信息 示例值： {"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}
DataDisks.N	否	Array of SpaceDataDisk	工作空间数据盘信息 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
VirtualPrivateCloud	否	SpaceVirtualPrivateCloud	私有网络相关信息 示例值：{"VpcId":"vpc-nrqwfyzv","SubnetId":"subnet-0tmm4ywk"}
InternetAccessible	否	SpaceInternetAccessible	公网带宽相关信息设置 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}

参数名称	必选	类型	描述
SpaceCount	否	Integer	购买工作空间数量 示例值：1
SpaceName	否	String	工作空间显示名称 示例值：workspace
LoginSettings	否	LoginSettings	工作空间登陆设置 示例值：{ "KeyIds": ["skey-9tzxxb4j"] }
SecurityGroupIds.N	否	Array of String	工作空间所属安全组 示例值：["sg-ajhn9qtq"]
EnhancedService	否	EnhancedService	增强服务 示例值：{"SecurityService":{"Enabled":true},"MonitorService":{"Enabled":true},"AutomationService":{"Enabled":true}}
DryRun	否	Boolean	是否只预检此次请求 示例值：false
UserData	否	String	提供给工作空间使用的用户数据 示例值：TXlVc2VyRGF0YQo=
DisasterRecoverGroupIds.N	否	Array of String	置放群组id 示例值：['ps-3p88qhfo']
TagSpecification.N	否	Array of TagSpecification	标签描述列表 示例值：[{"Tags":[{"Key":"ENV","Value":"product"}]}]
HpcClusterId	否	String	高性能计算集群ID 示例值：hpc-a5n666lo
CamRoleName	否	String	CAM角色名称 示例值：testroleName001
HostName	否	String	实例主机名。 • 点号（.）和短横线（-）不能作为 HostName 的首尾字符，不能连续使用。 • Windows 实例：主机名名字符长度为[2, 15]，允许字母（不限制大小写）、数字和短横线（-）组成，不支持点号（.），不能全是数字。 • 其他类型（Linux 等）实例：主机名字符长度为[2, 60]，允许支持多个点号，点之间为一段，每段允许字母（不限制大小写）、数字和短横线（-）组成。 • 购买多台实例，如果指定模式串`{R:x}`，表示生成数字`[x, x+n-1]`，其中`n`表示购买实例的数量，例如`server{R:3}`，购买1台时，实例主机名为`server3`；购买2台时，实例主机名分别为`server3`，`server4`。支持指定多个模式串`{R:x}`。 • 购买多台实例，如果不指定模式串，则在实例主机名添加后缀`1、2...n`，其中`n`表示购买实例的数量，例如`server`，购买2台时，实例主机名分别为`server1`，`server2`。 示例值：MyHostName

3. 输出参数

参数名称	类型	描述
SpaceIdSet	Array of String	工作空间ID 示例值：['wks-rn99mzt1']
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建工作空间

创建工作空间

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: CreateWorkspaces
<公共请求参数>

{
"SpaceChargeType": "PREPAID",
"SpaceChargePrepaid": {
"Period": 1,
"RenewFlag": "NOTIFY_AND_AUTO_RENEW"
},
"Placement": {
"Zone": "ap-guangzhou-2",
"ProjectId": 0
},
"VirtualPrivateCloud": {
"AsVpcGateway": false,
"VpcId": "vpc-nrqwfyzv",
"SubnetId": "subnet-0tmm4ywk",
"Ipv6AddressCount": 0
},
"SpaceType": "96A.96XLARGE2304",
"ImageId": "img-9qrfy1xt",
"SystemDisk": {
"DiskSize": 50,
"DiskType": "CLOUD_PREMIUM"
},
"DataDisks": [
{
"DiskSize": 50,
"DiskType": "CLOUD_PREMIUM"
},
{
"DiskSize": 50,
"DiskType": "CLOUD_PREMIUM"
}
],
"InternetAccessible": {
"PublicIpAssigned": true,
"InternetMaxBandwidthOut": 5
},
"SecurityGroupIds": [
"sg-fsx9rsr1"
],
"SpaceCount": 1,
"EnhancedService": {
"SecurityService": {
"Enabled": true
},
"MonitorService": {
"Enabled": true
```

```
},
"AutomationService": {
"Enabled": true
}
}
}
```

输出示例

```
{
"Response": {
"SpaceIdSet": [
"wks-avs9gqvt"
],
"RequestId": "efbbe00e-0175-4ee7-92c5-debc5763142d"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

销毁工作空间

最近更新时间：2025-05-21 01:32:55

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (TerminateWorkspaces) 用于主动退还工作空间。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：TerminateWorkspaces。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表，本接口仅支持其中的: ap-beijing, ap-guangzhou, ap-nanjing, ap-shanghai 。
SpaceIds.N	是	Array of String	工作空间ID 示例值：['wks-qk5u85i8']
ReleasePrepaidDataDisks	否	Boolean	释放空间挂载的包年包月数据盘。true表示销毁空间同时释放包年包月数据盘，false表示只销毁空间。 示例值：true

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 主动退还工作空间

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: TerminateWorkspaces
<公共请求参数>

{
"SpaceIds": [
"wks-kj2pb4dt"
]
}
```

输出示例

```
{
"Response": {
"RequestId": "952f8cb8-240a-45cb-aabf-7665be657072"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

修改工作空间的属性

最近更新时间：2025-05-21 01:32:55

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (ModifyWorkspacesAttribute) 用于修改工作空间的属性（目前只支持修改工作空间的名称）。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ModifyWorkspacesAttribute。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表，本接口仅支持其中的: ap-beijing, ap-guangzhou, ap-nanjing, ap-shanghai 。
SpaceIds.N	是	Array of String	工作空间列表 示例值：["wks-rf6ogz49"]
SpaceName	否	String	修改后的工作空间名称。可任意命名，但不得超过60个字符。 示例值：MySpaceName

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 修改工作空间的名称

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: ModifyWorkspacesAttribute
<公共请求参数>

{
"SpaceIds": [
"wks-gzut2d4n"
],
"SpaceName": "Workspace"
}
```

输出示例

```
{
"Response": {
"RequestId": "dbfa8866-7644-4181-806f-2334ea2da591"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。

查询工作空间列表

最近更新时间：2025-05-20 01:31:37

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（DescribeWorkspaces）用于查询工作空间列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeWorkspaces。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表，本接口仅支持其中的: ap-beijing, ap-guangzhou, ap-nanjing, ap-shanghai 。
SpaceIds.N	否	Array of String	集群ID列表。通过该参数可以指定需要查询信息的集群列表。 如果您不指定该参数，则返回Limit数量以内的集群信息。 示例值：["wks-xxx"]
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：20
Filters.N	否	Array of Filter	• zone 按照【可用区】进行过滤 类型：String 必选：否 • space-id 按照【工作空间实例ID】进行过滤 类型：String 必选：否 • cvm-instance-id 按照【CVM实例ID】进行过滤 类型：String 必选：否 • space-state 按照【工作空间状态】进行过滤 类型：String 必选：否 • space-name

参数名称	必选	类型	描述
			按照【工作空间别名】进行过滤 类型：String 必选：否 • space-charge-type 按照【工作空间实例付费模式】进行过滤 类型：String 必选：否 • tag-key 按照【标签键】进行过滤 类型：String 必选：否 • tag-value 按照【标签值】进行过滤 类型：String 必选：否 每次请求的 Filters 的上限为10， Filter.Values 的上限为5。 示例值：[{"Name":"space-id","Value":["wks-xxx"]}]

3. 输出参数

参数名称	类型	描述
SpaceSet	Array of SpaceInfo	集群概览信息列表
TotalCount	Integer	集群数量 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询工作空间列表

查看在广州二区的实例信息,限制返回结果最多为一项

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeWorkspaces
<公共请求参数>

{
"Limit": "1",
"Filters": [
{
"Values": [
"ap-guangzhou-2"
```

```
],
"Name": "zone"
}
],
"Offset": "0"
}
```

输出示例


```
{
"Response": {
"RequestId": "0142a337-d8cf-4ae1-857f-d090c1f802fdafd",
"SpaceSet": [
{
"CreatedTime": "2024-09-22T00:00:00+00:00",
"ExpiredTime": "2024-09-22T00:00:00+00:00",
"LatestOperation": "CreateWorkspaces",
"LatestOperationState": "SUCCESS",
"Placement": {
"Zone": "ap-shanghai-8"
},
"RenewFlag": "NOTIFY_AND_MANUAL_RENEW",
"ResourceId": "ins-3zjugd6z",
"SpaceChargeType": "PREPAID",
"SpaceFamily": "96AS",
"SpaceId": "wks-bfdafadfd",
"SpaceName": "workspace",
"SpaceState": "ONLINE",
"SpaceType": "96AS.4XLARGE160",
"Tags": []
}
],
"TotalCount": 1
}
}
```


5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.AtMostOne	参数互斥，最多只能传入一个参数
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.InvalidSpaceIdMalformed	工作空间实例ID格式不符合规范。

修改工作空间的续费标识

最近更新时间：2025-05-21 01:32:55

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (ModifyWorkspacesAttribute) 用于修改工作空间的属性（目前只支持修改工作空间的名称）。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：ModifyWorkspacesRenewFlag。
Version	是	String	公共参数，本接口取值：2023-03-21。
Region	是	String	公共参数，详见产品支持的 地域列表，本接口仅支持其中的: ap-beijing, ap-guangzhou, ap-nanjing, ap-shanghai 。
SpaceIds.N	是	Array of String	工作空间列表 示例值：["wks-rf6ogz49"]
RenewFlag	是	String	自动续费标识。取值范围： • NOTIFY_AND_AUTO_RENEW：通知过期且自动续费 • NOTIFY_AND_MANUAL_RENEW：通知过期不自动续费 • DISABLE_NOTIFY_AND_MANUAL_RENEW：不通知过期不自动续费 若该参数指定为NOTIFY_AND_AUTO_RENEW，在账户余额充足的情况下，实例到期后将按月自动续费。 示例值：NOTIFY_AND_AUTO_RENEW

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 修改工作空间实例的续费标识

修改工作空间实例的续费标识

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: ModifyWorkspacesRenewFlag
<公共请求参数>

{
"SpaceIds": [
```

```
"wks-xxxx"
],
"RenewFlag": "NOTIFY_AND_AUTO_RENEW"
}
```

输出示例

```
{
"Response": {
"RequestId": "84dea32c-55a7-4a07-b2e9-f7ba4dfaad58"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.InvalidSpaceIdMalformed	工作空间实例ID格式不符合规范。
InvalidParameterValue.SpaceIdNotFound	工作空间实例查找失败
InvalidParameterValue.ValueDuplicated	参数值重复。不支持此操作。
UnsupportedOperation.SpaceChargeType	实例的付费模式不支持当前操作。
UnsupportedOperation.WorkspaceStateArrears	隔离状态的工作空间实例不支持当前操作。

数据结构

最近更新时间：2025-05-14 02:04:32

CFSOption

描述CFS文件系统版本和挂载信息

被如下接口引用：AddClusterStorageOption, CreateCluster。

名称	类型	必选	描述
LocalPath	String	是	文件系统本地挂载路径。 示例值：/data/
RemotePath	String	是	文件系统远程挂载ip及路径。 示例值：172.0.0.1:/
Protocol	String	否	文件系统协议类型，默认值NFS 3.0。 • NFS 3.0。 • NFS 4.0。 • TURBO。 示例值：NFS 3.0
StorageType	String	否	文件系统存储类型，默认值SD；其中 SD 为通用标准型标准型存储， HP为通用性能型存储， TB为turbo标准型， TP 为turbo性能型。 示例值：SD
MountOption	String	否	文件系统挂载挂载命令参数选项。 – NFS 3.0默认值：vers=3,nolock,proto=tcp,noresvport – NFS 4.0默认值：vers=4.0,noresvport – TURBO默认值：user_xattr 示例值：vers=4.0,noresvport

CFSOptionOverview

CFS存储选项概览信息。

被如下接口引用：DescribeClusterStorageOption。

名称	类型	描述
LocalPath	String	文件系统本地挂载路径。 示例值：/data/
RemotePath	String	文件系统远程挂载ip及路径。 示例值：172.0.0.1:/
Protocol	String	文件系统协议类型。 • NFS 3.0。 • NFS 4.0。 • TURBO。 示例值：NFS 3.0
StorageType	String	文件系统存储类型，默认值SD；其中 SD 为通用标准型标准型存储， HP为通用性能型存储， TB为turbo标准型， TP 为turbo性能型。 示例值：SD
MountOption	String	文件系统挂载命令参数选项。 示例值：vers=3,nolock,proto=tcp,noresvport

ClusterActivity

符合条件的集群活动信息。

被如下接口引用：DescribeClusterActivities。

名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-myd8fgsc
ActivityId	String	集群活动ID。 示例值：cha-gvzj0zbd
ActivityType	String	集群活动类型。取值范围： • CreateAndAddNodes：创建实例并添加进集群 • RemoveNodesFromCluster：从集群移除实例 • TerminateNodes：销毁实例 • MountStorageOption：增加挂载选项并进行挂载 • UmountStorageOption：删除集群挂载存储选项并解挂载 示例值：TerminateNodes
ActivityStatus	String	集群活动状态。取值范围： • PENDING：等待运行 • RUNNING：运行中 • SUCCESSFUL：活动成功 • PARTIALLY_SUCCESSFUL：活动部分成功 • FAILED：活动失败 示例值：SUCCESSFUL
ActivityStatusCode	String	集群活动状态码。 示例值：ActivitySuccess
ResultDetail	String	集群活动结果详情。 注意：此字段可能返回 null，表示取不到有效值。 示例值：Activity success.
Cause	String	集群活动起因。 示例值：DeleteCluster
Description	String	集群活动描述。 示例值：删除指定集群，销毁实例，销毁所有节点。
RelatedNodeActivitySet	Array of NodeActivity	集群活动相关节点活动集合。 示例值：[{"NodeInstanceId":"ins-1zll2of0","NodeActivityStatus":"SUCCESSFUL"}]
StartTime	Timestamp ISO8601	集群活动开始时间。 示例值：2021-11-01T02:17:20Z
EndTime	Timestamp ISO8601	集群活动结束时间。 示例值：2021-11-01T02:17:38Z

ClusterOverview

集群概览信息。

被如下接口引用：DescribeClusters。

名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-2mv1i3d8
ClusterStatus	String	集群状态。取值范围： • PENDING：创建中 • INITING：初始化中

名称	类型	描述
		• INIT_FAILED：初始化失败 • RUNNING：运行中 • TERMINATING：销毁中 示例值：RUNNING
ClusterName	String	集群名称。 示例值：hpc-test
Placement	Placement	集群位置信息。 示例值：{ "Zone": "ap-guangzhou-6" }
CreateTime	Timestamp ISO8601	集群创建时间。 示例值：2024-12-18T13:46:34Z
SchedulerType	String	集群调度器。 示例值：SLURM
SchedulerVersion	String	集群调度器版本。 示例值：21.08.8
ComputeNodeCount	Integer	计算节点数量。 示例值：1
ComputeNodeSet	Array of ComputeNodeOverview	计算节点概览。 示例值：[{"NodeId":"ins-riredadfj6"},{"NodeId":"ins-mdafdsai"},{"NodeId":"ins-5icofadac"},{"NodeId":"ins-icfdadfy"}]
ManagerNodeCount	Integer	管控节点数量。 示例值：1
ManagerNodeSet	Array of ManagerNodeOverview	管控节点概览。 示例值：[{"NodeId":"node-rirfdafa"}]
LoginNodeSet	Array of LoginNodeOverview	登录节点概览。 示例值：[{"NodeId":"ins-riredadfj6"},{"NodeId":"ins-mdafdsai"},{"NodeId":"ins-5icofadac"},{"NodeId":"ins-icfdadfy"}]
LoginNodeCount	Integer	登录节点数量。 示例值：1
AutoScalingType	String	弹性伸缩类型。 示例值：THPC_AS
VpcId	String	集群所属私有网络ID。 示例值：vpc-r9fr2jzy
ClusterType	String	集群类型 示例值：GeneralComputing

ComputeNode

计算节点信息。

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}

名称	类型	必选	描述
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
SystemDisk	SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值： {"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
InternetAccessible	InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}
InstanceName	String	否	节点显示名称。 • 不指定节点显示名称则默认显示‘未命名’。 最多支持60个字符。 示例值：未命名
ProjectId	Integer	否	实例所属项目ID。该参数可以通过调用 DescribeProject 的返回值中的 projectId 字段来获取。不填为默认项目。 示例值：0
ResourceType	String	否	实例资源类型，默认是CVM资源 示例值：CVM

ComputeNodeOverview

计算节点概览。

被如下接口引用：DescribeClusters。

名称	类型	描述
NodeId	String	计算节点ID。 示例值：ins-jfhc307q

DataDisk

描述了数据盘的信息

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
DiskSize	Integer	是	数据盘大小，单位：GB。最小调整步长为10G，不同数据盘类型取值范围不同，具体限制详见：存储概述。默认值为0，表示不购买数据盘。更多限制详见产品文档。 注意：此字段可能返回 null，表示取不到有效值。 示例值：0
DiskType	String	否	数据盘类型。数据盘类型限制详见存储概述。取值范围： • LOCAL_NVME：本地NVME硬盘，与InstanceType强相关，不支持指定 • LOCAL_PRO：本地HDD硬盘，与InstanceType强相关，不支持指定 • CLOUD_BASIC：普通云硬盘 • CLOUD_PREMIUM：高性能云硬盘 • CLOUD_SSD：SSD云硬盘 • CLOUD_HSSD：增强型SSD云硬盘 • CLOUD_TSSD：极速型SSD云硬盘

名称	类型	必选	描述
			• CLOUD_BSSD：通用型SSD云硬盘 注意：此字段可能返回 null，表示取不到有效值。 示例值：CLOUD_BASIC

EnhancedService

描述了实例的增强服务启用情况与其设置，如云安全，腾讯云可观测平台等实例 Agent

被如下接口引用：CreateCluster, CreateWorkspaces, SetAutoScalingConfiguration。

名称	类型	必选	描述
SecurityService	RunSecurityServiceEnabled	否	开启云安全服务。若不指定该参数，则默认开启云安全服务。 示例值：TRUE
MonitorService	RunMonitorServiceEnabled	否	开启腾讯云可观测平台服务。若不指定该参数，则默认开启腾讯云可观测平台服务。 示例值：TRUE
AutomationService	RunAutomationServiceEnabled	否	开启云自动化助手服务（TencentCloud Automation Tools，TAT）。若不指定该参数，默认开启云自动化助手服务。 示例值：TRUE

ExpansionNodeConfig

弹性扩容节点配置信息。

被如下接口引用：SetAutoScalingConfiguration。

名称	类型	必选	描述
Placement	Placement	是	扩容实例所在的位置。 示例值：{"Zone": "ap-guangzhou-1"}
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
VirtualPrivateCloud	VirtualPrivateCloud	否	私有网络相关信息配置。 示例值：{"VpcId": "vpc-xxx", "SubnetId": "subnet-xxx"}
ProjectId	Integer	否	实例所属项目ID。该参数可以通过调用 DescribeProject 的返回值中的 projectId 字段来获取。不填为默认项目。 示例值：0

ExpansionNodeConfigOverview

扩容节点配置信息概览。

被如下接口引用：DescribeAutoScalingConfiguration。

名称	类型	描述
InstanceType	String	节点机型。 注意：此字段可能返回 null，表示取不到有效值。 示例值：S2.SMALL2
Placement	Placement	扩容实例所在的位置。 注意：此字段可能返回 null，表示取不到有效值。
InstanceChargeType	String	节点计费类型。 注意：此字段可能返回 null，表示取不到有效值。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 注意：此字段可能返回 null，表示取不到有效值。
VirtualPrivateCloud	VirtualPrivateCloud	私有网络相关信息配置。 注意：此字段可能返回 null，表示取不到有效值。
ImageId	String	指定有效的镜像ID，格式形如`img-xxx`。 注意：此字段可能返回 null，表示取不到有效值。 示例值：img-3la7wgnt
InternetAccessible	InternetAccessible	公网带宽相关信息设置。 注意：此字段可能返回 null，表示取不到有效值。
SystemDisk	SystemDisk	节点系统盘配置信息。 注意：此字段可能返回 null，表示取不到有效值。
DataDisks	Array of DataDisk	节点数据盘配置信息。 注意：此字段可能返回 null，表示取不到有效值。

Filter

描述键值对过滤器，用于条件过滤查询。例如过滤ID、名称、状态等

- 若存在多个`Filter`时，`Filter`间的关系为逻辑与（`AND`）关系。
- 若同一个`Filter`存在多个`Values`，同一`Filter`下`Values`间的关系为逻辑或（`OR`）关系。

被如下接口引用：DescribeClusters, DescribeNodes, DescribeWorkspaces。

名称	类型	必选	描述
Name	String	是	需要过滤的字段。 示例值：fiter-name
Values	Array of String	是	字段的过滤值。 示例值：fiter-value

GooseFSOption

描述GooseFS挂载信息

被如下接口引用：AddClusterStorageOption, CreateCluster。

名称	类型	必选	描述
LocalPath	String	是	文件系统本地挂载路径。 示例值：/data
RemotePath	String	是	文件系统远程挂载路径。 示例值：/data/goosefs-fuse
Masters	Array of String	是	文件系统master的ip和端口。 示例值：["172.16.0.2:55533","172.16.0.3:55533","172.16.0.4:55533"]

GooseFSOptionOverview

GooseFS存储选项概览信息。

被如下接口引用：DescribeClusterStorageOption。

名称	类型	描述
LocalPath	String	文件系统本地挂载路径。 示例值：/data/
RemotePath	String	文件系统远程挂载路径。 示例值：/data/goosefs-fuse/
Masters	Array of String	文件系统master的ip和端口。 示例值：["172.16.0.2:55533","172.16.0.3:55533","172.16.0.4:55533"]

GooseFSxOption

描述GooseFSx挂载信息

被如下接口引用：AddClusterStorageOption, CreateCluster。

名称	类型	必选	描述
Masters	Array of String	是	文件系统master的ip和端口列表。 示例值：["172.16.0.2:55533","172.16.0.3:55533","172.16.0.4:55533"]
LocalPath	String	是	文件系统的本地挂载路径。GooseFSx目前只支持挂载在/goosefsx/{文件系统ID}_proxy/目录下。 示例值：/goosefsx/x_c60_54oi5id3_proxy/

GooseFSxOptionOverview

GooseFSx存储选项概览信息。

被如下接口引用：DescribeClusterStorageOption。

名称	类型	描述
Masters	Array of String	文件系统master的ip和端口列表。 示例值：["172.16.0.2:55533","172.16.0.3:55533","172.16.0.4:55533"]
LocalPath	String	文件系统的本地挂载路径。GooseFSx目前只支持挂载在/goosefsx/{文件系统ID}_proxy/目录下。 示例值：/goosefsx/x_c60_54oi5id3_proxy/

InstanceChargePrepaid

描述了实例的计费模式

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
Period	Integer	是	购买实例的时长，单位：月。取值范围：1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 36, 48, 60。 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
RenewFlag	String	否	自动续费标识。取值范围： NOTIFY_AND_AUTO_RENEW：通知过期且自动续费 NOTIFY_AND_MANUAL_RENEW：通知过期不自动续费 DISABLE_NOTIFY_AND_MANUAL_RENEW：不通知过期不自动续费 默认取值：NOTIFY_AND_MANUAL_RENEW。若该参数指定为NOTIFY_AND_AUTO_RENEW，在账户余额充足的情况下，实例到期后将按月自动续费。 注意：此字段可能返回 null，表示取不到有效值。 示例值：NOTIFY_AND_AUTO_RENEW

InternetAccessible

描述了实例的公网可访问性，声明了实例的公网使用计费模式，最大带宽等

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
InternetChargeType	String	否	网络计费类型。取值范围： BANDWIDTH_PREPAID：预付费按带宽结算 TRAFFIC_POSTPAID_BY_HOUR：流量按小时后付费 BANDWIDTH_POSTPAID_BY_HOUR：带宽按小时后付费 BANDWIDTH_PACKAGE：带宽包用户 默认取值：非带宽包用户默认与子机付费类型保持一致。 注意：此字段可能返回 null，表示取不到有效值。 示例值：TRAFFIC_POSTPAID_BY_HOUR
InternetMaxBandwidthOut	Integer	否	公网出带宽上限，单位：Mbps。默认值：0Mbps。不同机型带宽上限范围不一致，具体限制详见购买网络带宽。 注意：此字段可能返回 null，表示取不到有效值。 示例值：0

LoginNode

登录节点信息。

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
SystemDisk	SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值： {"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
InternetAccessible	InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}
InstanceName	String	否	节点显示名称。 • 不指定节点显示名称则默认显示‘未命名’。 最多支持60个字符。 示例值：未命名

名称	类型	必选	描述
ProjectId	Integer	否	实例所属项目ID。该参数可以通过调用 DescribeProject 的返回值中的 projectId 字段来获取。不填为默认项目。 示例值：0

LoginNodeOverview

登录节点概览。

被如下接口引用：DescribeClusters。

名称	类型	描述
NodeId	String	登录节点ID。 示例值：ins-jfhc307q

LoginSettings

描述了实例登录相关配置与信息。

被如下接口引用：AddNodes, CreateCluster, CreateWorkspaces。

名称	类型	必选	描述
Password	String	否	实例登录密码。不同操作系统类型密码复杂度限制不一样，具体如下： • Linux实例密码必须8到30位，至少包括两项[a-z]，[A-Z]、[0-9] 和 [( ) ` ~ ! @ # $ % ^ & * - + = | { } [ ] : ; ' , . ? / ]中的特殊符号。 • Windows实例密码必须12到30位，至少包括三项[a-z]，[A-Z]，[0-9] 和 [ ( ) ` ~ ! @ # $ % ^ & * - + = | { } [ ] : ; ' , . ? /]中的特殊符号。 若不指定该参数，则由系统随机生成密码，并通过站内信方式通知到用户。 示例值：test@123456
KeyIds	Array of String	否	实例登录密钥 示例值：sky-xxx

ManagerNode

管控节点信息

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2

名称	类型	必选	描述
SystemDisk	SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值： {"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
InternetAccessible	InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}
InstanceName	String	否	节点显示名称。 • 不指定节点显示名称则默认显示‘未命名’。 • 购买多个节点，如果指定模式串`{R:x}`，表示生成数字`[x, x+n-1]`，其中n表示购买节点的数量，例如`server_{R:3}`，购买1个时，节点显示名称为`server_3`；购买2个时，节点显示名称分别为`server_3`，`server_4`。支持指定多个模式串`{R:x}`。购买多个节点，如果不指定模式串，则在节点显示名称添加后缀`1`、`2...n`，其中n表示购买节点的数量，例如`server_`，购买2个时，节点显示名称分别为`server_1`，`server_2`。 • 最多支持60个字符（包含模式串）。 示例值：未命名
ProjectId	Integer	否	实例所属项目ID。该参数可以通过调用 DescribeProject 的返回值中的 projectId 字段来获取。不填为默认项目。 示例值：0
EnhancedService	EnhancedService	否	增强服务。通过该参数可以指定是否开启云安全、腾讯云可观测平台等服务。若不指定该参数，则默认开启腾讯云可观测平台、云安全服务、自动化助手服务。 示例值：{"SecurityService":{"Enabled":true},"MonitorService":{"Enabled":true},"AutomationService":{"Enabled":true}}

ManagerNodeOverview

管控节点概览。

被如下接口引用：DescribeClusters。

名称	类型	描述
NodeId	String	管控节点ID。 示例值：ins-jfhc307q

NodeActivity

节点活动信息。

被如下接口引用：DescribeClusterActivities。

名称	类型	描述
NodeInstanceId	String	节点活动所在的实例ID。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ins-1zll2of0
NodeActivityStatus	String	节点活动状态。取值范围： • RUNNING：运行中 • SUCCESSFUL：活动成功 • FAILED：活动失败 示例值：SUCCESSFUL

名称	类型	描述
NodeActivityStatusCode	String	节点活动状态码。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ActivitySuccess
NodeActivityStatusReason	String	节点活动状态原因。 注意：此字段可能返回 null，表示取不到有效值。 示例值：Activity success.

NodeOverview

节点概览信息。

被如下接口引用：DescribeNodes。

名称	类型	描述
InstanceId	String	节点实例ID。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ins-f5ew7k5w
Zone	String	节点所在可用区信息。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ap-chongqing-1
NodeState	String	节点状态。 • SUBMITTED：已完成提交。 • CREATING：创建中。 • CREATED：完成创建。 • INITING：初始化中。 • INIT_FAILED：初始化失败。 • RUNNING：运行中。 • DELETING：销毁中。 注意：此字段可能返回 null，表示取不到有效值。 示例值：RUNNING
ImageId	String	镜像ID。 注意：此字段可能返回 null，表示取不到有效值。 示例值：img-l8og963d
QueueName	String	节点所属队列名称。 注意：此字段可能返回 null，表示取不到有效值。 示例值：compute
NodeRole	String	节点角色。 • Manager：管控节点。 • Compute：计算节点。 • Login：登录节点。 • ManagerBackup：备用管控节点。 注意：此字段可能返回 null，表示取不到有效值。 示例值：Compute
NodeType	String	节点类型。 • STATIC：静态节点。 • DYNAMIC：弹性节点。 注意：此字段可能返回 null，表示取不到有效值。 示例值：STATIC
NodeId	String	thpc集群节点id 注意：此字段可能返回 null，表示取不到有效值。 示例值：node-2sh754ah
NodeAllocateState	String	节点的工作状态 示例值：IDLE

NodeScript

描述节点执行脚本信息。

被如下接口引用：CreateCluster, DescribeInitNodeScripts, ModifyInitNodeScripts。

名称	类型	必选	描述
ScriptPath	String	是	节点执行脚本获取地址。 目前仅支持cos地址。地址最大长度：255。 示例值：cos://demo-xxxxxxxx.cos.ap-guangzhou.myqcloud.com/test/echo.sh
Timeout	Integer	否	脚本执行超时时间（包含拉取脚本的时间）。单位秒，默认值：30。取值范围：10~1200。 示例值：30

Placement

描述了实例的抽象位置

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, DescribeClusters, DescribeWorkspaces, SetAutoScalingConfiguration。

名称	类型	必选	描述
Zone	String	是	实例所属的可用区名称。该参数可以通过调用 DescribeZones 的返回值中的Zone字段来获取。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ap-guangzhou-1

QueueConfig

扩容队列配置。

被如下接口引用：SetAutoScalingConfiguration。

名称	类型	必选	描述
QueueName	String	是	队列名称。 示例值：compute
MinSize	Integer	否	队列中弹性节点数量最小值。默认值：0。取值范围：0~200。 示例值：0
MaxSize	Integer	否	队列中弹性节点数量最大值。默认值：10。取值范围：0~200。 示例值：10
EnableAutoExpansion	Boolean	否	是否开启自动扩容。 示例值：False
EnableAutoShrink	Boolean	否	是否开启自动缩容。 示例值：False
ImageId	String	否	指定有效的镜像ID，格式形如`img-xxx`。目前仅支持公有镜和特定自定义镜像。 示例值：ins-jfhc307q
SystemDisk	SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值：{ "DiskSize": 50 }
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值：[{ "DiskSize": 50, "DiskType": "CLOUD_PREMIUM" }]
InternetAccessible	InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{ "PublicIpAssigned": "false" }
ExpansionNodeConfigs	Array of ExpansionNodeConfig	否	扩容节点配置信息。 示例值：{"Placement": "ap-guangzhou-6"}

名称	类型	必选	描述
DesiredIdleNodeCapacity	Integer	否	队列中期望的空闲节点数量（包含弹性节点和静态节点）。默认值：0。队列中，处于空闲状态的节点小于此值，集群会扩容弹性节点；处于空闲状态的节点大于此值，集群会缩容弹性节点。 示例值：5
DesiredNodeCount	Integer	否	队列中期望的总节点数。 示例值：1
ScaleOutRatio	Integer	否	扩容比例。默认值：100。取值范围：1~100。 如果扩容比例为50，那么每轮只会扩容当前作业负载所需的50%数量的节点。 示例值：50
ScaleOutNodeThreshold	Integer	否	比例扩容阈值。默认值：0。取值范围：0~200。 当作业负载需要扩容节点数量大于此值，当前扩容轮次按照ScaleOutRatio配置的比例进行扩容。当作业负载需要扩容节点数量小于此值，当前扩容轮次扩容当前作业负载所需数量的节点。 此参数配合ScaleOutRatio参数进行使用，用于比例扩容场景下，在作业负载所需节点数量较小时，加快收敛速度。 示例值：0
MaxNodesPerCycle	Integer	否	每轮扩容最大节点个数。默认值：100。取值范围：1~100。 示例值：10
ScaleUpMemRatio	Integer	否	扩容过程中，作业的内存在匹配实例机型时增大比例（不会影响作业提交的内存大小，只影响匹配计算过程）。 针对场景：由于实例机型的总内存会大于实例内部的可用内存，16GB内存规格的实例，实例操作系统内的可用内存只有约14.9GB内存。假设此时提交一个需要15GB内存的作业， - 当ScaleUpMemRatio=0时，会匹配到16GB内存规格的实例,但是由于操作系统内的可用内存为14.9GB小于作业所需的15GB，扩容出来的实例作业无法运行起来。 - 当ScaleUpMemRatio=10时，匹配实例规格会按照15*(1+10%)=16.5GB来进行实例规格匹配，则不会匹配到16GB的实例，而是更大内存规格的实例来保证作业能够被运行起来。 示例值：0
EnhancedService	EnhancedService	否	增强服务。通过该参数可以指定是否开启云安全、腾讯云可观测平台等服务。若不指定该参数，则默认开启腾讯云可观测平台、云安全服务、自动化助手服务。 示例值：{"SecurityService":{"Enabled":true},"MonitorService":{"Enabled":true},"AutomationService":{"Enabled":true}}

QueueConfigOverview

扩容队列配置概览。

被如下接口引用：DescribeAutoScalingConfiguration。

名称	类型	描述
QueueName	String	队列名称。 示例值：compute
MinSize	Integer	队列中弹性节点数量最小值。取值范围0~200。 示例值：1
MaxSize	Integer	队列中弹性节点数量最大值。取值范围0~200。 示例值：5
EnableAutoExpansion	Boolean	是否开启自动扩容。 示例值：False
EnableAutoShrink	Boolean	是否开启自动缩容。 示例值：False
ExpansionNodeConfigs	Array of ExpansionNodeConfigOverview	扩容节点配置信息。 示例值：

名称	类型	描述
		[{"InstanceType":"M5.SMALL8","InstanceChargeType":"POSTPAID_BY_HC
DesiredIdleNodeCapacity	Integer	队列中期望的空闲节点数量（包含弹性节点和静态节点）。默认值：0。队列中，处于空闲状 点小于此值，集群会扩容弹性节点；处于空闲状态的节点大于此值，集群会缩容弹性节点。 示例值：5
DesiredNodeCount	Integer	队列中期望的总节点数。 示例值：0
ScaleOutRatio	Integer	扩容比例。默认值：100。取值范围：1~100。 如果扩容比例为50，那么每轮只会扩容当前作业负载所需的50%数量的节点。 示例值：50
ScaleOutNodeThreshold	Integer	比例扩容阈值。默认值：0。取值范围：0~200。 当作业负载需要扩容节点数量大于此值，当前扩容轮次按照ScaleOutRatio配置的的比例进 容。当作业负载需要扩容节点数量小于此值，当前扩容轮次扩容当前作业负载所需数量的节点 此参数配合ScaleOutRatio参数进行使用，用于比例扩容场景下，在作业负载所需节点数量 时，加快收敛速度。 示例值：0
MaxNodesPerCycle	Integer	每轮扩容最大节点个数。 示例值：10
ScaleUpMemRatio	Integer	扩容过程中，作业的内存在匹配实例机型时增大比例（不会影响作业提交的内存大小，只影响 计算过程）。 针对场景：由于实例机型的总内存会大于实例内部的可用内存，16GB内存规格的实例，实例 系统内的可用内存只有约14.9GB内存。假设此时提交一个需要15GB内存的作业， - 当ScaleUpMemRatio=0时，会匹配到16GB内存规格的实例,但是由于操作系统内的可 为14.9GB小于作业所需的15GB，扩容出来的实例作业无法运行起来。 - 当ScaleUpMemRatio=10时，匹配实例规格会按照15*(1+10%)=16.5GB来进行实例 配，则不会匹配到16GB的实例，而是更大内存规格的实例来保证作业能够被运行起来。 示例值：0

QueueOverview

队列信息概览。

被如下接口引用：DescribeQueues。

名称	类型	描述
QueueName	String	队列名称。 示例值：compute

RunAutomationServiceEnabled

描述了 “云自动化助手” 服务相关的信息。

被如下接口引用：CreateCluster, CreateWorkspaces, SetAutoScalingConfiguration。

名称	类型	必选	描述
Enabled	Boolean	否	是否开启云自动化助手。取值范围： • TRUE：表示开启云自动化助手服务 • FALSE：表示不开启云自动化助手服务 默认取值：TRUE。 示例值：true

RunMonitorServiceEnabled

描述了 “腾讯云可观测平台” 服务相关的信息。

被如下接口引用：CreateCluster, CreateWorkspaces, SetAutoScalingConfiguration。

名称	类型	必选	描述
Enabled	Boolean	否	是否开启腾讯云可观测平台服务。取值范围： • TRUE：表示开启腾讯云可观测平台服务 • FALSE：表示不开启腾讯云可观测平台服务 默认取值：TRUE。 示例值：true

RunSecurityServiceEnabled

描述了 “云安全” 服务相关的信息。

被如下接口引用：CreateCluster, CreateWorkspaces, SetAutoScalingConfiguration。

名称	类型	必选	描述
Enabled	Boolean	否	是否开启云安全服务。取值范围： • TRUE：表示开启云安全服务 • FALSE：表示不开启云安全服务 默认取值：TRUE。 示例值：true

SpaceChargePrepaid

描述了工作空间的计费模式

被如下接口引用：CreateWorkspaces。

名称	类型	必选	描述
Period	Integer	否	购买实例的时长，单位：月。取值范围：1, 2, 3, 12, 24, 36。默认取值为1。 注意：此字段可能返回 null，表示取不到有效值。 示例值：1
RenewFlag	String	否	自动续费标识。取值范围： NOTIFY_AND_AUTO_RENEW：通知过期且自动续费 NOTIFY_AND_MANUAL_RENEW：通知过期不自动续费 DISABLE_NOTIFY_AND_MANUAL_RENEW：不通知过期不自动续费 默认取值：NOTIFY_AND_MANUAL_RENEW。若该参数指定为NOTIFY_AND_AUTO_RENEW，在账户余额充足的情况下，实例到期后将按月自动续费。 注意：此字段可能返回 null，表示取不到有效值。 示例值：NOTIFY_AND_AUTO_RENEW

SpaceDataDisk

工作空间数据盘配置

被如下接口引用：CreateWorkspaces。

名称	类型	必选	描述
DiskType	String	否	数据盘类型。数据盘类型限制详见存储概述。取值范围：

名称	类型	必选	描述
			• LOCAL_BASIC：本地硬盘 • LOCAL_SSD：本地SSD硬盘 • LOCAL_NVME：本地NVME硬盘，与InstanceType强相关，不支持指定 • LOCAL_PRO：本地HDD硬盘，与InstanceType强相关，不支持指定 • CLOUD_BASIC：普通云硬盘 • CLOUD_PREMIUM：高性能云硬盘 • CLOUD_SSD：SSD云硬盘 • CLOUD_HSSD：增强型SSD云硬盘 • CLOUD_TSSD：极速型SSD云硬盘 • CLOUD_BSSD：通用型SSD云硬盘 默认取值：LOCAL_BASIC。 该参数对ResizeInstanceDisk接口无效。 示例值：CLOUD_SSD
DiskId	String	否	数据盘 注意：此字段可能返回 null，表示取不到有效值。 示例值：disk-ciezoimt
DiskSize	Integer	否	数据盘大小，单位：GB。最小调整步长为10G，不同数据盘类型取值范围不同，具体限制详见：存储概述。默认值为0，表示不购买数据盘。更多限制详见产品文档。 示例值：50
DeleteWithInstance	Boolean	否	数据盘是否随子机销毁。取值范围： • TRUE：子机销毁时，销毁数据盘，只支持按小时后付费云盘 • FALSE：子机销毁时，保留数据盘 默认取值：TRUE 该参数目前仅用于 RunInstances 接口。 注意：此字段可能返回 null，表示取不到有效值。 示例值：true
SnapshotId	String	否	数据盘快照ID。选择的数据盘快照大小需小于数据盘大小。 注意：此字段可能返回 null，表示取不到有效值。

名称	类型	必选	描述
			示例值：snap-6yczrj8x
Encrypt	Boolean	否	数据盘是加密。取值范围： • true：加密 • false：不加密 默认取值：false 该参数目前仅用于 RunInstances 接口。 注意：此字段可能返回 null，表示取不到有效值。 示例值：false
KmsKeyId	String	否	自定义CMK对应的ID，取值为UUID或者类似kms-abcd1234。用于加密云盘。 该参数目前仅用于 CreateWorkspaces 接口。 注意：此字段可能返回 null，表示取不到有效值。 示例值：kms-abcd1234
ThroughputPerformance	Integer	否	云硬盘性能，单位：MB/s 注意：此字段可能返回 null，表示取不到有效值。 示例值：20
BurstPerformance	Boolean	否	突发性能 注：内测中。 注意：此字段可能返回 null，表示取不到有效值。 示例值：false

SpaceInfo

描述工作空间的信息

被如下接口引用：DescribeWorkspaces。

名称	类型	描述
SpaceId	String	工作空间ID 示例值：wks-3rr4w3t7
SpaceFamily	String	工作空间类型 示例值：96AS
SpaceType	String	工作空间规格 示例值：96AS.4XLARGE160
SpaceName	String	工作空间名称 示例值：workspace
SpaceState	String	工作空间状态。取值范围： • PENDING：表示创建中 • LAUNCH_FAILED：表示创建失败 • ONLINE：表示运行中 • ARREARS：表示隔离中 • TERMINATING：表示销毁中。 示例值：ONLINE
SpaceChargeType	String	工作空间计费模式 示例值：PREPAID
ResourceId	String	工作空间对应资源ID 示例值：ins-3zjugd6z
RenewFlag	String	自动续费标识 示例值：NOTIFY_AND_MANUAL_RENEW

名称	类型	描述
Tags	Array of Tag	工作空间关联的工作列表 示例值：[{"Key": "env", "Value": "product"}]
CreatedTime	Timestamp ISO8601	创建时间 示例值：2024-09-22T00:00:00+00:00
ExpiredTime	Timestamp ISO8601	到期时间 示例值：2025-09-22T00:00:00+00:00
Placement	Placement	工作空间所在位置 示例值：{"Zone": "ap-guangzhou-2"}
LatestOperation	String	工作空间的最新操作 注意：此字段可能返回 null，表示取不到有效值。 示例值：CreateWorkspaces
LatestOperationState	String	工作空间的最新操作状态 注意：此字段可能返回 null，表示取不到有效值。 示例值：SUCCESS

SpaceInternetAccessible

描述了工作空间的公网可访问性，声明了工作空间的公网使用计费模式，最大带宽等

被如下接口引用：CreateWorkspaces。

名称	类型	必选	描述
InternetChargeType	String	否	网络计费类型 注意：此字段可能返回 null，表示取不到有效值。 示例值：BANDWIDTH_PREPAID
InternetMaxBandwidthOut	Integer	否	公网出带宽上限，默认为0 注意：此字段可能返回 null，表示取不到有效值。 示例值：10
PublicIpAssigned	Boolean	否	是否分配公网IP 注意：此字段可能返回 null，表示取不到有效值。 示例值：false
BandwidthPackageId	String	否	带宽包ID 注意：此字段可能返回 null，表示取不到有效值。 示例值：bwp-h1ro

SpacePlacement

描述了实例的抽象位置，包括其所在的可用区，所属的项目

被如下接口引用：CreateWorkspaces。

名称	类型	必选	描述
Zone	String	是	可用区 示例值：ap-guangzhou-1
ProjectId	Integer	否	项目，默认是0 示例值：0

SpaceSystemDisk

工作空间系统盘配置

被如下接口引用：CreateWorkspaces。

名称	类型	必选	描述
DiskType	String	否	系统盘类型。系统盘类型限制详见存储概述。取值范围： • LOCAL_BASIC：本地硬盘 • LOCAL_SSD：本地SSD硬盘 • CLOUD_BASIC：普通云硬盘 • CLOUD_SSD：SSD云硬盘 • CLOUD_PREMIUM：高性能云硬盘 • CLOUD_BSSD：通用性SSD云硬盘 • CLOUD_HSSD：增强型SSD云硬盘 • CLOUD_TSSD：极速型SSD云硬盘 默认取值：当前有库存的硬盘类型。 注意：此字段可能返回 null，表示取不到有效值。 示例值：CLOUD_HSSD
DiskSize	Integer	否	系统盘大小，单位：GB。默认值为 50 注意：此字段可能返回 null，表示取不到有效值。 示例值：50

SpaceVirtualPrivateCloud

描述了工作空间VPC相关信息，包括子网，IP信息等

被如下接口引用：CreateWorkspaces。

名称	类型	必选	描述
VpcId	String	是	私有网络ID 示例值：vpc-2ij
SubnetId	String	是	私有网络子网ID 示例值：subnet-2ks
AsVpcGateway	Boolean	否	是否用作公网网关 示例值：false
PrivateIpAddresses	Array of String	否	私有网络子网 IP 数组 示例值：["10.0.0.1"]
Ipv6AddressCount	Integer	否	为弹性网卡指定随机生成 示例值：1

StorageOption

描述集群文件系统选项

被如下接口引用：AddClusterStorageOption, CreateCluster。

名称	类型	必选	描述
CFSOptions	Array of CFSOption	否	集群挂载CFS文件系统选项。
GooseFSOptions	Array of GooseFSOption	否	集群挂载GooseFS文件系统选项。
GooseFSxOptions	Array of GooseFSxOption	否	集群挂载GooseFSx文件系统选项。

StorageOptionOverview

集群存储选项概览信息。

被如下接口引用：DescribeClusterStorageOption。

名称	类型	描述
CFSOptions	Array of CFSOptionOverview	CFS存储选项概览信息列表。
GooseFSOptions	Array of GooseFSOptionOverview	GooseFS存储选项概览信息列表。

名称	类型	描述
GooseFSxOptions	Array of GooseFSxOptionOverview	GooseFSx存储选项概览信息列表。

SystemDisk

描述了操作系统所在块设备即系统盘的信息

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
DiskType	String	否	系统盘类型。系统盘类型限制详见存储概述。取值范围： CLOUD_BASIC：普通云硬盘 CLOUD_SSD：SSD云硬盘 CLOUD_PREMIUM：高性能云硬盘 默认取值：当前有库存的硬盘类型。 注意：此字段可能返回 null，表示取不到有效值。 示例值：CLOUD_BASIC
DiskSize	Integer	否	系统盘大小，单位：GB。默认值为 50 注意：此字段可能返回 null，表示取不到有效值。 示例值：50

Tag

标签键值对。

被如下接口引用：CreateCluster, CreateWorkspaces, DescribeWorkspaces。

名称	类型	必选	描述
Key	String	是	标签键 示例值：env
Value	String	是	标签值 示例值：product

TagSpecification

创建资源工作空间时同时绑定的标签对说明

被如下接口引用：CreateWorkspaces。

名称	类型	必选	描述
ResourceType	String	是	标签绑定的资源类型 示例值：workspace
Tags	Array of Tag	是	标签对列表 示例值：[{"Name": "env", "Value": "test"}]

VirtualPrivateCloud

描述了VPC相关信息

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
VpcId	String	是	私有网络ID，形如vpc-xxx。有效的VpcId可通过登录控制台查询；也可以调用接口 DescribeVpcEx ，从接口返回中的unVpcId字段获取。若在创建子机时VpcId与SubnetId同时传入DEFAULT，则强制使用默认vpc网络。 注意：此字段可能返回 null，表示取不到有效值。 示例值：vpc-jfkanfen

名称	类型	必选	描述
SubnetId	String	是	私有网络子网ID，形如subnet-xxx。有效的私有网络子网ID可通过登录控制台查询；也可以调用接口 DescribeSubnets ，从接口返回中的unSubnetId字段获取。若在创建子机时SubnetId与VpcId同时传入DEFAULT，则强制使用默认vpc网络。 注意：此字段可能返回 null，表示取不到有效值。 示例值：subnet-jfnakenf

错误码

最近更新时间：2025-05-15 02:06:42

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

Error 中的 Code 表示错误码，Message 表示该错误的具体信息。

错误码列表

公共错误码

错误码	说明
ActionOffline	接口已下线。
AuthFailure.InvalidAuthorization	请求头部的 `Authorization` 不符合腾讯云标准。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。
AuthFailure.MFAFailure	MFA 错误。
AuthFailure.SecretIdNotFound	密钥不存在。请在 控制台 检查密钥是否已被删除或者禁用，如状态正常，请检查密钥是否填写正确，注意前后不得有空格。
AuthFailure.SignatureExpire	签名过期。Timestamp 和服务器时间相差不得超过五分钟，请检查本地时间是否和标准时间同步。
AuthFailure.SignatureFailure	签名错误。签名计算错误，请对照调用方式中的签名方法文档检查签名计算过程。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未授权。请参考 CAM 文档对鉴权的说明。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误（包括参数格式、类型等错误）。
InvalidParameterValue	参数取值错误。
InvalidRequest	请求 body 的 multipart 格式错误。
IpInBlacklist	IP 地址在黑名单中。
IpNotInWhitelist	IP 地址不在白名单中。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数。
NoSuchProduct	产品不存在
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。

错误码	说明
RequestLimitExceeded.GlobalRegionUinLimitExceeded	主账号超过频率限制。
RequestLimitExceeded.IPLimitExceeded	IP 限频。
RequestLimitExceeded.UinLimitExceeded	主账号限频。
RequestSizeLimitExceeded	请求包超过限制大小。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
ResponseSizeLimitExceeded	返回包超过限制大小。
ServiceUnavailable	当前服务暂时不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误，用户多传未定义的参数会导致错误。
UnsupportedOperation	操作不支持。
UnsupportedProtocol	http(s) 请求协议错误，只支持 GET 和 POST 请求。
UnsupportedRegion	接口不支持所传地域。

业务错误码

错误码	说明
AuthFailure	CAM签名/鉴权错误。
InternalError.AgentRunScriptFail	agent执行脚本失败。
InternalError.CallCAM	CAM服务调用失败。
InternalError.CallCvm	cvm调用失败。
InvalidParameter.AtMostOne	参数互斥，最多只能传入一个参数
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.InvalidFilterNotSupportedName	不支持指定过滤器的键。
InvalidParameterValue.InvalidSpaceIdMalformed	工作空间实例ID格式不符合规范。
InvalidParameterValue.LimitExceeded	参数值数量超过限制。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.ParametersNotSupported	字段不支持此值。
InvalidParameterValue.SpaceIdNotFound	工作空间实例查找失败
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooShort	无效参数值。参数值太短。
InvalidParameterValue.TooSmall	参数值过小。
InvalidParameterValue.ValueDuplicated	参数值重复。不支持此操作。
LimitExceeded.QueueNumLimit	队列数量达到上限。
OperationDenied	操作被拒绝。

错误码	说明
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.ImageId	无法找到镜像ID。
ResourceNotFound.LocalPath	无法找到本地挂载路径。
ResourceNotFound.NodeId	无法找到ID对应节点。
ResourceNotFound.Queue	无法找到指定队列。
UnsupportedOperation.AutoScalingType	弹性伸缩类型不支持此操作。
UnsupportedOperation.ClusterAcceptOtherRequest	集群资源正在处理其他请求。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。
UnsupportedOperation.InvalidNodeRole	类型节点不支持当前操作。
UnsupportedOperation.NodeStatusNotSupport	节点状态不支持此操作。
UnsupportedOperation.ParameterTooLarge	参数值过大，不支持此操作。
UnsupportedOperation.ParameterTooSmall	参数值过小，不支持此操作。
UnsupportedOperation.QueueNotEmpty	队列内存在节点，不支持此操作。
UnsupportedOperation.SpaceChargeType	实例的付费模式不支持当前操作。
UnsupportedOperation.VpcIdConflict	vpc冲突，不支持当前操作。
UnsupportedOperation.WorkspaceStateArrears	隔离状态的工作空间实例不支持当前操作。

高性能计算平台 API 2021-11-09 产品版本

最近更新时间：2023-10-23 00:45:01

您正在浏览thpc产品文档的版本: 2021-11-09

最新版本

- 2023-03-21

以往版本

- 2021-11-09 (当前版本)
- 2022-04-01

更新历史

最近更新时间：2023-05-16 02:18:25

第 9 次发布

发布时间：2022-04-27 06:17:04

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- BindAutoScalingGroup
  - 新增入参：QueueName

第 8 次发布

发布时间：2022-04-01 06:15:33

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateCluster
  - 新增入参：LoginNode, LoginNodeCount, Tags

新增数据结构：

- LoginNode
- Tag

第 7 次发布

发布时间：2022-03-30 06:13:05

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- LoginNodeOverview

修改数据结构：

- ClusterOverview
  - 新增成员：LoginNodeSet, LoginNodeCount

第 6 次发布

发布时间：2021-12-22 08:17:26

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeClusters

新增数据结构：

- ClusterOverview
- ComputeNodeOverview
- ManagerNodeOverview

第 5 次发布

发布时间：2021-12-21 08:11:27

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- CFSOption
  - **修改成员：** Protocol, StorageType

第 4 次发布

发布时间：2021-12-16 08:15:07

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- BindAutoScalingGroup

第 3 次发布

发布时间：2021-12-15 08:13:47

本次发布包含了以下内容：

改善已有的文档。

新增数据结构：

- GooseFSOption

修改数据结构：

- StorageOption
  - 新增成员：GooseFSOptions
  - **修改成员：** CFSOptions

第 2 次发布

发布时间：2021-12-13 08:14:47

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateCluster
  - 新增入参：StorageOption

新增数据结构：

- CFSOption
- StorageOption

第 1 次发布

发布时间：2021-11-29 12:09:18

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- CreateCluster
- DeleteCluster

新增数据结构：

- ComputeNode
- DataDisk
- InstanceChargePrepaid
- InternetAccessible
- LoginSettings
- ManagerNode
- Placement
- SystemDisk
- VirtualPrivateCloud

简介

最近更新时间：2025-04-25 02:01:27

高性能计算平台（TencentCloud High Performance Computing，THPC）是一款腾讯云自研的高性能计算资源管理服务，集成腾讯云上的计算、存储、网络等产品资源，并整合 HPC 专用作业管理调度、集群管理等软件，向用户提供弹性灵活、性能卓越、自助化的计算服务。可以帮助您高效地管理云上高性能计算资源，实现弹性使用云上高性能计算资源的需求。

API 概览

最近更新时间：2024-08-02 02:14:39

集群相关接口

接口名称	接口功能	频率限制（次/秒）
CreateCluster	创建集群	20
DeleteCluster	删除集群	20
DescribeClusters	查询集群列表	20

弹性伸缩相关接口

接口名称	接口功能	频率限制（次/秒）
BindAutoScalingGroup	绑定弹性伸缩组	20

> 注意：
> 以上给出的接口频率限制维度为 API + 接入地域 + 子账号，有关限频更多说明参考：API 频率限制说明

调用方式
请求结构

最近更新时间：2025-05-21 01:32:49

1. 服务地址

API 支持就近地域接入，本产品就近地域接入域名为 thpc.tencentcloudapi.com ，也支持指定地域域名访问，例如广州地域的域名为 thpc.ap-guangzhou.tencentcloudapi.com 。

推荐使用就近地域接入域名。根据调用接口时客户端所在位置，会自动解析到**最近的**某个具体地域的服务器。例如在广州发起请求，会自动解析到广州的服务器，效果和指定 thpc.ap-guangzhou.tencentcloudapi.com 是一致的。

注意：对时延敏感的业务，建议指定带地域的域名。

注意：域名是 API 的接入点，并不代表产品或者接口实际提供服务的地域。产品支持的地域列表请在调用方式/公共参数文档中查阅，接口支持的地域请在接口文档输入参数中查阅。

目前支持的域名列表为：

接入地域	域名
就近地域接入（推荐，只支持非金融区）	thpc.tencentcloudapi.com
华南地区（广州）	thpc.ap-guangzhou.tencentcloudapi.com
华东地区（上海）	thpc.ap-shanghai.tencentcloudapi.com
华东地区（南京）	thpc.ap-nanjing.tencentcloudapi.com
华北地区（北京）	thpc.ap-beijing.tencentcloudapi.com
西南地区（成都）	thpc.ap-chengdu.tencentcloudapi.com
西南地区（重庆）	thpc.ap-chongqing.tencentcloudapi.com
港澳台地区（中国香港）	thpc.ap-hongkong.tencentcloudapi.com
亚太东南（新加坡）	thpc.ap-singapore.tencentcloudapi.com
亚太东南（雅加达）	thpc.ap-jakarta.tencentcloudapi.com
亚太东南（曼谷）	thpc.ap-bangkok.tencentcloudapi.com
亚太东北（首尔）	thpc.ap-seoul.tencentcloudapi.com
亚太东北（东京）	thpc.ap-tokyo.tencentcloudapi.com
美国东部（弗吉尼亚）	thpc.na-ashburn.tencentcloudapi.com
美国西部（硅谷）	thpc.na-siliconvalley.tencentcloudapi.com
南美地区（圣保罗）	thpc.sa-saopaulo.tencentcloudapi.com
欧洲地区（法兰克福）	thpc.eu-frankfurt.tencentcloudapi.com

注意：由于金融区和非金融区是隔离不互通的，因此当访问金融区服务时（公共参数 Region 为金融区地域），需要同时指定带金融区地域的域名，最好和 Region 的地域保持一致。

金融区接入地域	金融区域名
华东地区（上海金融）	thpc.ap-shanghai-fsi.tencentcloudapi.com
华南地区（深圳金融）	thpc.ap-shenzhen-fsi.tencentcloudapi.com

2. 通信协议

腾讯云 API 的所有接口均通过 HTTPS 进行通信，提供高安全性的通信通道。

3. 请求方法

支持的 HTTP 请求方法:

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型:

- application/json（推荐），必须使用签名方法 v3（TC3-HMAC-SHA256）。
- application/x-www-form-urlencoded，必须使用签名方法 v1（HmacSHA1 或 HmacSHA256）。
- multipart/form-data（仅部分接口支持），必须使用签名方法 v3（TC3-HMAC-SHA256）。

GET 请求的请求包大小不得超过32KB。POST 请求使用签名方法 v1（HmacSHA1、HmacSHA256）时不得超过1MB。POST 请求使用签名方法 v3（TC3-HMAC-SHA256）时支持10MB。

4. 字符编码

均使用 `UTF-8` 编码。

公共参数

最近更新时间：2024-11-15 02:07:19

公共参数是用于标识用户和接口签名的参数，如非必要，在每个接口单独的文档中不再对这些参数进行说明，但每次请求均需要携带这些参数，才能正常发起请求。

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

签名方法 v3（有时也称作 TC3-HMAC-SHA256）相比签名方法 v1（有些文档可能会简称签名方法），更安全，支持更大的请求包，支持 POST JSON 格式，性能有一定提升，推荐使用该签名方法计算签名。完整介绍详见 签名方法 v3。

注意：出于简化的目的，部分接口文档中的示例使用的是签名方法 v1 GET 请求，而不是更安全的签名方法 v3。

使用签名方法 v3 时，公共参数需要统一放到 HTTP Header 请求头部中，如下表所示：

参数名称	类型	必选	描述
Action	String	是	HTTP 请求头：X-TC-Action。操作的接口名称。取值参考接口文档输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	HTTP 请求头：X-TC-Region。地域参数，用来标识希望操作哪个地域的数据。取值参考接口文档中输入参数章节关于公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	HTTP 请求头：X-TC-Timestamp。当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。**注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。**
Version	String	是	HTTP 请求头：X-TC-Version。操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKID***/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKID*** 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为具体产品名，通常为域名前缀。例如，域名 cvm.tencentcloudapi.com 意味着产品名是 cvm。本产品取值为 thpc；tc3_request 为固定字符串； - SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部； - Signature：签名摘要，计算过程详见 文档。
Token	String	否	HTTP 请求头：X-TC-Token。即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	HTTP 请求头：X-TC-Language。指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

假设用户想要查询广州地域的云服务器实例列表中的前十个，接口参数设置为偏移量 Offset=0，返回数量 Limit=10，则其请求结构按照请求 URL、请求头部、请求体示例如下：

HTTP GET 请求结构示例：

```
https://cvm.tencentcloudapi.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: ap-guangzhou
```

HTTP POST（application/json）请求结构示例：

```
https://cvm.tencentcloudapi.com/

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: application/json
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

{"Offset":0,"Limit":10}
```

HTTP POST（multipart/form-data）请求结构示例（仅特定的接口支持）：

```
https://cvm.tencentcloudapi.com/

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: multipart/form-data; boundary=58731222010402
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

--58731222010402
Content-Disposition: form-data; name="Offset"

0
--58731222010402
Content-Disposition: form-data; name="Limit"

10
--58731222010402--
```

使用签名方法 v1 的公共参数

使用签名方法 v1（有时会称作 HmacSHA256 和 HmacSHA1），公共参数需要统一放到请求串中，完整介绍详见文档

参数名称	类型	必选	描述
Action	String	是	操作的接口名称。取值参考接口文档中输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。
Nonce	Integer	是	随机正整数，与 Timestamp 联合起来，用于防止重放攻击。
SecretId	String	是	在 云API密钥 上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey ，而 SecretKey 会用来生成请求签名 Signature。
Signature	String	是	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见 文档。

参数名称	类型	必选	描述
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

假设用户想要查询广州地域的云服务器实例列表，其请求结构按照请求 URL、请求头部、请求体示例如下：

HTTP GET 请求结构示例：

```
https://cvm.tencentcloudapi.com/?Action=DescribeInstances&Version=2017-03-12&SignatureMethod=HmacSHA256&Timestamp=1527672334&Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*******************************

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

```
https://cvm.tencentcloudapi.com/

Host: cvm.tencentcloudapi.com
Content-Type: application/x-www-form-urlencoded

Action=DescribeInstances&Version=2017-03-12&SignatureMethod=HmacSHA256&Timestamp=1527672334&Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*****************************
```

地域列表

本产品所有接口 Region 字段的可选值如下表所示。如果接口不支持该表中的所有地域，则会在接口文档中单独说明。

地域	取值
华北地区（北京）	ap-beijing
西南地区（成都）	ap-chengdu
西南地区（重庆）	ap-chongqing
华南地区（广州）	ap-guangzhou
华东地区（南京）	ap-nanjing
华东地区（上海）	ap-shanghai

签名方法 v3

最近更新时间：2024-12-25 02:03:35

以下文档说明了签名方法 v3 的签名过程，但仅在您编写自己的代码来调用腾讯云 API 时才有用。我们推荐您使用 腾讯云 API Explorer ，腾讯云 SDK 和 腾讯云命令行工具（TCCLI）等开发者工具，从而无需学习如何对 API 请求进行签名。

推荐使用 API Explorer　　点击调试

您可以通过 API Explorer 的【签名串生成】模块查看每个接口签名的生成过程。

腾讯云 API 会对每个请求进行身份验证，用户需要使用安全凭证，经过特定的步骤对请求进行签名（Signature），每个请求都需要在公共参数中指定该签名结果并以指定的方式和格式发送请求。

为什么要进行签名

签名通过以下方式帮助保护请求：

1. 验证请求者的身份
   签名确保请求是由持有有效访问密钥的人发送的。请参阅控制台 云 API 密钥 页面获取密钥相关信息。

2. 保护传输中的数据
   为了防止请求在传输过程中被篡改，腾讯云 API 会使用请求参数来计算请求的哈希值，并将生成的哈希值加密后作为请求的一部分，发送到腾讯云 API 服务器。服务器会使用收到的请求参数以同样的过程计算哈希值，并验证请求中的哈希值。如果请求被篡改，将导致哈希值不一致，腾讯云 API 将拒绝本次请求。

签名方法 v3（TC3-HMAC-SHA256）功能上覆盖了以前的签名方法 v1，而且更安全，支持更大的请求，支持 JSON 格式，POST 请求支持传空数组和空字符串，性能有一定提升，推荐使用该签名方法计算签名。

首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v3”，可以对生成签名过程进行验证，也可直接生成 SDK 代码。推荐使用腾讯云 API 配套的 8 种常见的编程语言 SDK，已经封装了签名和请求过程，均已开源，支持 Python、Java、PHP、Go、NodeJS、.NET、C++、Ruby。

申请安全凭证

本文使用的安全凭证为密钥，密钥包括 SecretId 和 SecretKey。每个用户最多可以拥有两对密钥。

- SecretId：用于标识 API 调用者身份，可以简单类比为用户名。
- SecretKey：用于验证 API 调用者的身份，可以简单类比为密码。
- **用户必须严格保管安全凭证，避免泄露，否则将危及财产安全。如已泄露，请立刻禁用该安全凭证。**

申请安全凭证的具体步骤如下：

1. 登录 腾讯云管理中心控制台 。
2. 前往 云API密钥 的控制台页面。
3. 在 云API密钥 页面，单击【新建密钥】创建一对密钥。

签名版本 v3 签名过程

云 API 支持 GET 和 POST 请求。对于GET方法，只支持 `Content-Type: application/x-www-form-urlencoded` 协议格式。对于POST方法，目前支持 `Content-Type: application/json` 以及 `Content-Type: multipart/form-data` 两种协议格式，json 格式绝大多数接口均支持，multipart 格式只有特定接口支持，此时该接口不能使用 json 格式调用，参考具体业务接口文档说明。推荐使用 POST 请求，因为两者的结果并无差异，但 GET 请求只支持 32 KB 以内的请求包。

下面以云服务器查询广州区实例列表作为例子，分步骤介绍签名的计算过程。我们选择该接口是因为：

1. 云服务器默认已开通，该接口很常用；
2. 该接口是只读的，不会改变现有资源的状态；
3. 接口覆盖的参数种类较全，可以演示包含数据结构的数组如何使用。

在示例中，不论公共参数或者接口的参数，我们尽量选择容易犯错的情况。在实际调用接口时，请根据实际情况来，每个接口的参数并不相同，不要照抄这个例子的参数和值。此外，这里只展示了部分公共参数和接口输入参数，用户可以根据实际需要添加其他参数，例如 Language 和 Token 公共参数（在 HTTP 头部设置，添加 X-TC-前缀）。

假设用户的 SecretId 和 SecretKey 分别是：AKID******************************** 和 ********************************。用户想查看广州区云服务器名为“未命名”的主机状态，只返回一条数据。则请求可能为：

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f" \
-H "Content-Type: application/json; charset=utf-8" \
-H "Host: cvm.tencentcloudapi.com" \
-H "X-TC-Action: DescribeInstances" \
-H "X-TC-Timestamp: 1551113065" \
-H "X-TC-Version: 2017-03-12" \
-H "X-TC-Region: ap-guangzhou" \
-d '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
```

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

```
CanonicalRequest =
HTTPRequestMethod + '\n' +
CanonicalURI + '\n' +
CanonicalQueryString + '\n' +
CanonicalHeaders + '\n' +
SignedHeaders + '\n' +
HashedRequestPayload
```

字段名称	解释
HTTPRequestMethod	HTTP 请求方法（GET、POST）。此示例取值为 POST。
CanonicalURI	URI 参数，API 3.0 固定为正斜杠（/）。
CanonicalQueryString	发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串""，对于 GET 请求，则为 URL 中问号（?）后面的字符串内容，例如：Limit=10&Offset=0。 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码（特殊字符编码后需大写字母），字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入其他头部参与签名以提高自身请求的唯一性和安全性，此示例额外增加了接口名头部。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接； 2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。 此示例计算结果是 content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n。 注意：content-type 必须和实际发送的相符合，有些编程语言网络库即使未指定也会自动添加 charset 值，如果签名时和发送时不一致，服务器会返回签名校验失败。
SignedHeaders	参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。 拼接规则： 1. 头部 key 统一转成小写； 2. 多个头部 key（小写）按照 ASCII 升序进行拼接，并且以分号（;）分隔。 此示例为 content-type;host;x-tc-action
HashedRequestPayload	请求正文（payload，即 body，此示例为 {"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}）的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，即对 HTTP 请求正文做 SHA256 哈希，然后十六进制编

字段名称	解释
	码，最后编码串转换成小写字母。对于 GET 请求，RequestPayload 固定为空字符串。此示例计算结果是 35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064。

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com
x-tc-action:describeinstances

content-type;host;x-tc-action
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

```
StringToSign =
Algorithm + "\n" +
RequestTimestamp + "\n" +
CredentialScope + "\n" +
HashedCanonicalRequest
```

字段名称	解释
Algorithm	签名算法，目前固定为 TC3-HMAC-SHA256。
RequestTimestamp	请求时间戳，即请求头部的公共参数 X-TC-Timestamp 取值，取当前时间 UNIX 时间戳，精确到秒。此示例取值为 1551113065。
CredentialScope	凭证范围，格式为 Date/service/tc3_request，包含日期、所请求的服务和终止字符串（tc3_request）。**Date 为 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致**；service 为产品名，必须与调用的产品域名一致。此示例计算结果是 2019-02-25/cvm/tc3_request。
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。此示例计算结果是 7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84。

注意：

1. Date 必须从时间戳 X-TC-Timestamp 计算得到，且时区为 UTC+0。如果加入系统本地时区信息，例如东八区，将导致白天和晚上调用成功，但是凌晨时调用必定失败。假设时间戳为 1551113065，在东八区的时间是 2019-02-26 00:44:25，但是计算得到的 Date 取 UTC+0 的日期应为 2019-02-25，而不是 2019-02-26。
2. Timestamp 必须是当前系统时间，且需确保系统时间和标准时间是同步的，如果相差超过五分钟则必定失败。如果长时间不和标准时间同步，可能运行一段时间后，请求失败，返回签名过期错误。

根据以上规则，示例中得到的待签名字符串如下：

```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84
```

3. 计算签名

1）计算派生签名密钥，伪代码如下：

```
SecretKey = "********************************"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

派生出的密钥 `SecretDate`、`SecretService` 和 `SecretSigning` 是二进制的数据，可能包含不可打印字符，将其转为十六进制字符串打印的输出分别为：
da98fb70dcf6b112dc21038d1eeeb3a95c74b4dcb12c1131f864f6066bd02be0，
8d70cbefb03939f929db64d32dc2ba89b1095620119fe3e050e2b18c5bd2752f，
b596b923aad85185e2d1f6659d2a062e0a86731226e021e61bfe06f7ed05f5af。

请注意，不同的编程语言，HMAC 库函数中参数顺序可能不一样，请以实际情况为准。此处的伪代码密钥参数 key 在前，消息参数 data 在后。通常标准库函数会提供二进制格式的返回值，也可能会提供打印友好的十六进制格式的返回值，此处使用的是二进制格式。

字段名称	解释
SecretKey	原始的 SecretKey，即 `********************************`。
Date	即 Credential 中的 Date 字段信息。此示例取值为 `2019-02-25`。
Service	即 Credential 中的 Service 字段信息。此示例取值为 `cvm`。

2）计算签名，伪代码如下：

```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

此示例计算结果是 `10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f`。

4. 拼接 Authorization

按如下格式拼接 Authorization：

```
Authorization =
Algorithm + ' ' +
'Credential=' + SecretId + '/' + CredentialScope + ', ' +
'SignedHeaders=' + SignedHeaders + ', ' +
'Signature=' + Signature
```

字段名称	解释
Algorithm	签名方法，固定为 `TC3-HMAC-SHA256`。
SecretId	密钥对中的 SecretId，即 `AKID********************************`。
CredentialScope	见上文，凭证范围。此示例计算结果是 `2019-02-25/cvm/tc3_request`。
SignedHeaders	见上文，参与签名的头部信息。此示例取值为 `content-type;host;x-tc-action`。
Signature	签名值。此示例计算结果是 `10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f`。

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
```

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}
```

> ⚠ 注意：
>
> 请求发送时的 HTTP 头部（Header）和请求体（Payload）必须和签名计算过程中的内容完全一致，否则会返回签名不一致错误。可以通过打印实际请求内容，网络抓包等方式对比排查。

签名演示

在实际调用 API 3.0 时，推荐使用配套的腾讯云 SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

下面提供了不同产品的生成签名 demo，您可以找到对应的产品参考签名的生成：

- Signature Demo

为了更清楚地解释签名过程，下面以实际编程语言为例，将上述的签名过程完整实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK。

Java

```java
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TencentCloudAPITC3Demo {
private final static Charset UTF8 = StandardCharsets.UTF_8;
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
private final static String SECRET_ID = System.getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
```

```
private final static String SECRET_KEY = System.getenv("TENCENTCLOUD_SECRET_KEY");
private final static String CT_JSON = "application/json; charset=utf-8";

public static byte[] hmac256(byte[] key, String msg) throws Exception {
Mac mac = Mac.getInstance("HmacSHA256");
SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
mac.init(secretKeySpec);
return mac.doFinal(msg.getBytes(UTF8));
}

public static String sha256Hex(String s) throws Exception {
MessageDigest md = MessageDigest.getInstance("SHA-256");
byte[] d = md.digest(s.getBytes(UTF8));
return DatatypeConverter.printHexBinary(d).toLowerCase();
}

public static void main(String[] args) throws Exception {
String service = "cvm";
String host = "cvm.tencentcloudapi.com";
String region = "ap-guangzhou";
String action = "DescribeInstances";
String version = "2017-03-12";
String algorithm = "TC3-HMAC-SHA256";
String timestamp = "1551113065";
//String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
// 注意时区，否则容易出错
sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

// ************ 步骤 1: 拼接规范请求串 ************
String httpRequestMethod = "POST";
String canonicalUri = "/";
String canonicalQueryString = "";
String canonicalHeaders = "content-type:application/json; charset=utf-8\n"
+ "host:" + host + "\n" + "x-tc-action:" + action.toLowerCase() + "\n";
String signedHeaders = "content-type;host;x-tc-action";

String payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}";
String hashedRequestPayload = sha256Hex(payload);
String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryString + "\n"
+ canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
System.out.println(canonicalRequest);

// ************ 步骤 2: 拼接待签名字符串 ************
String credentialScope = date + "/" + service + "/" + "tc3_request";
String hashedCanonicalRequest = sha256Hex(canonicalRequest);
String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
System.out.println(stringToSign);

// ************ 步骤 3: 计算签名 ************
byte[] secretDate = hmac256(("TC3" + SECRET_KEY).getBytes(UTF8), date);
byte[] secretService = hmac256(secretDate, service);
byte[] secretSigning = hmac256(secretService, "tc3_request");
String signature = DatatypeConverter.printHexBinary(hmac256(secretSigning, stringToSign)).toLowerCase();
System.out.println(signature);
```

```
// ************* 步骤 4：拼接 Authorization *************
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
System.out.println(authorization);

TreeMap<String, String> headers = new TreeMap<String, String>();
headers.put("Authorization", authorization);
headers.put("Content-Type", CT_JSON);
headers.put("Host", host);
headers.put("X-TC-Action", action);
headers.put("X-TC-Timestamp", timestamp);
headers.put("X-TC-Version", version);
headers.put("X-TC-Region", region);

StringBuilder sb = new StringBuilder();
sb.append("curl -X POST https://").append(host)
.append(" -H \"Authorization: ").append(authorization).append("\"")
.append(" -H \"Content-Type: application/json; charset=utf-8\"")
.append(" -H \"Host: ").append(host).append("\"")
.append(" -H \"X-TC-Action: ").append(action).append("\"")
.append(" -H \"X-TC-Timestamp: ").append(timestamp).append("\"")
.append(" -H \"X-TC-Version: ").append(version).append("\"")
.append(" -H \"X-TC-Region: ").append(region).append("\"")
.append(" -d '").append(payload).append("'");
System.out.println(sb.toString());
}
}
```

Python

```
# -*- coding: utf-8 -*-
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# 密钥参数
# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secret_key = os.environ.get("TENCENTCLOUD_SECRET_KEY")

service = "cvm"
host = "cvm.tencentcloudapi.com"
endpoint = "https://" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
#timestamp = int(time.time())
timestamp = 1551113065
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 1, "Filters": [{"Values": [u"未命名"], "Name": "instance-name"}]}

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = "POST"
canonical_uri = "/"
canonical_querystring = ""
```

```
ct = "application/json; charset=utf-8"
payload = json.dumps(params)
canonical_headers = "content-type:%s\nhost:%s\nx-tc-action:%s\n" % (ct, host, action.lower())
signed_headers = "content-type;host;x-tc-action"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
canonical_uri + "\n" +
canonical_querystring + "\n" +
canonical_headers + "\n" +
signed_headers + "\n" +
hashed_request_payload)
print(canonical_request)

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
str(timestamp) + "\n" +
credential_scope + "\n" +
hashed_canonical_request)
print(string_to_sign)


# ************* 步骤 3：计算签名 *************
# 计算签名摘要函数
def sign(key, msg):
return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
print(signature)

# ************* 步骤 4：拼接 Authorization *************
authorization = (algorithm + " " +
"Credential=" + secret_id + "/" + credential_scope + ", " +
"SignedHeaders=" + signed_headers + ", " +
"Signature=" + signature)
print(authorization)

print('curl -X POST ' + endpoint
+ ' -H "Authorization: ' + authorization + '"'
+ ' -H "Content-Type: application/json; charset=utf-8"'
+ ' -H "Host: ' + host + '"'
+ ' -H "X-TC-Action: ' + action + '"'
+ ' -H "X-TC-Timestamp: ' + str(timestamp) + '"'
+ ' -H "X-TC-Version: ' + version + '"'
+ ' -H "X-TC-Region: ' + region + '"'
+ " -d '" + payload + "'")
```

Golang

```
package main

import (
"crypto/hmac"
```

```
"crypto/sha256"
"encoding/hex"
"fmt"
"os"
"strings"
"time"
)

func sha256hex(s string) string {
b := sha256.Sum256([]byte(s))
return hex.EncodeToString(b[:])
}

func hmacsha256(s, key string) string {
hashed := hmac.New(sha256.New, []byte(key))
hashed.Write([]byte(s))
return string(hashed.Sum(nil))
}

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
host := "cvm.tencentcloudapi.com"
algorithm := "TC3-HMAC-SHA256"
service := "cvm"
version := "2017-03-12"
action := "DescribeInstances"
region := "ap-guangzhou"
//var timestamp int64 = time.Now().Unix()
var timestamp int64 = 1551113065

// step 1: build canonical request string
httpRequestMethod := "POST"
canonicalURI := "/"
canonicalQueryString := ""
canonicalHeaders := fmt.Sprintf("content-type:%s\nhost:%s\nx-tc-action:%s\n",
"application/json; charset=utf-8", host, strings.ToLower(action))
signedHeaders := "content-type;host;x-tc-action"
payload := `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`
hashedRequestPayload := sha256hex(payload)
canonicalRequest := fmt.Sprintf("%s\n%s\n%s\n%s\n%s\n%s",
httpRequestMethod,
canonicalURI,
canonicalQueryString,
canonicalHeaders,
signedHeaders,
hashedRequestPayload)
fmt.Println(canonicalRequest)

// step 2: build string to sign
date := time.Unix(timestamp, 0).UTC().Format("2006-01-02")
credentialScope := fmt.Sprintf("%s/%s/tc3_request", date, service)
hashedCanonicalRequest := sha256hex(canonicalRequest)
string2sign := fmt.Sprintf("%s\n%d\n%s\n%s",
algorithm,
```

```
timestamp,
credentialScope,
hashedCanonicalRequest)
fmt.Println(string2sign)

// step 3: sign string
secretDate := hmacsha256(date, "TC3"+secretKey)
secretService := hmacsha256(service, secretDate)
secretSigning := hmacsha256("tc3_request", secretService)
signature := hex.EncodeToString([]byte(hmacsha256(string2sign, secretSigning)))
fmt.Println(signature)

// step 4: build authorization
authorization := fmt.Sprintf("%s Credential=%s/%s, SignedHeaders=%s, Signature=%s",
algorithm,
secretId,
credentialScope,
signedHeaders,
signature)
fmt.Println(authorization)

curl := fmt.Sprintf(`curl -X POST https://%s\
-H "Authorization: %s"\
-H "Content-Type: application/json; charset=utf-8"\
-H "Host: %s" -H "X-TC-Action: %s"\
-H "X-TC-Timestamp: %d"\
-H "X-TC-Version: %s"\
-H "X-TC-Region: %s"\
-d '%s'`, host, authorization, host, action, timestamp, version, region, payload)
fmt.Println(curl)
}
```

PHP

```
<?php
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$host = "cvm.tencentcloudapi.com";
$service = "cvm";
$version = "2017-03-12";
$action = "DescribeInstances";
$region = "ap-guangzhou";
// $timestamp = time();
$timestamp = 1551113065;
$algorithm = "TC3-HMAC-SHA256";

// step 1: build canonical request string
$httpRequestMethod = "POST";
$canonicalUri = "/";
$canonicalQueryString = "";
$canonicalHeaders = implode("\n", [
"content-type:application/json; charset=utf-8",
"host:".$host,
"x-tc-action:".strtolower($action),
```

```
""
]);
$signedHeaders = implode(";", [
"content-type",
"host",
"x-tc-action",
]);
$payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}';
$hashedRequestPayload = hash("SHA256", $payload);
$canonicalRequest = $httpRequestMethod."\n"
.$canonicalUri."\n"
.$canonicalQueryString."\n"
.$canonicalHeaders."\n"
.$signedHeaders."\n"
.$hashedRequestPayload;
echo $canonicalRequest.PHP_EOL;

// step 2: build string to sign
$date = gmdate("Y-m-d", $timestamp);
$credentialScope = $date."/".$service."/tc3_request";
$hashedCanonicalRequest = hash("SHA256", $canonicalRequest);
$stringToSign = $algorithm."\n"
.$timestamp."\n"
.$credentialScope."\n"
.$hashedCanonicalRequest;
echo $stringToSign.PHP_EOL;

// step 3: sign string
$secretDate = hash_hmac("SHA256", $date, "TC3".$secretKey, true);
$secretService = hash_hmac("SHA256", $service, $secretDate, true);
$secretSigning = hash_hmac("SHA256", "tc3_request", $secretService, true);
$signature = hash_hmac("SHA256", $stringToSign, $secretSigning);
echo $signature.PHP_EOL;

// step 4: build authorization
$authorization = $algorithm
." Credential=".$secretId."/".$credentialScope
.", SignedHeaders=".$signedHeaders.", Signature=".$signature;
echo $authorization.PHP_EOL;

$curl = "curl -X POST https://".$host
.' -H "Authorization: '.$authorization.'"'
.' -H "Content-Type: application/json; charset=utf-8"'
.' -H "Host: '.$host.'"'
.' -H "X-TC-Action: '.$action.'"'
.' -H "X-TC-Timestamp: '.$timestamp.'"'
.' -H "X-TC-Version: '.$version.'"'
.' -H "X-TC-Region: '.$region.'"'
." -d '".$payload."'";
echo $curl.PHP_EOL;
```

Ruby

```
# -*- coding: UTF-8 -*-
# require ruby>=2.3.0
require 'digest'
```

```ruby
require 'json'
require 'time'
require 'openssl'

# 密钥参数
# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

service = 'cvm'
host = 'cvm.tencentcloudapi.com'
endpoint = 'https://' + host
region = 'ap-guangzhou'
action = 'DescribeInstances'
version = '2017-03-12'
algorithm = 'TC3-HMAC-SHA256'
# timestamp = Time.now.to_i
timestamp = 1551113065
date = Time.at(timestamp).utc.strftime('%Y-%m-%d')

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = 'POST'
canonical_uri = '/'
canonical_querystring = ''
canonical_headers = "content-type:application/json; charset=utf-8\nhost:#{host}\nx-tc-action:#{action.downcase}\n"
signed_headers = 'content-type;host;x-tc-action'
# params = { 'Limit' => 1, 'Filters' => [{ 'Name' => 'instance-name', 'Values' => ['未命名'] }] }
# payload = JSON.generate(params, { 'ascii_only' => true, 'space' => ' ' })
# json will generate in random order, to get specified result in example, we hard-code it here.
payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
hashed_request_payload = Digest::SHA256.hexdigest(payload)
canonical_request = [
http_request_method,
canonical_uri,
canonical_querystring,
canonical_headers,
signed_headers,
hashed_request_payload,
].join("\n")

puts canonical_request

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + '/' + service + '/' + 'tc3_request'
hashed_request_payload = Digest::SHA256.hexdigest(canonical_request)
string_to_sign = [
algorithm,
timestamp.to_s,
credential_scope,
hashed_request_payload,
].join("\n")
puts string_to_sign

# ************* 步骤 3：计算签名 *************
digest = OpenSSL::Digest.new('sha256')
secret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)
```

```
secret_service = OpenSSL::HMAC.digest(digest, secret_date, service)
secret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')
signature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)
puts signature

# ************* 步骤 4：拼接 Authorization *************
authorization = "#{algorithm} Credential=#{secret_id}/#{credential_scope}, SignedHeaders=#{signed_headers}, Signature=#{si
gnature}"
puts authorization

puts 'curl -X POST ' + endpoint \
+ ' -H "Authorization: ' + authorization + '"' \
+ ' -H "Content-Type: application/json; charset=utf-8"' \
+ ' -H "Host: ' + host + '"' \
+ ' -H "X-TC-Action: ' + action + '"' \
+ ' -H "X-TC-Timestamp: ' + timestamp.to_s + '"' \
+ ' -H "X-TC-Version: ' + version + '"' \
+ ' -H "X-TC-Region: ' + region + '"' \
+ " -d '" + payload + "'"
```

DotNet

```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

public class Application
{
public static string SHA256Hex(string s)
{
using (SHA256 algo = SHA256.Create())
{
byte[] hashbytes = algo.ComputeHash(Encoding.UTF8.GetBytes(s));
StringBuilder builder = new StringBuilder();
for (int i = 0; i < hashbytes.Length; ++i)
{
builder.Append(hashbytes[i].ToString("x2"));
}
return builder.ToString();
}
}

public static byte[] HmacSHA256(byte[] key, byte[] msg)
{
using (HMACSHA256 mac = new HMACSHA256(key))
{
return mac.ComputeHash(msg);
}
}

public static Dictionary<String, String> BuildHeaders(string secretid,
string secretkey, string service, string endpoint, string region,
string action, string version, DateTime date, string requestPayload)
{
string datestr = date.ToString("yyyy-MM-dd");
```

```
DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, MidpointRounding.AwayFromZero) / 1000;
// ************* 步骤 1: 拼接规范请求串 *************
string algorithm = "TC3-HMAC-SHA256";
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string contentType = "application/json";
string canonicalHeaders = "content-type:" + contentType + "; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.ToLower() + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string hashedRequestPayload = SHA256Hex(requestPayload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
Console.WriteLine(canonicalRequest);

// ************* 步骤 2: 拼接待签名字符串 *************
string credentialScope = datestr + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = SHA256Hex(canonicalRequest);
string stringToSign = algorithm + "\n"
+ requestTimestamp.ToString() + "\n"
+ credentialScope + "\n"
+ hashedCanonicalRequest;
Console.WriteLine(stringToSign);

// ************* 步骤 3: 计算签名 *************
byte[] tc3SecretKey = Encoding.UTF8.GetBytes("TC3" + secretkey);
byte[] secretDate = HmacSHA256(tc3SecretKey, Encoding.UTF8.GetBytes(datestr));
byte[] secretService = HmacSHA256(secretDate, Encoding.UTF8.GetBytes(service));
byte[] secretSigning = HmacSHA256(secretService, Encoding.UTF8.GetBytes("tc3_request"));
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringToSign));
string signature = BitConverter.ToString(signatureBytes).Replace("-", "").ToLower();
Console.WriteLine(signature);

// ************* 步骤 4: 拼接 Authorization *************
string authorization = algorithm + " "
+ "Credential=" + secretid + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", "
+ "Signature=" + signature;
Console.WriteLine(authorization);

Dictionary<string, string> headers = new Dictionary<string, string>();
headers.Add("Authorization", authorization);
headers.Add("Host", endpoint);
headers.Add("Content-Type", contentType + "; charset=utf-8");
headers.Add("X-TC-Timestamp", requestTimestamp.ToString());
headers.Add("X-TC-Version", version);
headers.Add("X-TC-Action", action);
headers.Add("X-TC-Region", region);
return headers;
}
public static void Main(string[] args)
```

```
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID*******************************
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

string service = "cvm";
string endpoint = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";

// 此处由于示例规范的原因，采用时间戳2019-02-26 00:44:25，此参数作为示例，如果在项目中，您应当使用：
// DateTime date = DateTime.UtcNow;
// 注意时区，建议此时间统一采用UTC时间戳，否则容易出错
DateTime date = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc).AddSeconds(1551113065);
string requestPayload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name
\"}]}";

Dictionary<string, string> headers = BuildHeaders(SECRET_ID, SECRET_KEY, service
, endpoint, region, action, version, date, requestPayload);

Console.WriteLine("POST https://cvm.tencentcloudapi.com");
foreach (KeyValuePair<string, string> kv in headers)
{
Console.WriteLine(kv.Key + ": " + kv.Value);
}
Console.WriteLine();
Console.WriteLine(requestPayload);
}
}
```

NodeJS

```
const crypto = require('crypto');

function sha256(message, secret = '', encoding) {
const hmac = crypto.createHmac('sha256', secret)
return hmac.update(message).digest(encoding)
}

function getHash(message, encoding = 'hex') {
const hash = crypto.createHash('sha256')
return hash.update(message).digest(encoding)
}

function getDate(timestamp) {
const date = new Date(timestamp * 1000)
const year = date.getUTCFullYear()
const month = ('0' + (date.getUTCMonth() + 1)).slice(-2)
const day = ('0' + date.getUTCDate()).slice(-2)
return `${year}-${month}-${day}`
}

function main(){
```

```
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

const endpoint = "cvm.tencentcloudapi.com"
const service = "cvm"
const region = "ap-guangzhou"
const action = "DescribeInstances"
const version = "2017-03-12"
//const timestamp = getTime()
const timestamp = 1551113065
//时间处理，获取世界时间日期
const date = getDate(timestamp)

// ************* 步骤 1：拼接规范请求串 *************
const payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}"

const hashedRequestPayload = getHash(payload);
const httpRequestMethod = "POST"
const canonicalUri = "/"
const canonicalQueryString = ""
const canonicalHeaders = "content-type:application/json; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.toLowerCase() + "\n"
const signedHeaders = "content-type;host;x-tc-action"

const canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload
console.log(canonicalRequest)

// ************* 步骤 2：拼接待签名字符串 *************
const algorithm = "TC3-HMAC-SHA256"
const hashedCanonicalRequest = getHash(canonicalRequest);
const credentialScope = date + "/" + service + "/" + "tc3_request"
const stringToSign = algorithm + "\n" +
timestamp + "\n" +
credentialScope + "\n" +
hashedCanonicalRequest
console.log(stringToSign)

// ************* 步骤 3：计算签名 *************
const kDate = sha256(date, 'TC3' + SECRET_KEY)
const kService = sha256(service, kDate)
const kSigning = sha256('tc3_request', kService)
const signature = sha256(stringToSign, kSigning, 'hex')
console.log(signature)

// ************* 步骤 4：拼接 Authorization *************
const authorization = algorithm + " " +
"Credential=" + SECRET_ID + "/" + credentialScope + ", " +
"SignedHeaders=" + signedHeaders + ", " +
```

```
"Signature=" + signature
console.log(authorization)

const curlcmd = 'curl -X POST ' + "https://" + endpoint
+ ' -H "Authorization: ' + authorization + '"'
+ ' -H "Content-Type: application/json; charset=utf-8"'
+ ' -H "Host: ' + endpoint + '"'
+ ' -H "X-TC-Action: ' + action + '"'
+ ' -H "X-TC-Timestamp: ' + timestamp.toString() + '"'
+ ' -H "X-TC-Version: ' + version + '"'
+ ' -H "X-TC-Region: ' + region + '"'
+ " -d '" + payload + "'"
console.log(curlcmd)
}
main()
```

C++

```
#include <algorithm>
#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdio.h>
#include <time.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>

using namespace std;

string get_data(int64_t &timestamp)
{
string utcDate;
char buff[20] = {0};
// time_t timenow;
struct tm sttime;
sttime = *gmtime(&timestamp);
strftime(buff, sizeof(buff), "%Y-%m-%d", &sttime);
utcDate = string(buff);
return utcDate;
}

string int2str(int64_t n)
{
std::stringstream ss;
ss << n;
return ss.str();
}

string sha256Hex(const string &str)
{
char buf[3];
unsigned char hash[SHA256_DIGEST_LENGTH];
SHA256_CTX sha256;
SHA256_Init(&sha256);
```

```
SHA256_Update(&sha256, str.c_str(), str.size());
SHA256_Final(hash, &sha256);
std::string NewString = "";
for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
{
snprintf(buf, sizeof(buf), "%02x", hash[i]);
NewString = NewString + buf;
}
return NewString;
}

string HmacSha256(const string &key, const string &input)
{
unsigned char hash[32];

HMAC_CTX *h;
#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif

HMAC_Init_ex(h, &key[0], key.length(), EVP_sha256(), NULL);
HMAC_Update(h, ( unsigned char* )&input[0], input.length());
unsigned int len = 32;
HMAC_Final(h, hash, &len);

#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

std::stringstream ss;
ss << std::setfill('0');
for (int i = 0; i < len; i++)
{
ss << hash[i];
}

return (ss.str());
}

string HexEncode(const string &input)
{
static const char* const lut = "0123456789abcdef";
size_t len = input.length();

string output;
output.reserve(2 * len);
for (size_t i = 0; i < len; ++i)
{
const unsigned char c = input[i];
output.push_back(lut[c >> 4]);
output.push_back(lut[c & 15]);
```

```
}
return output;
}

int main()
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
string SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");

string service = "cvm";
string host = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";
int64_t timestamp = 1551113065;
string date = get_data(timestamp);

// ************* 步骤 1: 拼接规范请求串 *************
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string lower = action;
std::transform(action.begin(), action.end(), lower.begin(), ::tolower);
string canonicalHeaders = string("content-type:application/json; charset=utf-8\n")
+ "host:" + host + "\n"
+ "x-tc-action:" + lower + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}";
string hashedRequestPayload = sha256Hex(payload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
cout << canonicalRequest << endl;

// ************* 步骤 2: 拼接待签名字符串 *************
string algorithm = "TC3-HMAC-SHA256";
string RequestTimestamp = int2str(timestamp);
string credentialScope = date + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = sha256Hex(canonicalRequest);
string stringToSign = algorithm + "\n" + RequestTimestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
cout << stringToSign << endl;

// ************* 步骤 3: 计算签名 ***************
string kKey = "TC3" + SECRET_KEY;
string kDate = HmacSha256(kKey, date);
string kService = HmacSha256(kDate, service);
string kSigning = HmacSha256(kService, "tc3_request");
string signature = HexEncode(HmacSha256(kSigning, stringToSign));
cout << signature << endl;

// ************* 步骤 4: 拼接 Authorization *************
```

```
string authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
cout << authorization << endl;

string curlcmd = "curl -X POST https://" + host + "\n"
+ " -H \"Authorization: " + authorization + "\"\n"
+ " -H \"Content-Type: application/json; charset=utf-8\"" + "\n"
+ " -H \"Host: " + host + "\"\n"
+ " -H \"X-TC-Action: " + action + "\"\n"
+ " -H \"X-TC-Timestamp: " + RequestTimestamp + "\"\n"
+ " -H \"X-TC-Version: " + version + "\"\n"
+ " -H \"X-TC-Region: " + region + "\"\n"
+ " -d '" + payload + "\'";
cout << curlcmd << endl;
return 0;
};
```

C

```
#include <ctype.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>

void get_utc_date(int64_t timestamp, char* utc, int len)
{
// time_t timenow;
struct tm sttime;
sttime = *gmtime(&timestamp);
strftime(utc, len, "%Y-%m-%d", &sttime);
}

void sha256_hex(const char* str, char* result)
{
char buf[3];
unsigned char hash[SHA256_DIGEST_LENGTH];
SHA256_CTX sha256;
SHA256_Init(&sha256);
SHA256_Update(&sha256, str, strlen(str));
SHA256_Final(hash, &sha256);
for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
{
snprintf(buf, sizeof(buf), "%02x", hash[i]);
strcat(result, buf);
}
}

void hmac_sha256(const char* key, int key_len,
const char* input, int input_len,
unsigned char* output, unsigned int* output_len)
{
HMAC_CTX *h;
```

```
#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif

HMAC_Init_ex(h, key, key_len, EVP_sha256(), NULL);
HMAC_Update(h, ( unsigned char* )input, input_len);
HMAC_Final(h, output, output_len);


#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

}

void hex_encode(const char* input, int input_len, char* output)
{
static const char* const lut = "0123456789abcdef";

char add_out[128] = {0};
char temp[2] = {0};
for (size_t i = 0; i < input_len; ++i)
{
const unsigned char c = input[i];
temp[0] = lut[c >> 4];
strcat(add_out, temp);
temp[0] = lut[c & 15];
strcat(add_out, temp);
}
strncpy(output, add_out, 128);
}

void lowercase(const char * src, char * dst)
{
for (int i = 0; src[i]; i++)
{
dst[i] = tolower(src[i]);
}
}

int main()
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
const char* SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const char* SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");
const char* service = "cvm";
const char* host = "cvm.tencentcloudapi.com";
const char* region = "ap-guangzhou";
const char* action = "DescribeInstances";
```

```
const char* version = "2017-03-12";
int64_t timestamp = 1551113065;
char date[20] = {0};
get_utc_date(timestamp, date, sizeof(date));

// ************* 步骤 1: 拼接规范请求串 *************
const char* http_request_method = "POST";
const char* canonical_uri = "/";
const char* canonical_query_string = "";
char canonical_headers[100] = {"content-type:application/json; charset=utf-8\nhost:"};
strcat(canonical_headers, host);
strcat(canonical_headers, "\nx-tc-action:");
char value[100] = {0};
lowercase(action, value);
strcat(canonical_headers, value);
strcat(canonical_headers, "\n");
const char* signed_headers = "content-type;host;x-tc-action";
const char* payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name
\"}]}";
char hashed_request_payload[100] = {0};
sha256_hex(payload, hashed_request_payload);

char canonical_request[256] = {0};
sprintf(canonical_request, "%s\n%s\n%s\n%s\n%s\n%s", http_request_method,
canonical_uri, canonical_query_string, canonical_headers,
signed_headers, hashed_request_payload);
printf("%s\n", canonical_request);


// ************* 步骤 2: 拼接待签名字符串 *************
const char* algorithm = "TC3-HMAC-SHA256";
char request_timestamp[16] = {0};
sprintf(request_timestamp, "%d", timestamp);
char credential_scope[64] = {0};
strcat(credential_scope, date);
sprintf(credential_scope, "%s/%s/tc3_request", date, service);
char hashed_canonical_request[100] = {0};
sha256_hex(canonical_request, hashed_canonical_request);
char string_to_sign[256] = {0};
sprintf(string_to_sign, "%s\n%s\n%s\n%s", algorithm, request_timestamp,
credential_scope, hashed_canonical_request);
printf("%s\n", string_to_sign);

// ************* 步骤 3: 计算签名 ***************
char k_key[64] = {0};
sprintf(k_key, "%s%s", "TC3", SECRET_KEY);
unsigned char k_date[64] = {0};
unsigned int output_len = 0;
hmac_sha256(k_key, strlen(k_key), date, strlen(date), k_date, &output_len);
unsigned char k_service[64] = {0};
hmac_sha256(k_date, output_len, service, strlen(service), k_service, &output_len);
unsigned char k_signing[64] = {0};
hmac_sha256(k_service, output_len, "tc3_request", strlen("tc3_request"), k_signing, &output_len);
unsigned char k_hmac_sha_sign[64] = {0};
hmac_sha256(k_signing, output_len, string_to_sign, strlen(string_to_sign), k_hmac_sha_sign, &output_len);

char signature[128] = {0};
```

```
hex_encode(k_hmac_sha_sign, output_len, signature);
printf("%s\n", signature);

// ************* 步骤 4: 拼接 Authorization *************
char authorization[512] = {0};
sprintf(authorization, "%s Credential=%s/%s, SignedHeaders=%s, Signature=%s",
algorithm, SECRET_ID, credential_scope, signed_headers, signature);
printf("%s\n", authorization);

char curlcmd[10240] = {0};
sprintf(curlcmd, "curl -X POST https://%s\n \
-H \"Authorization: %s\"\n \
-H \"Content-Type: application/json; charset=utf-8\"\n \
-H \"Host: %s\"\n \
-H \"X-TC-Action: %s\"\n \
-H \"X-TC-Timestamp: %s\"\n \
-H \"X-TC-Version: %s\"\n \
-H \"X-TC-Region: %s\"\n \
-d \'%s\'",
host, authorization, host, action, request_timestamp, version, region, payload);
printf("%s\n", curlcmd);
return 0;
}
```

其他语言

- Lua: GitHub
- Swift: GitHub
- Dart: GitHub
- Shell(Bash): GitHub

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

错误码	错误描述
AuthFailure.SignatureExpire	签名过期。Timestamp 与服务器接收到请求的时间相差不得超过五分钟。
AuthFailure.SecretIdNotFound	密钥不存在。请到控制台查看密钥是否被禁用，是否少复制了字符或者多了字符。
AuthFailure.SignatureFailure	签名错误。可能是签名计算错误，或者签名与实际发送的内容不相符合，也有可能是密钥 SecretKey 错误导致的。
AuthFailure.TokenFailure	临时证书 Token 错误。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。

签名方法

最近更新时间：2024-12-25 02:03:35

签名方法 v1 简单易用，但是功能和安全性都不如签名方法 v3，推荐使用签名方法 v3。

首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v1”，可以生成签名过程进行验证，并提供了部分编程语言的签名示例，也可直接生成 SDK 代码。推荐使用腾讯云 API 配套的 8 种常见的编程语言 SDK，已经封装了签名和请求过程，均已开源，支持 Python、Java、PHP、Go、NodeJS、.NET、C++、Ruby。

推荐使用 API Explorer　点击调试

您可以通过 API Explorer 的【签名串生成】模块查看每个接口签名的生成过程。

腾讯云 API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。
签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往 云API密钥页面 申请，否则无法调用云 API 接口。

1. 申请安全凭证

在第一次使用云 API 之前，请前往 云 API 密钥页面 申请安全凭证。
安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- **用户必须严格保管安全凭证，避免泄露。**

申请安全凭证的具体步骤如下：

1. 登录 腾讯云管理中心控制台 。
2. 前往 云 API 密钥 的控制台页面
3. 在 云 API 密钥 页面，单击【新建密钥】即可以创建一对 SecretId/SecretKey。

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

有了安全凭证 SecretId 和 SecretKey 后，就可以生成签名串了。以下是使用签名方法 v1 生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKID********************************
- SecretKey: ********************************

注意：这里只是示例，请根据用户实际申请的 SecretId 和 SecretKey 进行后续操作！

以云服务器查看实例列表（DescribeInstances）请求为例，当用户调用这一接口时，其请求参数可能如下：

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥 ID	AKID********************************
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例 ID	ins-09dx96dg
Offset	偏移量	0
Limit	最大允许输出	20
Version	接口版本号	2017-03-12

这里只展示了部分公共参数和接口输入参数，用户可以根据实际需要添加其他参数，例如 Language 和 Token 公共参数。

2.1. 对参数排序

首先对所有请求参数按参数名的字典序（ ASCII 码）升序排序。注意：1）只按参数名进行排序，参数值保持对应即可，不参与比大小；2）按 ASCII 码比大小，如 InstanceIds.2 要排在 InstanceIds.12 后面，不是按字母表，也不是按数值。用户可以借助编程语言中的相关排序函数来实现这一功能，如 PHP 中的 ksort 函数。上述示例参数的排序结果如下:

```
{
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : 'AKID********************************',
'Timestamp' : 1465185768,
'Version': '2017-03-12',
}
```

使用其它程序设计语言开发时，可对上面示例中的参数进行排序，得到的结果一致即可。

2.2. 拼接请求字符串

此步骤生成请求字符串。
将把上一步排序好的请求参数格式化成“参数名称=参数值”的形式，如对 Action 参数，其参数名称为 "Action" ，参数值为 "DescribeInstances" ，因此格式化后就为 Action=DescribeInstances 。
注意：“参数值”为原始值而非 url 编码后的值。

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID*******
************************&Timestamp=1465185768&Version=2017-03-12
```

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机:查看实例列表(DescribeInstances)的请求域名为：cvm.tencentcloudapi.com。实际的请求域名根据接口所属模块的不同而不同，详见各接口说明。
3. 请求路径: 当前版本云API的请求路径固定为 / 。
4. 请求字符串: 即上一步生成的请求字符串。

签名原文串的拼接规则为： 请求方法 + 请求主机 +请求路径 + ? + 请求字符串 。

示例的拼接结果为:

```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-gu
angzhou&SecretId=AKID********************************&Timestamp=1465185768&Version=2017-03-12
```

2.4. 生成签名串

此步骤生成签名串。
首先使用 HMAC-SHA1 算法对上一步中获得的**签名原文字符串**进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例:

```
$secretKey = '********************************';
$srcStr = 'GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&R
egion=ap-guangzhou&SecretId=AKID********************************&Timestamp=1465185768&Version=2017-03-12';
```

```
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

使用其它程序设计语言开发时，可用上面示例中的原文进行签名验证，得到的签名串与例子中的一致即可。

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 7RAM2xfNMO9EiVTNmPg06MRnCvQ= ，最终得到的签名串请求参数（Signature）为：7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D，它将用于生成最终的请求 URL。

注意：如果用户的请求方法是 GET，或者请求方法为 POST 同时 Content-Type 为 application/x-www-form-urlencoded，则发送请求时所有请求参数的值均需要做 URL 编码，参数键和=符号不需要编码。非 ASCII 字符在 URL 编码前需要先以 UTF-8 进行编码。

注意：有些编程语言的网络库会自动为所有参数进行 urlencode，在这种情况下，就不需要对签名串进行 URL 编码了，否则两次 URL 编码会导致签名失败。

注意：其他参数值也需要进行编码，编码采用 RFC 3986。使用 %XY 对特殊字符例如汉字进行百分比编码，其中“X”和“Y”为十六进制字符（0-9 和大写字母 A-F），使用小写将引发错误。

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

错误代码	错误描述
AuthFailure.SignatureExpire	签名过期
AuthFailure.SecretIdNotFound	密钥不存在
AuthFailure.SignatureFailure	签名错误
AuthFailure.TokenFailure	token 错误
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）

5. 签名演示

在实际调用 API 3.0 时，推荐使用配套的腾讯云 SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

下面提供了不同产品的生成签名 demo，您可以找到对应的产品参考签名的生成：

- Signature Demo

为了更清楚的解释签名过程，下面以实际编程语言为例，将上述的签名过程具体实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK 。

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-`

guangzhou&SecretId=AKID********************************&Signature=7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D&Timestamp=1465185768&Version=2017-03-12 。

注意：由于示例中的密钥是虚构的，时间戳也不是系统当前时间，因此如果将此 url 在浏览器中打开或者用 curl 等命令调用时会返回鉴权错误：签名过期。为了得到一个可以正常返回的 url ，需要修改示例中的 SecretId 和 SecretKey 为真实的密钥，并使用系统当前时间戳作为 Timestamp 。

注意：在下面的示例中，不同编程语言，甚至同一语言每次执行得到的 url 可能都有所不同，表现为参数的顺序不同，但这并不影响正确性。只要所有参数都在，且签名计算正确即可。

注意：以下代码仅适用于 API 3.0，不能直接用于其他的签名流程，请以对应的实际文档为准。

Java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TencentCloudAPIDemo {
private final static String CHARSET = "UTF-8";

public static String sign(String s, String key, String method) throws Exception {
Mac mac = Mac.getInstance(method);
SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAlgorithm());
mac.init(secretKeySpec);
byte[] hash = mac.doFinal(s.getBytes(CHARSET));
return DatatypeConverter.printBase64Binary(hash);
}

public static String getStringToSign(TreeMap<String, Object> params) {
StringBuilder s2s = new StringBuilder("GETcvm.tencentcloudapi.com/?");
// 签名时要求对参数进行字典排序，此处用TreeMap保证顺序
for (String k : params.keySet()) {
s2s.append(k).append("=").append(params.get(k).toString()).append("&");
}
return s2s.toString().substring(0, s2s.length() - 1);
}

public static String getUrl(TreeMap<String, Object> params) throws UnsupportedEncodingException {
StringBuilder url = new StringBuilder("https://cvm.tencentcloudapi.com/?");
// 实际请求的url中对参数顺序没有要求
for (String k : params.keySet()) {
// 需要对请求串进行urlencode，由于key都是英文字母，故此处仅对其value进行urlencode
url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).append("&");
}
return url.toString().substring(0, url.length() - 1);
}

public static void main(String[] args) throws Exception {
TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
// 实际调用时应当使用随机数，例如: params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
params.put("Nonce", 11886); // 公共参数
// 实际调用时应当使用系统当前时间，例如: params.put("Timestamp", System.currentTimeMillis() / 1000);
params.put("Timestamp", 1465185768); // 公共参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
params.put("SecretId", System.getenv("TENCENTCLOUD_SECRET_ID")); // 公共参数
```

```
params.put("Action", "DescribeInstances"); // 公共参数
params.put("Version", "2017-03-12"); // 公共参数
params.put("Region", "ap-guangzhou"); // 公共参数
params.put("Limit", 20); // 业务参数
params.put("Offset", 0); // 业务参数
params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
params.put("Signature", sign(getStringToSign(params), System.getenv("TENCENTCLOUD_SECRET_KEY"), "HmacSHA1")); // 公共参数
System.out.println(getUrl(params));
}
}
```

Python

注意：如果是在 Python 2 环境中运行，需要先安装 requests 依赖包： `pip install requests` 。

```
# -*- coding: utf8 -*-
import base64
import hashlib
import hmac
import os
import time

import requests

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secret_key = os.environ.get("TENCENTCLOUD_SECRET_KEY")

def get_string_to_sign(method, endpoint, params):
s = method + endpoint + "/?"
query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
return s + query_str

def sign_str(key, s, method):
hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
return base64.b64encode(hmac_str)

if __name__ == '__main__':
endpoint = "cvm.tencentcloudapi.com"
data = {
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : secret_id,
'Timestamp' : 1465185768, # int(time.time())
'Version': '2017-03-12'
}
s = get_string_to_sign("GET", endpoint, data)
data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
print(data["Signature"])
# 此处会实际调用，成功后可能产生计费
```

```
# resp = requests.get("https://" + endpoint, params=data)
# print(resp.url)
```

Golang

```
package main

import (
"bytes"
"crypto/hmac"
"crypto/sha1"
"encoding/base64"
"fmt"
"os"
"sort"
"strconv"
)

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
params := map[string]string{
"Nonce": "11886",
"Timestamp": strconv.Itoa(1465185768),
"Region": "ap-guangzhou",
"SecretId": secretId,
"Version": "2017-03-12",
"Action": "DescribeInstances",
"InstanceIds.0": "ins-09dx96dg",
"Limit": strconv.Itoa(20),
"Offset": strconv.Itoa(0),
}

var buf bytes.Buffer
buf.WriteString("GET")
buf.WriteString("cvm.tencentcloudapi.com")
buf.WriteString("/")
buf.WriteString("?")

// sort keys by ascii asc order
keys := make([]string, 0, len(params))
for k, _ := range params {
keys = append(keys, k)
}
sort.Strings(keys)

for i := range keys {
k := keys[i]
buf.WriteString(k)
buf.WriteString("=")
buf.WriteString(params[k])
buf.WriteString("&")
}
buf.Truncate(buf.Len() - 1)
```

```
hashed := hmac.New(sha1.New, []byte(secretKey))
hashed.Write(buf.Bytes())

fmt.Println(base64.StdEncoding.EncodeToString(hashed.Sum(nil)))
}
```

PHP

```
<?php
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$param["Nonce"] = 11886;//rand();
$param["Timestamp"] = 1465185768;//time();
$param["Region"] = "ap-guangzhou";
$param["SecretId"] = $secretId;
$param["Version"] = "2017-03-12";
$param["Action"] = "DescribeInstances";
$param["InstanceIds.0"] = "ins-09dx96dg";
$param["Limit"] = 20;
$param["Offset"] = 0;

ksort($param);

$signStr = "GETcvm.tencentcloudapi.com/?";
foreach ( $param as $key => $value ) {
$signStr = $signStr . $key . "=" . $value . "&";
}
$signStr = substr($signStr, 0, -1);

$signature = base64_encode(hash_hmac("sha1", $signStr, $secretKey, true));
echo $signature.PHP_EOL;
// need to install and enable curl extension in php.ini
// $param["Signature"] = $signature;
// $url = "https://cvm.tencentcloudapi.com/?".http_build_query($param);
// echo $url.PHP_EOL;
// $ch = curl_init();
// curl_setopt($ch, CURLOPT_URL, $url);
// $output = curl_exec($ch);
// curl_close($ch);
// echo json_decode($output);
```

Ruby

```
# -*- coding: UTF-8 -*-
# require ruby>=2.3.0
require 'time'
require 'openssl'
require 'base64'

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
```

```
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

method = 'GET'
endpoint = 'cvm.tencentcloudapi.com'
data = {
'Action' => 'DescribeInstances',
'InstanceIds.0' => 'ins-09dx96dg',
'Limit' => 20,
'Nonce' => 11886,
'Offset' => 0,
'Region' => 'ap-guangzhou',
'SecretId' => secret_id,
'Timestamp' => 1465185768, # Time.now.to_i
'Version' => '2017-03-12',
}
sign = method + endpoint + '/?'
params = []
data.sort.each do |item|
params << "#{item[0]}=#{item[1]}"
end
sign += params.join('&')
digest = OpenSSL::Digest.new('sha1')
data['Signature'] = Base64.encode64(OpenSSL::HMAC.digest(digest, secret_key, sign))
puts data['Signature']

# require 'net/http'
# uri = URI('https://' + endpoint)
# uri.query = URI.encode_www_form(data)
# p uri
# res = Net::HTTP.get_response(uri)
# puts res.body
```

DotNet

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Security.Cryptography;
using System.Text;

public class Application {
public static string Sign(string signKey, string secret)
{
string signRet = string.Empty;
using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
{
byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
signRet = Convert.ToBase64String(hash);
}
return signRet;
}
public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, string reques
tHost, string requestPath)
{
string retStr = "";
retStr += requestMethod;
```

```
retStr += requestHost;
retStr += requestPath;
retStr += "?";
string v = "";
foreach (string key in requestParams.Keys)
{
v += string.Format("{0}={1}&", key, requestParams[key]);
}
retStr += v.TrimEnd('&');
return retStr;
}

public static void Main(string[] args)
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

string endpoint = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";
double RequestTimestamp = 1465185768; // 时间戳 2019-02-26 00:44:25,此参数作为示例，以实际为准
// long timestamp = ToTimestamp() / 1000;
// string requestTimestamp = timestamp.ToString();
Dictionary<string, string> param = new Dictionary<string, string>();
param.Add("Limit", "20");
param.Add("Offset", "0");
param.Add("InstanceIds.0", "ins-09dx96dg");
param.Add("Action", action);
param.Add("Nonce", "11886");
// param.Add("Nonce", Math.Abs(new Random().Next()).ToString());

param.Add("Timestamp", RequestTimestamp.ToString());
param.Add("Version", version);

param.Add("SecretId", SECRET_ID);
param.Add("Region", region);
SortedDictionary<string, string> headers = new SortedDictionary<string, string>(param, StringComparer.Ordinal);
string sigInParam = MakeSignPlainText(headers, "GET", endpoint, "/");
string sigOutParam = Sign(SECRET_KEY, sigInParam);
Console.WriteLine(sigOutParam);
}
}
```

NodeJS

```
const crypto = require('crypto');

function get_req_url(params, endpoint){
params['Signature'] = encodeURIComponent(params['Signature']);
const url_strParam = sort_params(params)
return "https://" + endpoint + "/?" + url_strParam.slice(1);
}
```

```
function formatSignString(reqMethod, endpoint, path, strParam){
let strSign = reqMethod + endpoint + path + "?" + strParam.slice(1);
return strSign;
}
function sha1(secretKey, strsign){
let signMethodMap = {'HmacSHA1': "sha1"};
let hmac = crypto.createHmac(signMethodMap['HmacSHA1'], secretKey || "");
return hmac.update(Buffer.from(strsign, 'utf8')).digest('base64')
}

function sort_params(params){
let strParam = "";
let keys = Object.keys(params);
keys.sort();
for (let k in keys) {
//k = k.replace(/_/g, '.');
strParam += ("&" + keys[k] + "=" + params[keys[k]]);
}
return strParam
}

function main(){
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

const endpoint = "cvm.tencentcloudapi.com"
const Region = "ap-guangzhou"
const Version = "2017-03-12"
const Action = "DescribeInstances"
const Timestamp = 1465185768 // 时间戳 2016-06-06 12:02:48，此参数作为示例，以实际为准
// const Timestamp = Math.round(Date.now() / 1000)
const Nonce = 11886 // 随机正整数
//const nonce = Math.round(Math.random() * 65535)

let params = {};
params['Action'] = Action;
params['InstanceIds.0'] = 'ins-09dx96dg';
params['Limit'] = 20;
params['Offset'] = 0;
params['Nonce'] = Nonce;
params['Region'] = Region;
params['SecretId'] = SECRET_ID;
params['Timestamp'] = Timestamp;
params['Version'] = Version;

// 1. 对参数排序，并拼接请求字符串
strParam = sort_params(params)

// 2. 拼接签名原文字符串
const reqMethod = "GET";
const path = "/";
strSign = formatSignString(reqMethod, endpoint, path, strParam)
// console.log(strSign)
```

```
// 3. 生成签名串
params['Signature'] = sha1(SECRET_KEY, strSign)
console.log(params['Signature'])

// 4. 进行url编码并拼接请求url
// const req_url = get_req_url(params, endpoint)
// console.log(params['Signature'])
// console.log(req_url)
}
main()
```

返回结果

最近更新时间：2024-03-12 19:53:00

云 API 3.0 接口默认返回 JSON 数据，返回非 JSON 格式的接口会在文档中做出说明。返回 JSON 数据时最大限制为 50 MB，如果返回的数据超过最大限制，请求会失败并返回内部错误。请根据接口文档中给出的过滤功能（例如时间范围）或者分页功能，控制返回数据不要过大。

注意：目前只要请求被服务端正常处理了，响应的 HTTP 状态码均为200。例如返回的消息体里的错误码是签名失败，但 HTTP 状态码是200，而不是401。

正确返回结果

以云服务器的接口查看实例状态列表 (DescribeInstancesStatus) 2017-03-12 版本为例，若调用成功，其可能的返回如下为：

```
{
"Response": {
"TotalCount": 0,
"InstanceStatusSet": [],
"RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
}
}
```

- Response 及其内部的 RequestId 是固定的字段，无论请求成功与否，只要 API 处理了，则必定会返回。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系 腾讯云客服 或 提交工单，并提供该 ID 来解决问题。
- 除了固定的字段外，其余均为具体接口定义的字段，不同的接口所返回的字段参见接口文档中的定义。此例中的 TotalCount 和 InstanceStatusSet 均为 DescribeInstancesStatus 接口定义的字段，由于调用请求的用户暂时还没有云服务器实例，因此 TotalCount 在此情况下的返回值为 0， InstanceStatusSet 列表为空。

错误返回结果

若调用失败，其返回值示例如下为：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

- Error 的出现代表着该请求调用失败。Error 字段连同其内部的 Code 和 Message 字段在调用失败时是必定返回的。
- Code 表示具体出错的错误码，当请求出错时可以先根据该错误码在公共错误码和当前接口对应的错误码列表里面查找对应原因和解决方案。
- Message 显示出了这个错误发生的具体原因，随着业务发展或体验优化，此文本可能会经常保持变更或更新，用户不应依赖这个返回值。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系 腾讯云客服 或 提交工单，并提供该 ID 来解决问题。

公共错误码

返回结果中如果存在 Error 字段，则表示调用 API 接口失败。 Error 中的 Code 字段表示错误码，所有业务都可能出现的错误码为公共错误码。完整的错误码列表请参考本产品“API 文档”目录下的“错误码”页面。

参数类型

最近更新时间：2022-08-10 06:08:58

目前腾讯云 API 3.0 输入参数和输出参数支持如下几种数据格式：

- String: 字符串。
- Integer：整型，上限为无符号64位整数。SDK 3.0 不同编程语言支持的类型有所差异，建议以所使用编程语言的最大整型定义，例如 `Golang` 的 `uint64`。
- Boolean：布尔型。
- Float：浮点型。
- Double：双精度浮点型。
- Date：字符串，日期格式。例如：2022-01-01。
- Timestamp：字符串，时间格式。例如：2022-01-01 00:00:00。
- Timestamp ISO8601：ISO 8601 是由国际标准化组织（International Organization for Standardization，ISO）发布的关于日期和时间格式的国际标准，对应国标《GB/T 7408-2005数据元和交换格式信息交换日期和时间表示法》。建议以所使用编程语言的标准库进行格式解析。例如：2022-01-01T00:00:00+08:00。
- Binary：二进制内容，需要以特定协议请求和解析。

集群相关接口
删除集群

最近更新时间：2025-04-25 02:01:27

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（DeleteCluster）用于删除一个指定的集群。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteCluster。
Version	是	String	公共参数，本接口取值：2021-11-09。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除集群

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DeleteCluster
&ClusterId=hpc-5lyv94lq
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "8b71050a-55c1-4f3b-bf66-5e4f8510a5a0"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
ResourceNotFound.ClusterId	集群不存在。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

创建集群

最近更新时间：2025-05-15 02:06:25

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (CreateCluster) 用于创建并启动集群。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateCluster。
Version	是	String	公共参数，本接口取值：2021-11-09。
Region	是	String	公共参数，详见产品支持的 地域列表。
Placement	是	Placement	集群中实例所在的位置。 示例值：{ "Zone": "ap-guangzhou-6" }
ManagerNode	否	ManagerNode	指定管理节点。 示例值：{"InstanceType": "S2.SMALL2"}
ManagerNodeCount	否	Integer	指定管理节点的数量。默认取值：1。取值范围：1~2。 示例值：1
ComputeNode	否	ComputeNode	指定计算节点。 示例值：{"InstanceType": "S2.SMALL2"}
ComputeNodeCount	否	Integer	指定计算节点的数量。默认取值：0。 示例值：1
SchedulerType	否	String	调度器类型。 • SLURM：SLURM调度器。 示例值：SLURM
ImageId	否	String	指定有效的镜像ID，格式形如img-xxx。目前仅支持公有镜像。 示例值：img-xxx
VirtualPrivateCloud	否	VirtualPrivateCloud	私有网络相关信息配置。 示例值：{ "VpcId": "vpc-rhfaxx31", "SubnetId": "subnet-x7vxgqe" }
LoginSettings	否	LoginSettings	集群登录设置。 示例值：{ "KeyIds": ["skey-9tzxxb4j"] }
SecurityGroupIds.N	否	Array of String	集群中实例所属安全组。该参数可以通过调用 DescribeSecurityGroups 的返回值中的sgId字段来获 则绑定默认安全组。 示例值：["sg-ajhn9qtq"]
ClientToken	否	String	用于保证请求幂等性的字符串。该字符串由客户生成，需保证不同请求之间唯一，最大值不超过64个AS 数，则无法保证请求的幂等性。 示例值：system-f3827db9-c58a-49cc-bf10-33fc1923a34a
DryRun	否	Boolean	是否只预检此次请求。 true：发送检查请求，不会创建实例。检查项包括是否填写了必需参数，请求格式，业务限制和云服务器 如果检查不通过，则返回对应错误码；

参数名称	必选	类型	描述
			如果检查通过，则返回RequestId. false（默认）：发送正常请求，通过检查后直接创建实例 示例值：False
AccountType	否	String	域名字服务类型。默认值：NIS • NIS：NIS域名字服务。 示例值：NIS
ClusterName	否	String	集群显示名称。 示例值：cluster-test
StorageOption	否	StorageOption	集群存储选项 示例值：{"CFSOptions": [{"LocalPath":"/mnt/","RemotePath":"127.0.0.1@tcp0:/test/cfs/","Protocol":"TURBO","S
LoginNode.N	否	Array of LoginNode	已废弃。 指定登录节点。 示例值：{"InstanceType": "S2.SMALL2"}
LoginNodeCount	否	Integer	已废弃。 指定登录节点的数量。默认取值：0。取值范围：0~10。 示例值：1
Tags.N	否	Array of Tag	创建集群时同时绑定的标签对说明。 示例值：[{"Key": "test", "Value": "test"}]

3. 输出参数

参数名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-5lyv94lq
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建集群

输入示例

```
https://thpc.tencentcloudapi.com/?Action=CreateCluster
&Placement.Zone=ap-guangzhou-2
&SchedulerType=SLURM
&ImageId=img-3la7wgnt
&ManagerNode.InstanceType=S2.SMALL2
&ManagerNodeCount=1
&ComputeNode.InstanceType=S2.SMALL2
&ComputeNode.InstanceChargeType=SPOTPAID
&ComputeNodeCount=2
&<公共请求参数>
```

输出示例

```
{
"Response": {
"ClusterId": "hpc-5lyv94lq",
"RequestId": "b2ac2379-6453-4eab-8f63-7ade00cb67b0"
```

```
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.CallCvm	cvm调用失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooSmall	参数值过小。

查询集群列表

最近更新时间：2025-04-25 02:01:27

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（DescribeClusters）用于查询集群列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeClusters。
Version	是	String	公共参数，本接口取值：2021-11-09。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterIds.N	否	Array of String	集群ID列表。通过该参数可以指定需要查询信息的集群列表。 如果您不指定该参数，则返回Limit数量以内的集群信息。 示例值：["hpc-xxx"]
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：20

3. 输出参数

参数名称	类型	描述
ClusterSet	Array of ClusterOverview	集群概览信息列表。 示例值：[{"ClusterId":"hpc-qrb7ybn0","ClusterName":"unnamed","ClusterStatus":"RUNNING"}]
TotalCount	Integer	集群数量。 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询集群列表

查询集群列表。

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DescribeClusters
&Offset=0
&Limit=1
&<公共请求参数>
```

输出示例


```
{
"Response": {
"ClusterSet": [
{
"ClusterId": "hpc-qrb7ybn0",
"ClusterName": "unnamed",
"ClusterStatus": "RUNNING",
"Placement": {
"Zone": "ap-guangzhou-2"
},
"CreateTime": "2021-12-07T03:29:09Z",
"SchedulerType": "SGE",
"ComputeNodeCount": 1,
"ComputeNodeSet": [
{
"NodeId": "ins-jfhc307q"
}
],
"ManagerNodeCount": 1,
"ManagerNodeSet": [
{
"NodeId": "ins-cv6ygpkq"
}
],
"LoginNodeCount": 1,
"LoginNodeSet": [
{
"NodeId": "ins-cv6ygpkx"
}
]
}
],
"TotalCount": 1,
"RequestId": "8a39c8e3-129b-4628-b763-ef96caaa2f4d"
}
}
```


5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee

- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。

弹性伸缩相关接口
绑定弹性伸缩组

最近更新时间：2025-04-25 02:01:27

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(BindAutoScalingGroup)用于为集群队列绑定弹性伸缩组

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：BindAutoScalingGroup。
Version	是	String	公共参数，本接口取值：2021-11-09。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
LaunchConfigurationId	是	String	弹性伸缩启动配置ID。 示例值：asc-jhf9c1gi
AutoScalingGroupId	是	String	弹性伸缩组ID。 示例值：asg-aesc6pcq
QueueName	否	String	队列名称。 示例值：all.q
ExpansionBusyTime	否	Integer	任务连续等待时间，队列的任务处于连续等待的时间。单位秒。默认值120。 示例值：120
ShrinkIdleTime	否	Integer	节点连续空闲（未运行作业）时间，一个节点连续处于空闲状态时间。单位秒。默认值300。 示例值：300
EnableAutoExpansion	否	Boolean	是否开启自动扩容，默认值true。 示例值：True
EnableAutoShrink	否	Boolean	是否开启自动缩容，默认值true。 示例值：True
DryRun	否	Boolean	是否只预检此次请求。 true：发送检查请求，不会绑定弹性伸缩组。检查项包括是否填写了必需参数，请求格式，业务限制。 如果检查不通过，则返回对应错误码； 如果检查通过，则返回RequestId。 false（默认）：发送正常请求，通过检查后直接绑定弹性伸缩组。 示例值：False

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需

参数名称	类型	描述
		要提供该次请求的 RequestId。

4. 示例

示例1 绑定弹性伸缩组

输入示例

```
https://thpc.tencentcloudapi.com/?Action=BindAutoScalingGroup
&ClusterId=hpc-5lyv94lq
&LaunchConfigurationId=asc-jhf9c1gi
&AutoScalingGroupId=asg-aesc6pcq
&ExpansionBusyTime=120
&ShrinkIdleTime=300
&DryRun=false
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "f4d90874-f565-463d-ba1d-c707517eeaa1"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameterValue.TooSmall	参数值过小。

错误码	描述
ResourceNotFound.AutoScalingGroupId	无法找到弹性伸缩组ID。
ResourceNotFound.LaunchConfigurationId	无法找到ID对应的弹性伸缩启动配置。
UnsupportedOperation.AutoScalingGroupAlreadyBinded	该伸缩组已绑定集群，请更换伸缩组。
UnsupportedOperation.BindAutoScalingGroup	指定的集群或集群队列当前不支持绑定弹性伸缩组。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

数据结构

最近更新时间：2025-05-14 02:04:15

CFSOption

描述CFS文件系统版本和挂载信息

被如下接口引用：CreateCluster。

名称	类型	必选	描述
LocalPath	String	是	文件系统本地挂载路径 示例值：/data
RemotePath	String	是	文件系统远程挂载ip及路径 示例值：172.0.0.1:/
Protocol	String	否	文件系统协议类型，默认值NFS 3.0。 • NFS 3.0。 • NFS 4.0。 • TURBO。 示例值：NFS 3.0
StorageType	String	否	文件系统存储类型，默认值SD；其中 SD 为通用标准型标准型存储， HP为通用性能型存储， TB为turbo标准型， TP 为turbo性能型。 示例值：SD

ClusterOverview

集群概览信息。

被如下接口引用：DescribeClusters。

名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-2mv1i3d8
ClusterStatus	String	集群状态。取值范围： • PENDING：创建中 • INITING：初始化中 • INIT_FAILED：初始化失败 • RUNNING：运行中 • TERMINATING：销毁中 示例值：RUNNING
ClusterName	String	集群名称。 示例值：hpc-test
Placement	Placement	集群位置信息。 示例值：{ "Zone": "ap-guangzhou-6" }
CreateTime	Timestamp ISO8601	集群创建时间。 示例值：2024-12-18T13:46:34Z
SchedulerType	String	集群调度器。 示例值：SLURM
ComputeNodeCount	Integer	计算节点数量。 示例值：1
ComputeNodeSet	Array of ComputeNodeOverview	计算节点概览。 示例值：[{"NodeId":"ins-riredadfj6"},{"NodeId":"ins-mdafdsai"},{"NodeId":"ins-

名称	类型	描述
		5icofadac"},{"NodeId":"ins-icfdadfy"}]
ManagerNodeCount	Integer	管控节点数量。 示例值：1
ManagerNodeSet	Array of ManagerNodeOverview	管控节点概览。 示例值：[{"NodeId":"node-rirfdafa"}]
LoginNodeSet	Array of LoginNodeOverview	登录节点概览。 示例值：[{"NodeId":"ins-riredadfj6"},{"NodeId":"ins-mdafdsai"},{"NodeId":"ins-5icofadac"},{"NodeId":"ins-icfdadfy"}]
LoginNodeCount	Integer	登录节点数量。 示例值：1

ComputeNode

计算节点信息。

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
SystemDisk	SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值： {"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
InternetAccessible	InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}
InstanceName	String	否	节点显示名称。 • 不指定节点显示名称则默认显示‘未命名’。 最多支持60个字符。 示例值：未命名

ComputeNodeOverview

计算节点概览。

被如下接口引用：DescribeClusters。

名称	类型	描述
NodeId	String	计算节点ID。 示例值：ins-jfhc307q

DataDisk

描述了数据盘的信息

被如下接口引用：CreateCluster。

名称	类型	必选	描述
DiskSize	Integer	是	数据盘大小，单位：GB。最小调整步长为10G，不同数据盘类型取值范围不同，具体限制详见：存储概述。默认值为0，表示不购买数据盘。更多限制详见产品文档。 示例值：0
DiskType	String	否	数据盘类型。数据盘类型限制详见存储概述。取值范围： • LOCAL_BASIC：本地硬盘 • LOCAL_SSD：本地SSD硬盘 • LOCAL_NVME：本地NVME硬盘，与InstanceType强相关，不支持指定 • LOCAL_PRO：本地HDD硬盘，与InstanceType强相关，不支持指定 • CLOUD_BASIC：普通云硬盘 • CLOUD_PREMIUM：高性能云硬盘 • CLOUD_SSD：SSD云硬盘 • CLOUD_HSSD：增强型SSD云硬盘 • CLOUD_TSSD：极速型SSD云硬盘 默认取值：LOCAL_BASIC。 示例值：LOCAL_BASIC

GooseFSOption

描述GooseFS挂载信息

被如下接口引用：CreateCluster。

名称	类型	必选	描述
LocalPath	String	是	文件系统本地挂载路径 示例值：/data
RemotePath	String	是	文件系统远程挂载路径 示例值：/data/goosefs-fuse
Masters	Array of String	是	文件系统master的ip和端口 示例值：["172.16.0.2:55533","172.16.0.3:55533","172.16.0.4:55533"]

InstanceChargePrepaid

描述了实例的计费模式

被如下接口引用：CreateCluster。

名称	类型	必选	描述
Period	Integer	是	购买实例的时长，单位：月。取值范围：1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 36, 48, 60。 示例值：1
RenewFlag	String	否	自动续费标识。取值范围： NOTIFY_AND_AUTO_RENEW：通知过期且自动续费 NOTIFY_AND_MANUAL_RENEW：通知过期不自动续费 DISABLE_NOTIFY_AND_MANUAL_RENEW：不通知过期不自动续费 默认取值：NOTIFY_AND_MANUAL_RENEW。若该参数指定为NOTIFY_AND_AUTO_RENEW，在账户余额充足

名称	类型	必选	描述
			的情况下，实例到期后将按月自动续费。 示例值：NOTIFY_AND_AUTO_RENEW

InternetAccessible

描述了实例的公网可访问性，声明了实例的公网使用计费模式，最大带宽等

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InternetChargeType	String	否	网络计费类型。取值范围： BANDWIDTH_PREPAID：预付费按带宽结算 TRAFFIC_POSTPAID_BY_HOUR：流量按小时后付费 BANDWIDTH_POSTPAID_BY_HOUR：带宽按小时后付费 BANDWIDTH_PACKAGE：带宽包用户 默认取值：非带宽包用户默认与子机付费类型保持一致。 示例值：TRAFFIC_POSTPAID_BY_HOUR
InternetMaxBandwidthOut	Integer	否	公网出带宽上限，单位：Mbps。默认值：0Mbps。不同机型带宽上限范围不一致，具体限制详见购买网络带宽。 示例值：0

LoginNode

登录节点信息。

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
SystemDisk	Array of SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值： {"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
InternetAccessible	Array of InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}
InstanceName	String	否	节点显示名称。 • 不指定节点显示名称则默认显示‘未命名’。最多支持60个字符。

名称	类型	必选	描述
			示例值：未命名

LoginNodeOverview

登录节点概览。

被如下接口引用：DescribeClusters。

名称	类型	描述
NodeId	String	登录节点ID。 示例值：ins-jfhc307q

LoginSettings

描述了实例登录相关配置与信息。

被如下接口引用：CreateCluster。

名称	类型	必选	描述
Password	String	否	实例登录密码。不同操作系统类型密码复杂度限制不一样，具体如下： • Linux实例密码必须8到30位，至少包括两项[a-z]，[A-Z]、[0-9] 和 [( ) ` ~ ! @ # $ % ^ & * - + = | { } [ ] : ; ' , . ? / ]中的特殊符号。 • Windows实例密码必须12到30位，至少包括三项[a-z]，[A-Z]，[0-9] 和 [ ( ) ` ~ ! @ # $ % ^ & * - + = | { } [ ] : ; ' , . ? /] 中的特殊符号。 若不指定该参数，则由系统随机生成密码，并通过站内信方式通知到用户。 示例值：test@123456

ManagerNode

管控节点信息

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
SystemDisk	SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值： {"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}

名称	类型	必选	描述
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
InternetAccessible	InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}
InstanceName	String	否	节点显示名称。 • 不指定节点显示名称则默认显示‘未命名’。 • 购买多个节点，如果指定模式串{R:x}，表示生成数字[x, x+n-1]，其中n表示购买节点的数量，例如server_{R:3}，购买1个时，节点显示名称为server_3；购买2个时，节点显示名称分别为server_3，server_4。支持指定多个模式串{R:x}。购买多个节点，如果不指定模式串，则在节点显示名称添加后缀1、2...n，其中n表示购买节点的数量，例如server_，购买2个时，节点显示名称分别为server_1，server_2。 • 最多支持60个字符（包含模式串）。 示例值：未命名

ManagerNodeOverview

管控节点概览。

被如下接口引用：DescribeClusters。

名称	类型	描述
NodeId	String	管控节点ID。 示例值：ins-jfhc307q

Placement

描述了实例的抽象位置

被如下接口引用：CreateCluster, DescribeClusters。

名称	类型	必选	描述
Zone	String	是	实例所属的可用区名称。该参数可以通过调用 DescribeZones 的返回值中的Zone字段来获取。 示例值：ap-guangzhou-1

StorageOption

描述集群文件系统选项

被如下接口引用：CreateCluster。

名称	类型	必选	描述
CFSOptions	Array of CFSOption	否	集群挂载CFS文件系统选项
GooseFSOptions	Array of GooseFSOption	否	集群挂在GooseFS文件系统选项

SystemDisk

描述了操作系统所在块设备即系统盘的信息

被如下接口引用：CreateCluster。

名称	类型	必选	描述
DiskType	String	否	系统盘类型。系统盘类型限制详见存储概述。取值范围： LOCAL_BASIC：本地硬盘

名称	类型	必选	描述
			LOCAL_SSD：本地SSD硬盘 CLOUD_BASIC：普通云硬盘 CLOUD_SSD：SSD云硬盘 CLOUD_PREMIUM：高性能云硬盘 默认取值：当前有库存的硬盘类型。 示例值：CLOUD_BASIC
DiskSize	Integer	否	系统盘大小，单位：GB。默认值为 50 示例值：50

Tag

标签键值对。

被如下接口引用：CreateCluster。

名称	类型	必选	描述
Key	String	是	标签键 示例值：env
Value	String	是	标签值 示例值：product

VirtualPrivateCloud

描述了VPC相关信息

被如下接口引用：CreateCluster。

名称	类型	必选	描述
VpcId	String	是	私有网络ID，形如vpc-xxx。有效的VpcId可通过登录控制台查询；也可以调用接口 DescribeVpcEx ，从接口返回中的unVpcId字段获取。若在创建子机时VpcId与SubnetId同时传入DEFAULT，则强制使用默认vpc网络。 示例值：vpc-jinanlfe
SubnetId	String	是	私有网络子网ID，形如subnet-xxx。有效的私有网络子网ID可通过登录控制台查询；也可以调用接口 DescribeSubnets ，从接口返回中的unSubnetId字段获取。若在创建子机时SubnetId与VpcId同时传入DEFAULT，则强制使用默认vpc网络。 示例值：subnet-jknfane

错误码

最近更新时间：2024-12-20 01:57:56

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

Error 中的 Code 表示错误码，Message 表示该错误的具体信息。

错误码列表

公共错误码

错误码	说明
ActionOffline	接口已下线。
AuthFailure.InvalidAuthorization	请求头部的 `Authorization` 不符合腾讯云标准。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。
AuthFailure.MFAFailure	MFA 错误。
AuthFailure.SecretIdNotFound	密钥不存在。请在 控制台 检查密钥是否已被删除或者禁用，如状态正常，请检查密钥是否填写正确，注意前后不得有空格。
AuthFailure.SignatureExpire	签名过期。Timestamp 和服务器时间相差不得超过五分钟，请检查本地时间是否和标准时间同步。
AuthFailure.SignatureFailure	签名错误。签名计算错误，请对照调用方式中的签名方法文档检查签名计算过程。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未授权。请参考 CAM 文档对鉴权的说明。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误（包括参数格式、类型等错误）。
InvalidParameterValue	参数取值错误。
InvalidRequest	请求 body 的 multipart 格式错误。
IpInBlacklist	IP 地址在黑名单中。
IpNotInWhitelist	IP 地址不在白名单中。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数。

错误码	说明
NoSuchProduct	产品不存在
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
RequestLimitExceeded.GlobalRegionUinLimitExceeded	主账号超过频率限制。
RequestLimitExceeded.IPLimitExceeded	IP 限频。
RequestLimitExceeded.UinLimitExceeded	主账号限频。
RequestSizeLimitExceeded	请求包超过限制大小。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
ResponseSizeLimitExceeded	返回包超过限制大小。
ServiceUnavailable	当前服务暂时不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误，用户多传未定义的参数会导致错误。
UnsupportedOperation	操作不支持。
UnsupportedProtocol	http(s) 请求协议错误，只支持 GET 和 POST 请求。
UnsupportedRegion	接口不支持所传地域。

业务错误码

错误码	说明
InternalError.CallCvm	cvm调用失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooSmall	参数值过小。
ResourceNotFound.AutoScalingGroupId	无法找到弹性伸缩组ID。
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.LaunchConfigurationId	无法找到ID对应的弹性伸缩启动配置。
UnsupportedOperation.AutoScalingGroupAlreadyBinded	该伸缩组已绑定集群，请更换伸缩组。
UnsupportedOperation.BindAutoScalingGroup	指定的集群或集群队列当前不支持绑定弹性伸缩组。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

高性能计算平台 API 2022-04-01
产品版本

最近更新时间：2023-03-30 02:29:54

您正在浏览thpc产品文档的版本: 2022-04-01

最新版本

- 2023-03-21

以往版本

- 2021-11-09
- 2022-04-01 (当前版本)

更新历史

最近更新时间：2025-03-25 02:09:52

第 14 次发布

发布时间：2025-03-25 02:09:47

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- QueueOverview
  - **修改成员：** QueueName

第 13 次发布

发布时间：2024-12-31 01:30:33

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- QueueConfigOverview
  - **修改成员：** QueueName, MinSize, MaxSize, EnableAutoExpansion, EnableAutoShrink, ExpansionNodeConfigs

第 12 次发布

发布时间：2023-02-27 01:46:46

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AddQueue
- DeleteQueue
- DescribeNodes
- DescribeQueues

新增数据结构：

- Filter
- NodeOverview
- QueueOverview

第 11 次发布

发布时间：2023-02-22 01:58:40

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- AddNodes
  - 新增入参：NodeType

第 10 次发布

发布时间：2023-02-08 02:08:58

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- AddNodes
  - **修改入参：** ImageId

第 9 次发布

发布时间：2023-01-11 01:46:25

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeAutoScalingConfiguration

新增数据结构：

- ExpansionNodeConfigOverview
- QueueConfigOverview

第 8 次发布

发布时间：2023-01-10 01:50:30

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- DescribeClusterStorageOption
  - 新增出参：StorageOption

新增数据结构：

- CFSOptionOverview
- GooseFSOptionOverview
- StorageOptionOverview

第 7 次发布

发布时间：2022-12-21 02:09:45

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AddClusterStorageOption
- DeleteClusterStorageOption
- DescribeClusterStorageOption

第 6 次发布

发布时间：2022-11-16 06:42:38

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- DescribeClusterActivities

新增数据结构：

- ClusterActivity
- NodeActivity

第 5 次发布

发布时间：2022-11-04 06:42:08

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- CreateCluster
  - 新增入参：AutoScalingType

第 4 次发布

发布时间：2022-10-27 07:00:42

本次发布包含了以下内容：

改善已有的文档。

修改数据结构：

- ClusterOverview
  - 新增成员：VpcId

第 3 次发布

发布时间：2022-10-21 06:49:31

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- SetAutoScalingConfiguration

新增数据结构：

- ExpansionNodeConfig
- QueueConfig

第 2 次发布

发布时间：2022-04-27 10:43:23

本次发布包含了以下内容：

改善已有的文档。

修改接口：

- BindAutoScalingGroup
  - 新增入参：QueueName

第 1 次发布

发布时间：2022-04-04 11:19:56

本次发布包含了以下内容：

改善已有的文档。

新增接口：

- AddNodes

- BindAutoScalingGroup
- CreateCluster
- DeleteCluster
- DeleteNodes
- DescribeClusters

新增数据结构：

- CFSOption
- ClusterOverview
- ComputeNode
- ComputeNodeOverview
- DataDisk
- GooseFSOption
- InstanceChargePrepaid
- InternetAccessible
- LoginNode
- LoginNodeOverview
- LoginSettings
- ManagerNode
- ManagerNodeOverview
- Placement
- StorageOption
- SystemDisk
- Tag
- VirtualPrivateCloud

简介

最近更新时间：2025-04-25 02:01:32

高性能计算平台（TencentCloud High Performance Computing，THPC）是一款腾讯云自研的高性能计算资源管理服务，集成腾讯云上的计算、存储、网络等产品资源，并整合 HPC 专用作业管理调度、集群管理等软件，向用户提供弹性灵活、性能卓越、自助化的计算服务。可以帮助您高效地管理云上高性能计算资源，实现弹性使用云上高性能计算资源的需求。

API 概览

最近更新时间：2024-08-02 02:14:47

节点相关接口

接口名称	接口功能	频率限制（次/秒）
DeleteNodes	删除节点	20
AddNodes	添加节点	3
DescribeNodes	查询指定集群节点列表	20

存储选项相关接口

接口名称	接口功能	频率限制（次/秒）
AddClusterStorageOption	添加集群存储选项	20
DeleteClusterStorageOption	删除集群存储选项	20
DescribeClusterStorageOption	查询集群存储选项	20

集群相关接口

接口名称	接口功能	频率限制（次/秒）
DeleteCluster	删除集群	20
CreateCluster	创建集群	20
DescribeClusterActivities	查询集群活动历史记录	20
DescribeClusters	查询集群列表	20

弹性伸缩相关接口

接口名称	接口功能	频率限制（次/秒）
DescribeAutoScalingConfiguration	查询弹性伸缩配置信息	20
SetAutoScalingConfiguration	设置弹性伸缩配置信息	3
BindAutoScalingGroup	绑定弹性伸缩组	20

队列相关接口

接口名称	接口功能	频率限制（次/秒）
AddQueue	添加队列	20
DeleteQueue	删除队列	20
DescribeQueues	查询队列列表	20

⚠ 注意：
以上给出的接口频率限制维度为 API + 接入地域 + 子账号，有关限频更多说明参考：API 频率限制说明

调用方式
请求结构

最近更新时间：2025-05-21 01:32:51

1. 服务地址

API 支持就近地域接入，本产品就近地域接入域名为 thpc.tencentcloudapi.com ，也支持指定地域域名访问，例如广州地域的域名为 thpc.ap-guangzhou.tencentcloudapi.com 。

推荐使用就近地域接入域名。根据调用接口时客户端所在位置，会自动解析到**最近的**某个具体地域的服务器。例如在广州发起请求，会自动解析到广州的服务器，效果和指定 thpc.ap-guangzhou.tencentcloudapi.com 是一致的。

注意：对时延敏感的业务，建议指定带地域的域名。

注意：域名是 API 的接入点，并不代表产品或者接口实际提供服务的地域。产品支持的地域列表请在调用方式/公共参数文档中查阅，接口支持的地域请在接口文档输入参数中查阅。

目前支持的域名列表为：

接入地域	域名
就近地域接入（推荐，只支持非金融区）	thpc.tencentcloudapi.com
华南地区（广州）	thpc.ap-guangzhou.tencentcloudapi.com
华东地区（上海）	thpc.ap-shanghai.tencentcloudapi.com
华东地区（南京）	thpc.ap-nanjing.tencentcloudapi.com
华北地区（北京）	thpc.ap-beijing.tencentcloudapi.com
西南地区（成都）	thpc.ap-chengdu.tencentcloudapi.com
西南地区（重庆）	thpc.ap-chongqing.tencentcloudapi.com
港澳台地区（中国香港）	thpc.ap-hongkong.tencentcloudapi.com
亚太东南（新加坡）	thpc.ap-singapore.tencentcloudapi.com
亚太东南（雅加达）	thpc.ap-jakarta.tencentcloudapi.com
亚太东南（曼谷）	thpc.ap-bangkok.tencentcloudapi.com
亚太东北（首尔）	thpc.ap-seoul.tencentcloudapi.com
亚太东北（东京）	thpc.ap-tokyo.tencentcloudapi.com
美国东部（弗吉尼亚）	thpc.na-ashburn.tencentcloudapi.com
美国西部（硅谷）	thpc.na-siliconvalley.tencentcloudapi.com
南美地区（圣保罗）	thpc.sa-saopaulo.tencentcloudapi.com
欧洲地区（法兰克福）	thpc.eu-frankfurt.tencentcloudapi.com

注意：由于金融区和非金融区是隔离不互通的，因此当访问金融区服务时（公共参数 Region 为金融区地域），需要同时指定带金融区地域的域名，最好和 Region 的地域保持一致。

金融区接入地域	金融区域名
华东地区（上海金融）	thpc.ap-shanghai-fsi.tencentcloudapi.com
华南地区（深圳金融）	thpc.ap-shenzhen-fsi.tencentcloudapi.com

2. 通信协议

腾讯云 API 的所有接口均通过 HTTPS 进行通信，提供高安全性的通信通道。

3. 请求方法

支持的 HTTP 请求方法:

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型:

- application/json（推荐），必须使用签名方法 v3（TC3-HMAC-SHA256）。
- application/x-www-form-urlencoded，必须使用签名方法 v1（HmacSHA1 或 HmacSHA256）。
- multipart/form-data（仅部分接口支持），必须使用签名方法 v3（TC3-HMAC-SHA256）。

GET 请求的请求包大小不得超过32KB。POST 请求使用签名方法 v1（HmacSHA1、HmacSHA256）时不得超过1MB。POST 请求使用签名方法 v3（TC3-HMAC-SHA256）时支持10MB。

4. 字符编码

均使用 `UTF-8` 编码。

公共参数

最近更新时间：2024-11-15 02:07:26

公共参数是用于标识用户和接口签名的参数，如非必要，在每个接口单独的文档中不再对这些参数进行说明，但每次请求均需要携带这些参数，才能正常发起请求。

公共参数的具体内容会因您使用的签名方法版本不同而有所差异。

使用签名方法 v3 的公共参数

签名方法 v3（有时也称作 TC3-HMAC-SHA256）相比签名方法 v1（有些文档可能会简称签名方法），更安全，支持更大的请求包，支持 POST JSON 格式，性能有一定提升，推荐使用该签名方法计算签名。完整介绍详见 签名方法 v3。

注意：出于简化的目的，部分接口文档中的示例使用的是签名方法 v1 GET 请求，而不是更安全的签名方法 v3。

使用签名方法 v3 时，公共参数需要统一放到 HTTP Header 请求头部中，如下表所示：

参数名称	类型	必选	描述
Action	String	是	HTTP 请求头：X-TC-Action。操作的接口名称。取值参考接口文档输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	HTTP 请求头：X-TC-Region。地域参数，用来标识希望操作哪个地域的数据。取值参考接口文档中输入参数章节关于公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	HTTP 请求头：X-TC-Timestamp。当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。**注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。**
Version	String	是	HTTP 请求头：X-TC-Version。操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKID***/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKID*** 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为具体产品名，通常为域名前缀。例如，域名 cvm.tencentcloudapi.com 意味着产品名是 cvm。本产品取值为 thpc；tc3_request 为固定字符串； - SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部； - Signature：签名摘要，计算过程详见 文档。
Token	String	否	HTTP 请求头：X-TC-Token。即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	HTTP 请求头：X-TC-Language。指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

假设用户想要查询广州地域的云服务器实例列表中的前十个，接口参数设置为偏移量 Offset=0，返回数量 Limit=10，则其请求结构按照请求 URL、请求头部、请求体示例如下：

HTTP GET 请求结构示例：

```
https://cvm.tencentcloudapi.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: ap-guangzhou
```

HTTP POST（application/json）请求结构示例：

```
https://cvm.tencentcloudapi.com/

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: application/json
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

{"Offset":0,"Limit":10}
```

HTTP POST（multipart/form-data）请求结构示例（仅特定的接口支持）：

```
https://cvm.tencentcloudapi.com/

Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2018-05-30/cvm/tc3_request, SignedHeaders=content-type;host, Signature=582c400e06b5924a6f2b5d7d672d79c15b13162d9279b0855cfba6789a8edb4c
Content-Type: multipart/form-data; boundary=58731222010402
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1527672334
X-TC-Region: ap-guangzhou

--58731222010402
Content-Disposition: form-data; name="Offset"

0
--58731222010402
Content-Disposition: form-data; name="Limit"

10
--58731222010402--
```

使用签名方法 v1 的公共参数

使用签名方法 v1（有时会称作 HmacSHA256 和 HmacSHA1），公共参数需要统一放到请求串中，完整介绍详见文档

参数名称	类型	必选	描述
Action	String	是	操作的接口名称。取值参考接口文档中输入参数章节关于公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	-	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。**注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。**
Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。
Nonce	Integer	是	随机正整数，与 Timestamp 联合起来，用于防止重放攻击。
SecretId	String	是	在 云API密钥 上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey ，而 SecretKey 会用来生成请求签名 Signature。
Signature	String	是	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见 文档。

参数名称	类型	必选	描述
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	即 安全凭证服务 所颁发的临时安全凭证中的 Token，使用时需要将 SecretId 和 SecretKey 的值替换为临时安全凭证中的 TmpSecretId 和 TmpSecretKey。使用长期密钥时不能设置此 Token 字段。
Language	String	否	指定接口返回的语言，仅部分接口支持此参数。取值：zh-CN，en-US。zh-CN 返回中文，en-US 返回英文。

假设用户想要查询广州地域的云服务器实例列表，其请求结构按照请求 URL、请求头部、请求体示例如下：

HTTP GET 请求结构示例：

```
https://cvm.tencentcloudapi.com/?Action=DescribeInstances&Version=2017-03-12&SignatureMethod=HmacSHA256&Timestamp=1527672334&Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*******************************

Host: cvm.tencentcloudapi.com
```

HTTP POST 请求结构示例：

```
https://cvm.tencentcloudapi.com/

Host: cvm.tencentcloudapi.com
Content-Type: application/x-www-form-urlencoded

Action=DescribeInstances&Version=2017-03-12&SignatureMethod=HmacSHA256&Timestamp=1527672334&Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02&Region=ap-guangzhou&Nonce=23823223&SecretId=AKID*****************************
```

地域列表

本产品所有接口 Region 字段的可选值如下表所示。如果接口不支持该表中的所有地域，则会在接口文档中单独说明。

地域	取值
华北地区（北京）	ap-beijing
西南地区（成都）	ap-chengdu
西南地区（重庆）	ap-chongqing
华南地区（广州）	ap-guangzhou
华东地区（南京）	ap-nanjing
华东地区（上海）	ap-shanghai

签名方法 v3

最近更新时间：2024-12-25 02:03:42

以下文档说明了签名方法 v3 的签名过程，但仅在您编写自己的代码来调用腾讯云 API 时才有用。我们推荐您使用 腾讯云 API Explorer ，腾讯云 SDK 和 腾讯云命令行工具（TCCLI）等开发者工具，从而无需学习如何对 API 请求进行签名。

推荐使用 API Explorer　　点击调试

您可以通过 API Explorer 的【签名串生成】模块查看每个接口签名的生成过程。

腾讯云 API 会对每个请求进行身份验证，用户需要使用安全凭证，经过特定的步骤对请求进行签名（Signature），每个请求都需要在公共参数中指定该签名结果并以指定的方式和格式发送请求。

为什么要进行签名

签名通过以下方式帮助保护请求：

1. 验证请求者的身份
   签名确保请求是由持有有效访问密钥的人发送的。请参阅控制台 云 API 密钥 页面获取密钥相关信息。

2. 保护传输中的数据
   为了防止请求在传输过程中被篡改，腾讯云 API 会使用请求参数来计算请求的哈希值，并将生成的哈希值加密后作为请求的一部分，发送到腾讯云 API 服务器。服务器会使用收到的请求参数以同样的过程计算哈希值，并验证请求中的哈希值。如果请求被篡改，将导致哈希值不一致，腾讯云 API 将拒绝本次请求。

签名方法 v3（TC3-HMAC-SHA256）功能上覆盖了以前的签名方法 v1，而且更安全，支持更大的请求，支持 JSON 格式，POST 请求支持传空数组和空字符串，性能有一定提升，推荐使用该签名方法计算签名。

首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v3”，可以对生成签名过程进行验证，也可直接生成 SDK 代码。推荐使用腾讯云 API 配套的 8 种常见的编程语言 SDK，已经封装了签名和请求过程，均已开源，支持 Python、Java、PHP、Go、NodeJS、.NET、C++、Ruby。

申请安全凭证

本文使用的安全凭证为密钥，密钥包括 SecretId 和 SecretKey。每个用户最多可以拥有两对密钥。

- SecretId：用于标识 API 调用者身份，可以简单类比为用户名。
- SecretKey：用于验证 API 调用者的身份，可以简单类比为密码。
- **用户必须严格保管安全凭证，避免泄露，否则将危及财产安全。如已泄露，请立刻禁用该安全凭证。**

申请安全凭证的具体步骤如下：

1. 登录 腾讯云管理中心控制台 。
2. 前往 云API密钥 的控制台页面。
3. 在 云API密钥 页面，单击【新建密钥】创建一对密钥。

签名版本 v3 签名过程

云 API 支持 GET 和 POST 请求。对于GET方法，只支持 `Content-Type: application/x-www-form-urlencoded` 协议格式。对于POST方法，目前支持 `Content-Type: application/json` 以及 `Content-Type: multipart/form-data` 两种协议格式，json 格式绝大多数接口均支持，multipart 格式只有特定接口支持，此时该接口不能使用 json 格式调用，参考具体业务接口文档说明。推荐使用 POST 请求，因为两者的结果并无差异，但 GET 请求只支持 32 KB 以内的请求包。

下面以云服务器查询广州区实例列表作为例子，分步骤介绍签名的计算过程。我们选择该接口是因为：

1. 云服务器默认已开通，该接口很常用；
2. 该接口是只读的，不会改变现有资源的状态；
3. 接口覆盖的参数种类较全，可以演示包含数据结构的数组如何使用。

在示例中，不论公共参数或者接口的参数，我们尽量选择容易犯错的情况。在实际调用接口时，请根据实际情况来，每个接口的参数并不相同，不要照抄这个例子的参数和值。此外，这里只展示了部分公共参数和接口输入参数，用户可以根据实际需要添加其他参数，例如 Language 和 Token 公共参数（在 HTTP 头部设置，添加 X-TC- 前缀）。

假设用户的 SecretId 和 SecretKey 分别是：`AKID********************************` 和 `********************************`。用户想查看广州区云服务器名为“未命名”的主机状态，只返回一条数据。则请求可能为：

```
curl -X POST https://cvm.tencentcloudapi.com \
-H "Authorization: TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f" \
-H "Content-Type: application/json; charset=utf-8" \
-H "Host: cvm.tencentcloudapi.com" \
-H "X-TC-Action: DescribeInstances" \
-H "X-TC-Timestamp: 1551113065" \
-H "X-TC-Version: 2017-03-12" \
-H "X-TC-Region: ap-guangzhou" \
-d '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
```

下面详细解释签名计算过程。

1. 拼接规范请求串

按如下伪代码格式拼接规范请求串（CanonicalRequest）：

```
CanonicalRequest =
HTTPRequestMethod + '\n' +
CanonicalURI + '\n' +
CanonicalQueryString + '\n' +
CanonicalHeaders + '\n' +
SignedHeaders + '\n' +
HashedRequestPayload
```

字段名称	解释
HTTPRequestMethod	HTTP 请求方法（GET、POST）。此示例取值为 `POST`。
CanonicalURI	URI 参数，API 3.0 固定为正斜杠（/）。
CanonicalQueryString	发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串""，对于 GET 请求，则为 URL 中问号（?）后面的字符串内容，例如：Limit=10&Offset=0。 注意：CanonicalQueryString 需要参考 RFC3986 进行 URLEncode 编码（特殊字符编码后需大写字母），字符集 UTF-8。推荐使用编程语言标准库进行编码。
CanonicalHeaders	参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入其他头部参与签名以提高自身请求的唯一性和安全性，此示例额外增加了接口名头部。 拼接规则： 1. 头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接； 2. 多个头部，按照头部 key（小写）的 ASCII 升序进行拼接。 此示例计算结果是 `content-type:application/json; charset=utf-8\nhost:cvm.tencentcloudapi.com\nx-tc-action:describeinstances\n`。 注意：content-type 必须和实际发送的相符合，有些编程语言网络库即使未指定也会自动添加 charset 值，如果签名时和发送时不一致，服务器会返回签名校验失败。
SignedHeaders	参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。 拼接规则： 1. 头部 key 统一转成小写； 2. 多个头部 key（小写）按照 ASCII 升序进行拼接，并且以分号（;）分隔。 此示例为 content-type;host;x-tc-action
HashedRequestPayload	请求正文（payload，即 body，此示例为 `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`）的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，即对 HTTP 请求正文做 SHA256 哈希，然后十六进制编

字段名称	解释
	码，最后编码串转换成小写字母。对于 GET 请求，RequestPayload 固定为空字符串。此示例计算结果是 35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064。

根据以上规则，示例中得到的规范请求串如下：

```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com
x-tc-action:describeinstances

content-type;host;x-tc-action
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. 拼接待签名字符串

按如下格式拼接待签名字符串：

```
StringToSign =
Algorithm + "\n" +
RequestTimestamp + "\n" +
CredentialScope + "\n" +
HashedCanonicalRequest
```

字段名称	解释
Algorithm	签名算法，目前固定为 TC3-HMAC-SHA256。
RequestTimestamp	请求时间戳，即请求头部的公共参数 X-TC-Timestamp 取值，取当前时间 UNIX 时间戳，精确到秒。此示例取值为 1551113065。
CredentialScope	凭证范围，格式为 Date/service/tc3_request，包含日期、所请求的服务和终止字符串（tc3_request）。**Date 为 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致**；service 为产品名，必须与调用的产品域名一致。此示例计算结果是 2019-02-25/cvm/tc3_request。
HashedCanonicalRequest	前述步骤拼接所得规范请求串的哈希值，计算伪代码为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。此示例计算结果是 7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84。

注意：

1. **Date 必须从时间戳 X-TC-Timestamp 计算得到，且时区为 UTC+0。如果加入系统本地时区信息，例如东八区，将导致白天和晚上调用成功，但是凌晨时调用必定失败。假设时间戳为 1551113065，在东八区的时间是 2019-02-26 00:44:25，但是计算得到的 Date 取 UTC+0 的日期应为 2019-02-25，而不是 2019-02-26。**
2. **Timestamp 必须是当前系统时间，且需确保系统时间和标准时间是同步的，如果相差超过五分钟则必定失败。如果长时间不和标准时间同步，可能运行一段时间后，请求失败，返回签名过期错误。**

根据以上规则，示例中得到的待签名字符串如下：

```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
7019a55be8395899b900fb5564e4200d984910f34794a27cb3fb7d10ff6a1e84
```

3. 计算签名

1）计算派生签名密钥，伪代码如下：

```
SecretKey = "********************************"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

派生出的密钥 `SecretDate`、`SecretService` 和 `SecretSigning` 是二进制的数据，可能包含不可打印字符，将其转为十六进制字符串打印的输出分别为：
da98fb70dcf6b112dc21038d1eeeb3a95c74b4dcb12c1131f864f6066bd02be0，
8d70cbefb03939f929db64d32dc2ba89b1095620119fe3e050e2b18c5bd2752f，
b596b923aad85185e2d1f6659d2a062e0a86731226e021e61bfe06f7ed05f5af。

请注意，不同的编程语言，HMAC 库函数中参数顺序可能不一样，请以实际情况为准。此处的伪代码密钥参数 key 在前，消息参数 data 在后。通常标准库函数会提供二进制格式的返回值，也可能会提供打印友好的十六进制格式的返回值，此处使用的是二进制格式。

字段名称	解释
SecretKey	原始的 SecretKey，即 `********************************`。
Date	即 Credential 中的 Date 字段信息。此示例取值为 `2019-02-25`。
Service	即 Credential 中的 Service 字段信息。此示例取值为 `cvm`。

2）计算签名，伪代码如下：

```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

此示例计算结果是 `10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f`。

4. 拼接 Authorization

按如下格式拼接 Authorization：

```
Authorization =
Algorithm + ' ' +
'Credential=' + SecretId + '/' + CredentialScope + ', ' +
'SignedHeaders=' + SignedHeaders + ', ' +
'Signature=' + Signature
```

字段名称	解释
Algorithm	签名方法，固定为 `TC3-HMAC-SHA256`。
SecretId	密钥对中的 SecretId，即 `AKID********************************`。
CredentialScope	见上文，凭证范围。此示例计算结果是 `2019-02-25/cvm/tc3_request`。
SignedHeaders	见上文，参与签名的头部信息。此示例取值为 `content-type;host;x-tc-action`。
Signature	签名值。此示例计算结果是 `10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f`。

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKID********************************/2019-02-25/cvm/tc3_request, SignedHeaders=content-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
```

最终完整的调用信息如下：

```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKID*******************************/2019-02-25/cvm/tc3_request, SignedHeaders=c
ontent-type;host;x-tc-action, Signature=10b1a37a7301a02ca19a647ad722d5e43b4b3cff309d421d85b46093f6ab6c4f
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}
```

⚠ 注意：

请求发送时的 HTTP 头部（Header）和请求体（Payload）必须和签名计算过程中的内容完全一致，否则会返回签名不一致错误。可以通过打印实际请求内容，网络抓包等方式对比排查。

签名演示

在实际调用 API 3.0 时，推荐使用配套的腾讯云 SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

下面提供了不同产品的生成签名 demo，您可以找到对应的产品参考签名的生成：

- Signature Demo

为了更清楚地解释签名过程，下面以实际编程语言为例，将上述的签名过程完整实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK。

Java

```java
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TencentCloudAPITC3Demo {
private final static Charset UTF8 = StandardCharsets.UTF_8;
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
private final static String SECRET_ID = System.getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
```

```
private final static String SECRET_KEY = System.getenv("TENCENTCLOUD_SECRET_KEY");
private final static String CT_JSON = "application/json; charset=utf-8";

public static byte[] hmac256(byte[] key, String msg) throws Exception {
Mac mac = Mac.getInstance("HmacSHA256");
SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
mac.init(secretKeySpec);
return mac.doFinal(msg.getBytes(UTF8));
}

public static String sha256Hex(String s) throws Exception {
MessageDigest md = MessageDigest.getInstance("SHA-256");
byte[] d = md.digest(s.getBytes(UTF8));
return DatatypeConverter.printHexBinary(d).toLowerCase();
}

public static void main(String[] args) throws Exception {
String service = "cvm";
String host = "cvm.tencentcloudapi.com";
String region = "ap-guangzhou";
String action = "DescribeInstances";
String version = "2017-03-12";
String algorithm = "TC3-HMAC-SHA256";
String timestamp = "1551113065";
//String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
// 注意时区，否则容易出错
sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

// ************ 步骤 1: 拼接规范请求串 ************
String httpRequestMethod = "POST";
String canonicalUri = "/";
String canonicalQueryString = "";
String canonicalHeaders = "content-type:application/json; charset=utf-8\n"
+ "host:" + host + "\n" + "x-tc-action:" + action.toLowerCase() + "\n";
String signedHeaders = "content-type;host;x-tc-action";

String payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}";
String hashedRequestPayload = sha256Hex(payload);
String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryString + "\n"
+ canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
System.out.println(canonicalRequest);

// ************ 步骤 2: 拼接待签名字符串 ************
String credentialScope = date + "/" + service + "/" + "tc3_request";
String hashedCanonicalRequest = sha256Hex(canonicalRequest);
String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
System.out.println(stringToSign);

// ************ 步骤 3: 计算签名 ************
byte[] secretDate = hmac256(("TC3" + SECRET_KEY).getBytes(UTF8), date);
byte[] secretService = hmac256(secretDate, service);
byte[] secretSigning = hmac256(secretService, "tc3_request");
String signature = DatatypeConverter.printHexBinary(hmac256(secretSigning, stringToSign)).toLowerCase();
System.out.println(signature);
```

```
// ************* 步骤 4：拼接 Authorization *************
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
System.out.println(authorization);

TreeMap<String, String> headers = new TreeMap<String, String>();
headers.put("Authorization", authorization);
headers.put("Content-Type", CT_JSON);
headers.put("Host", host);
headers.put("X-TC-Action", action);
headers.put("X-TC-Timestamp", timestamp);
headers.put("X-TC-Version", version);
headers.put("X-TC-Region", region);

StringBuilder sb = new StringBuilder();
sb.append("curl -X POST https://").append(host)
.append(" -H \"Authorization: ").append(authorization).append("\"")
.append(" -H \"Content-Type: application/json; charset=utf-8\"")
.append(" -H \"Host: ").append(host).append("\"")
.append(" -H \"X-TC-Action: ").append(action).append("\"")
.append(" -H \"X-TC-Timestamp: ").append(timestamp).append("\"")
.append(" -H \"X-TC-Version: ").append(version).append("\"")
.append(" -H \"X-TC-Region: ").append(region).append("\"")
.append(" -d '").append(payload).append("'");
System.out.println(sb.toString());
}
}
```

Python

```
# -*- coding: utf-8 -*-
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# 密钥参数
# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secret_key = os.environ.get("TENCENTCLOUD_SECRET_KEY")

service = "cvm"
host = "cvm.tencentcloudapi.com"
endpoint = "https://" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
#timestamp = int(time.time())
timestamp = 1551113065
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 1, "Filters": [{"Values": [u"未命名"], "Name": "instance-name"}]}

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = "POST"
canonical_uri = "/"
canonical_querystring = ""
```

```
ct = "application/json; charset=utf-8"
payload = json.dumps(params)
canonical_headers = "content-type:%s\nhost:%s\nx-tc-action:%s\n" % (ct, host, action.lower())
signed_headers = "content-type;host;x-tc-action"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
canonical_uri + "\n" +
canonical_querystring + "\n" +
canonical_headers + "\n" +
signed_headers + "\n" +
hashed_request_payload)
print(canonical_request)

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
str(timestamp) + "\n" +
credential_scope + "\n" +
hashed_canonical_request)
print(string_to_sign)


# ************* 步骤 3：计算签名 *************
# 计算签名摘要函数
def sign(key, msg):
return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
print(signature)

# ************* 步骤 4：拼接 Authorization *************
authorization = (algorithm + " " +
"Credential=" + secret_id + "/" + credential_scope + ", " +
"SignedHeaders=" + signed_headers + ", " +
"Signature=" + signature)
print(authorization)

print('curl -X POST ' + endpoint
+ ' -H "Authorization: ' + authorization + '"'
+ ' -H "Content-Type: application/json; charset=utf-8"'
+ ' -H "Host: ' + host + '"'
+ ' -H "X-TC-Action: ' + action + '"'
+ ' -H "X-TC-Timestamp: ' + str(timestamp) + '"'
+ ' -H "X-TC-Version: ' + version + '"'
+ ' -H "X-TC-Region: ' + region + '"'
+ " -d '" + payload + "'")
```

Golang

```
package main

import (
"crypto/hmac"
```

```
"crypto/sha256"
"encoding/hex"
"fmt"
"os"
"strings"
"time"
)

func sha256hex(s string) string {
b := sha256.Sum256([]byte(s))
return hex.EncodeToString(b[:])
}

func hmacsha256(s, key string) string {
hashed := hmac.New(sha256.New, []byte(key))
hashed.Write([]byte(s))
return string(hashed.Sum(nil))
}

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
host := "cvm.tencentcloudapi.com"
algorithm := "TC3-HMAC-SHA256"
service := "cvm"
version := "2017-03-12"
action := "DescribeInstances"
region := "ap-guangzhou"
//var timestamp int64 = time.Now().Unix()
var timestamp int64 = 1551113065

// step 1: build canonical request string
httpRequestMethod := "POST"
canonicalURI := "/"
canonicalQueryString := ""
canonicalHeaders := fmt.Sprintf("content-type:%s\nhost:%s\nx-tc-action:%s\n",
"application/json; charset=utf-8", host, strings.ToLower(action))
signedHeaders := "content-type;host;x-tc-action"
payload := `{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}`
hashedRequestPayload := sha256hex(payload)
canonicalRequest := fmt.Sprintf("%s\n%s\n%s\n%s\n%s\n%s",
httpRequestMethod,
canonicalURI,
canonicalQueryString,
canonicalHeaders,
signedHeaders,
hashedRequestPayload)
fmt.Println(canonicalRequest)

// step 2: build string to sign
date := time.Unix(timestamp, 0).UTC().Format("2006-01-02")
credentialScope := fmt.Sprintf("%s/%s/tc3_request", date, service)
hashedCanonicalRequest := sha256hex(canonicalRequest)
string2sign := fmt.Sprintf("%s\n%d\n%s\n%s",
algorithm,
```

```
timestamp,
credentialScope,
hashedCanonicalRequest)
fmt.Println(string2sign)

// step 3: sign string
secretDate := hmacsha256(date, "TC3"+secretKey)
secretService := hmacsha256(service, secretDate)
secretSigning := hmacsha256("tc3_request", secretService)
signature := hex.EncodeToString([]byte(hmacsha256(string2sign, secretSigning)))
fmt.Println(signature)

// step 4: build authorization
authorization := fmt.Sprintf("%s Credential=%s/%s, SignedHeaders=%s, Signature=%s",
algorithm,
secretId,
credentialScope,
signedHeaders,
signature)
fmt.Println(authorization)

curl := fmt.Sprintf(`curl -X POST https://%s\
-H "Authorization: %s"\
-H "Content-Type: application/json; charset=utf-8"\
-H "Host: %s" -H "X-TC-Action: %s"\
-H "X-TC-Timestamp: %d"\
-H "X-TC-Version: %s"\
-H "X-TC-Region: %s"\
-d '%s'`, host, authorization, host, action, timestamp, version, region, payload)
fmt.Println(curl)
}
```

PHP

```
<?php
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$host = "cvm.tencentcloudapi.com";
$service = "cvm";
$version = "2017-03-12";
$action = "DescribeInstances";
$region = "ap-guangzhou";
// $timestamp = time();
$timestamp = 1551113065;
$algorithm = "TC3-HMAC-SHA256";

// step 1: build canonical request string
$httpRequestMethod = "POST";
$canonicalUri = "/";
$canonicalQueryString = "";
$canonicalHeaders = implode("\n", [
"content-type:application/json; charset=utf-8",
"host:".$host,
"x-tc-action:".strtolower($action),
```

```
""
]);
$signedHeaders = implode(";", [
"content-type",
"host",
"x-tc-action",
]);
$payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}';
$hashedRequestPayload = hash("SHA256", $payload);
$canonicalRequest = $httpRequestMethod."\n"
.$canonicalUri."\n"
.$canonicalQueryString."\n"
.$canonicalHeaders."\n"
.$signedHeaders."\n"
.$hashedRequestPayload;
echo $canonicalRequest.PHP_EOL;

// step 2: build string to sign
$date = gmdate("Y-m-d", $timestamp);
$credentialScope = $date."/".$service."/tc3_request";
$hashedCanonicalRequest = hash("SHA256", $canonicalRequest);
$stringToSign = $algorithm."\n"
.$timestamp."\n"
.$credentialScope."\n"
.$hashedCanonicalRequest;
echo $stringToSign.PHP_EOL;

// step 3: sign string
$secretDate = hash_hmac("SHA256", $date, "TC3".$secretKey, true);
$secretService = hash_hmac("SHA256", $service, $secretDate, true);
$secretSigning = hash_hmac("SHA256", "tc3_request", $secretService, true);
$signature = hash_hmac("SHA256", $stringToSign, $secretSigning);
echo $signature.PHP_EOL;

// step 4: build authorization
$authorization = $algorithm
." Credential=".$secretId."/".$credentialScope
.", SignedHeaders=".$signedHeaders.", Signature=".$signature;
echo $authorization.PHP_EOL;

$curl = "curl -X POST https://".$host
.' -H "Authorization: '.$authorization.'"'
.' -H "Content-Type: application/json; charset=utf-8"'
.' -H "Host: '.$host.'"'
.' -H "X-TC-Action: '.$action.'"'
.' -H "X-TC-Timestamp: '.$timestamp.'"'
.' -H "X-TC-Version: '.$version.'"'
.' -H "X-TC-Region: '.$region.'"'
." -d '".$payload."'";
echo $curl.PHP_EOL;
```

Ruby

```
# -*- coding: UTF-8 -*-
# require ruby>=2.3.0
require 'digest'
```

```
require 'json'
require 'time'
require 'openssl'

# 密钥参数
# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

service = 'cvm'
host = 'cvm.tencentcloudapi.com'
endpoint = 'https://' + host
region = 'ap-guangzhou'
action = 'DescribeInstances'
version = '2017-03-12'
algorithm = 'TC3-HMAC-SHA256'
# timestamp = Time.now.to_i
timestamp = 1551113065
date = Time.at(timestamp).utc.strftime('%Y-%m-%d')

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = 'POST'
canonical_uri = '/'
canonical_querystring = ''
canonical_headers = "content-type:application/json; charset=utf-8\nhost:#{host}\nx-tc-action:#{action.downcase}\n"
signed_headers = 'content-type;host;x-tc-action'
# params = { 'Limit' => 1, 'Filters' => [{ 'Name' => 'instance-name', 'Values' => ['未命名'] }] }
# payload = JSON.generate(params, { 'ascii_only' => true, 'space' => ' ' })
# json will generate in random order, to get specified result in example, we hard-code it here.
payload = '{"Limit": 1, "Filters": [{"Values": ["\u672a\u547d\u540d"], "Name": "instance-name"}]}'
hashed_request_payload = Digest::SHA256.hexdigest(payload)
canonical_request = [
http_request_method,
canonical_uri,
canonical_querystring,
canonical_headers,
signed_headers,
hashed_request_payload,
].join("\n")

puts canonical_request

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + '/' + service + '/' + 'tc3_request'
hashed_request_payload = Digest::SHA256.hexdigest(canonical_request)
string_to_sign = [
algorithm,
timestamp.to_s,
credential_scope,
hashed_request_payload,
].join("\n")
puts string_to_sign

# ************* 步骤 3：计算签名 *************
digest = OpenSSL::Digest.new('sha256')
secret_date = OpenSSL::HMAC.digest(digest, 'TC3' + secret_key, date)
```

```
secret_service = OpenSSL::HMAC.digest(digest, secret_date, service)
secret_signing = OpenSSL::HMAC.digest(digest, secret_service, 'tc3_request')
signature = OpenSSL::HMAC.hexdigest(digest, secret_signing, string_to_sign)
puts signature

# ************* 步骤 4：拼接 Authorization *************
authorization = "#{algorithm} Credential=#{secret_id}/#{credential_scope}, SignedHeaders=#{signed_headers}, Signature=#{si
gnature}"
puts authorization

puts 'curl -X POST ' + endpoint \
+ ' -H "Authorization: ' + authorization + '"' \
+ ' -H "Content-Type: application/json; charset=utf-8"' \
+ ' -H "Host: ' + host + '"' \
+ ' -H "X-TC-Action: ' + action + '"' \
+ ' -H "X-TC-Timestamp: ' + timestamp.to_s + '"' \
+ ' -H "X-TC-Version: ' + version + '"' \
+ ' -H "X-TC-Region: ' + region + '"' \
+ " -d '" + payload + "'"
```

DotNet

```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

public class Application
{
public static string SHA256Hex(string s)
{
using (SHA256 algo = SHA256.Create())
{
byte[] hashbytes = algo.ComputeHash(Encoding.UTF8.GetBytes(s));
StringBuilder builder = new StringBuilder();
for (int i = 0; i < hashbytes.Length; ++i)
{
builder.Append(hashbytes[i].ToString("x2"));
}
return builder.ToString();
}
}

public static byte[] HmacSHA256(byte[] key, byte[] msg)
{
using (HMACSHA256 mac = new HMACSHA256(key))
{
return mac.ComputeHash(msg);
}
}

public static Dictionary<String, String> BuildHeaders(string secretid,
string secretkey, string service, string endpoint, string region,
string action, string version, DateTime date, string requestPayload)
{
string datestr = date.ToString("yyyy-MM-dd");
```

```
DateTime startTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);
long requestTimestamp = (long)Math.Round((date - startTime).TotalMilliseconds, MidpointRounding.AwayFromZero) / 1000;
// ************* 步骤 1: 拼接规范请求串 *************
string algorithm = "TC3-HMAC-SHA256";
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string contentType = "application/json";
string canonicalHeaders = "content-type:" + contentType + "; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.ToLower() + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string hashedRequestPayload = SHA256Hex(requestPayload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
Console.WriteLine(canonicalRequest);

// ************* 步骤 2: 拼接待签名字符串 *************
string credentialScope = datestr + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = SHA256Hex(canonicalRequest);
string stringToSign = algorithm + "\n"
+ requestTimestamp.ToString() + "\n"
+ credentialScope + "\n"
+ hashedCanonicalRequest;
Console.WriteLine(stringToSign);

// ************* 步骤 3: 计算签名 *************
byte[] tc3SecretKey = Encoding.UTF8.GetBytes("TC3" + secretkey);
byte[] secretDate = HmacSHA256(tc3SecretKey, Encoding.UTF8.GetBytes(datestr));
byte[] secretService = HmacSHA256(secretDate, Encoding.UTF8.GetBytes(service));
byte[] secretSigning = HmacSHA256(secretService, Encoding.UTF8.GetBytes("tc3_request"));
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringToSign));
string signature = BitConverter.ToString(signatureBytes).Replace("-", "").ToLower();
Console.WriteLine(signature);

// ************* 步骤 4: 拼接 Authorization *************
string authorization = algorithm + " "
+ "Credential=" + secretid + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", "
+ "Signature=" + signature;
Console.WriteLine(authorization);

Dictionary<string, string> headers = new Dictionary<string, string>();
headers.Add("Authorization", authorization);
headers.Add("Host", endpoint);
headers.Add("Content-Type", contentType + "; charset=utf-8");
headers.Add("X-TC-Timestamp", requestTimestamp.ToString());
headers.Add("X-TC-Version", version);
headers.Add("X-TC-Action", action);
headers.Add("X-TC-Region", region);
return headers;
}
public static void Main(string[] args)
```

```
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

string service = "cvm";
string endpoint = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";

// 此处由于示例规范的原因，采用时间戳2019-02-26 00:44:25，此参数作为示例，如果在项目中，您应当使用：
// DateTime date = DateTime.UtcNow;
// 注意时区，建议此时间统一采用UTC时间戳，否则容易出错
DateTime date = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc).AddSeconds(1551113065);
string requestPayload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name
\"}]}";

Dictionary<string, string> headers = BuildHeaders(SECRET_ID, SECRET_KEY, service
, endpoint, region, action, version, date, requestPayload);

Console.WriteLine("POST https://cvm.tencentcloudapi.com");
foreach (KeyValuePair<string, string> kv in headers)
{
Console.WriteLine(kv.Key + ": " + kv.Value);
}
Console.WriteLine();
Console.WriteLine(requestPayload);
}
}
```

NodeJS

```
const crypto = require('crypto');

function sha256(message, secret = '', encoding) {
const hmac = crypto.createHmac('sha256', secret)
return hmac.update(message).digest(encoding)
}

function getHash(message, encoding = 'hex') {
const hash = crypto.createHash('sha256')
return hash.update(message).digest(encoding)
}

function getDate(timestamp) {
const date = new Date(timestamp * 1000)
const year = date.getUTCFullYear()
const month = ('0' + (date.getUTCMonth() + 1)).slice(-2)
const day = ('0' + date.getUTCDate()).slice(-2)
return `${year}-${month}-${day}`
}

function main(){
```

```
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

const endpoint = "cvm.tencentcloudapi.com"
const service = "cvm"
const region = "ap-guangzhou"
const action = "DescribeInstances"
const version = "2017-03-12"
//const timestamp = getTime()
const timestamp = 1551113065
//时间处理，获取世界时间日期
const date = getDate(timestamp)

// ************* 步骤 1：拼接规范请求串 *************
const payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}"

const hashedRequestPayload = getHash(payload);
const httpRequestMethod = "POST"
const canonicalUri = "/"
const canonicalQueryString = ""
const canonicalHeaders = "content-type:application/json; charset=utf-8\n"
+ "host:" + endpoint + "\n"
+ "x-tc-action:" + action.toLowerCase() + "\n"
const signedHeaders = "content-type;host;x-tc-action"

const canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload
console.log(canonicalRequest)

// ************* 步骤 2：拼接待签名字符串 *************
const algorithm = "TC3-HMAC-SHA256"
const hashedCanonicalRequest = getHash(canonicalRequest);
const credentialScope = date + "/" + service + "/" + "tc3_request"
const stringToSign = algorithm + "\n" +
timestamp + "\n" +
credentialScope + "\n" +
hashedCanonicalRequest
console.log(stringToSign)

// ************* 步骤 3：计算签名 *************
const kDate = sha256(date, 'TC3' + SECRET_KEY)
const kService = sha256(service, kDate)
const kSigning = sha256('tc3_request', kService)
const signature = sha256(stringToSign, kSigning, 'hex')
console.log(signature)

// ************* 步骤 4：拼接 Authorization *************
const authorization = algorithm + " " +
"Credential=" + SECRET_ID + "/" + credentialScope + ", " +
"SignedHeaders=" + signedHeaders + ", " +
```

```
"Signature=" + signature
console.log(authorization)

const curlcmd = 'curl -X POST ' + "https://" + endpoint
+ ' -H "Authorization: ' + authorization + '"'
+ ' -H "Content-Type: application/json; charset=utf-8"'
+ ' -H "Host: ' + endpoint + '"'
+ ' -H "X-TC-Action: ' + action + '"'
+ ' -H "X-TC-Timestamp: ' + timestamp.toString() + '"'
+ ' -H "X-TC-Version: ' + version + '"'
+ ' -H "X-TC-Region: ' + region + '"'
+ " -d '" + payload + "'"
console.log(curlcmd)
}
main()
```

C++

```
#include <algorithm>
#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdio.h>
#include <time.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>

using namespace std;

string get_data(int64_t &timestamp)
{
string utcDate;
char buff[20] = {0};
// time_t timenow;
struct tm sttime;
sttime = *gmtime(&timestamp);
strftime(buff, sizeof(buff), "%Y-%m-%d", &sttime);
utcDate = string(buff);
return utcDate;
}

string int2str(int64_t n)
{
std::stringstream ss;
ss << n;
return ss.str();
}

string sha256Hex(const string &str)
{
char buf[3];
unsigned char hash[SHA256_DIGEST_LENGTH];
SHA256_CTX sha256;
SHA256_Init(&sha256);
```

```
SHA256_Update(&sha256, str.c_str(), str.size());
SHA256_Final(hash, &sha256);
std::string NewString = "";
for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
{
snprintf(buf, sizeof(buf), "%02x", hash[i]);
NewString = NewString + buf;
}
return NewString;
}

string HmacSha256(const string &key, const string &input)
{
unsigned char hash[32];

HMAC_CTX *h;
#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif

HMAC_Init_ex(h, &key[0], key.length(), EVP_sha256(), NULL);
HMAC_Update(h, ( unsigned char* )&input[0], input.length());
unsigned int len = 32;
HMAC_Final(h, hash, &len);

#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

std::stringstream ss;
ss << std::setfill('0');
for (int i = 0; i < len; i++)
{
ss << hash[i];
}

return (ss.str());
}

string HexEncode(const string &input)
{
static const char* const lut = "0123456789abcdef";
size_t len = input.length();

string output;
output.reserve(2 * len);
for (size_t i = 0; i < len; ++i)
{
const unsigned char c = input[i];
output.push_back(lut[c >> 4]);
output.push_back(lut[c & 15]);
```

```
}
return output;
}

int main()
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
string SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");

string service = "cvm";
string host = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";
int64_t timestamp = 1551113065;
string date = get_data(timestamp);

// ************* 步骤 1: 拼接规范请求串 *************
string httpRequestMethod = "POST";
string canonicalUri = "/";
string canonicalQueryString = "";
string lower = action;
std::transform(action.begin(), action.end(), lower.begin(), ::tolower);
string canonicalHeaders = string("content-type:application/json; charset=utf-8\n")
+ "host:" + host + "\n"
+ "x-tc-action:" + lower + "\n";
string signedHeaders = "content-type;host;x-tc-action";
string payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name\"}]}";
string hashedRequestPayload = sha256Hex(payload);
string canonicalRequest = httpRequestMethod + "\n"
+ canonicalUri + "\n"
+ canonicalQueryString + "\n"
+ canonicalHeaders + "\n"
+ signedHeaders + "\n"
+ hashedRequestPayload;
cout << canonicalRequest << endl;

// ************* 步骤 2: 拼接待签名字符串 *************
string algorithm = "TC3-HMAC-SHA256";
string RequestTimestamp = int2str(timestamp);
string credentialScope = date + "/" + service + "/" + "tc3_request";
string hashedCanonicalRequest = sha256Hex(canonicalRequest);
string stringToSign = algorithm + "\n" + RequestTimestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
cout << stringToSign << endl;

// ************* 步骤 3: 计算签名 ***************
string kKey = "TC3" + SECRET_KEY;
string kDate = HmacSha256(kKey, date);
string kService = HmacSha256(kDate, service);
string kSigning = HmacSha256(kService, "tc3_request");
string signature = HexEncode(HmacSha256(kSigning, stringToSign));
cout << signature << endl;

// ************* 步骤 4: 拼接 Authorization *************
```

```
string authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
cout << authorization << endl;

string curlcmd = "curl -X POST https://" + host + "\n"
+ " -H \"Authorization: " + authorization + "\"\n"
+ " -H \"Content-Type: application/json; charset=utf-8\"" + "\n"
+ " -H \"Host: " + host + "\"\n"
+ " -H \"X-TC-Action: " + action + "\"\n"
+ " -H \"X-TC-Timestamp: " + RequestTimestamp + "\"\n"
+ " -H \"X-TC-Version: " + version + "\"\n"
+ " -H \"X-TC-Region: " + region + "\"\n"
+ " -d '" + payload + "\'";
cout << curlcmd << endl;
return 0;
};
```

C

```
#include <ctype.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>

void get_utc_date(int64_t timestamp, char* utc, int len)
{
// time_t timenow;
struct tm sttime;
sttime = *gmtime(&timestamp);
strftime(utc, len, "%Y-%m-%d", &sttime);
}

void sha256_hex(const char* str, char* result)
{
char buf[3];
unsigned char hash[SHA256_DIGEST_LENGTH];
SHA256_CTX sha256;
SHA256_Init(&sha256);
SHA256_Update(&sha256, str, strlen(str));
SHA256_Final(hash, &sha256);
for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
{
snprintf(buf, sizeof(buf), "%02x", hash[i]);
strcat(result, buf);
}
}

void hmac_sha256(const char* key, int key_len,
const char* input, int input_len,
unsigned char* output, unsigned int* output_len)
{
HMAC_CTX *h;
```

```
#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX hmac;
HMAC_CTX_init(&hmac);
h = &hmac;
#else
h = HMAC_CTX_new();
#endif

HMAC_Init_ex(h, key, key_len, EVP_sha256(), NULL);
HMAC_Update(h, ( unsigned char* )input, input_len);
HMAC_Final(h, output, output_len);


#if OPENSSL_VERSION_NUMBER < 0x10100000L
HMAC_CTX_cleanup(h);
#else
HMAC_CTX_free(h);
#endif

}

void hex_encode(const char* input, int input_len, char* output)
{
static const char* const lut = "0123456789abcdef";

char add_out[128] = {0};
char temp[2] = {0};
for (size_t i = 0; i < input_len; ++i)
{
const unsigned char c = input[i];
temp[0] = lut[c >> 4];
strcat(add_out, temp);
temp[0] = lut[c & 15];
strcat(add_out, temp);
}
strncpy(output, add_out, 128);
}

void lowercase(const char * src, char * dst)
{
for (int i = 0; src[i]; i++)
{
dst[i] = tolower(src[i]);
}
}

int main()
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
const char* SECRET_ID = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const char* SECRET_KEY = getenv("TENCENTCLOUD_SECRET_KEY");
const char* service = "cvm";
const char* host = "cvm.tencentcloudapi.com";
const char* region = "ap-guangzhou";
const char* action = "DescribeInstances";
```

```
const char* version = "2017-03-12";
int64_t timestamp = 1551113065;
char date[20] = {0};
get_utc_date(timestamp, date, sizeof(date));

// ************* 步骤 1: 拼接规范请求串 *************
const char* http_request_method = "POST";
const char* canonical_uri = "/";
const char* canonical_query_string = "";
char canonical_headers[100] = {"content-type:application/json; charset=utf-8\nhost:"};
strcat(canonical_headers, host);
strcat(canonical_headers, "\nx-tc-action:");
char value[100] = {0};
lowercase(action, value);
strcat(canonical_headers, value);
strcat(canonical_headers, "\n");
const char* signed_headers = "content-type;host;x-tc-action";
const char* payload = "{\"Limit\": 1, \"Filters\": [{\"Values\": [\"\\u672a\\u547d\\u540d\"], \"Name\": \"instance-name
\"}]}";
char hashed_request_payload[100] = {0};
sha256_hex(payload, hashed_request_payload);

char canonical_request[256] = {0};
sprintf(canonical_request, "%s\n%s\n%s\n%s\n%s\n%s", http_request_method,
canonical_uri, canonical_query_string, canonical_headers,
signed_headers, hashed_request_payload);
printf("%s\n", canonical_request);


// ************* 步骤 2: 拼接待签名字符串 *************
const char* algorithm = "TC3-HMAC-SHA256";
char request_timestamp[16] = {0};
sprintf(request_timestamp, "%d", timestamp);
char credential_scope[64] = {0};
strcat(credential_scope, date);
sprintf(credential_scope, "%s/%s/tc3_request", date, service);
char hashed_canonical_request[100] = {0};
sha256_hex(canonical_request, hashed_canonical_request);
char string_to_sign[256] = {0};
sprintf(string_to_sign, "%s\n%s\n%s\n%s", algorithm, request_timestamp,
credential_scope, hashed_canonical_request);
printf("%s\n", string_to_sign);

// ************* 步骤 3: 计算签名 ***************
char k_key[64] = {0};
sprintf(k_key, "%s%s", "TC3", SECRET_KEY);
unsigned char k_date[64] = {0};
unsigned int output_len = 0;
hmac_sha256(k_key, strlen(k_key), date, strlen(date), k_date, &output_len);
unsigned char k_service[64] = {0};
hmac_sha256(k_date, output_len, service, strlen(service), k_service, &output_len);
unsigned char k_signing[64] = {0};
hmac_sha256(k_service, output_len, "tc3_request", strlen("tc3_request"), k_signing, &output_len);
unsigned char k_hmac_sha_sign[64] = {0};
hmac_sha256(k_signing, output_len, string_to_sign, strlen(string_to_sign), k_hmac_sha_sign, &output_len);

char signature[128] = {0};
```

```
hex_encode(k_hmac_sha_sign, output_len, signature);
printf("%s\n", signature);

// ************* 步骤 4: 拼接 Authorization *************
char authorization[512] = {0};
sprintf(authorization, "%s Credential=%s/%s, SignedHeaders=%s, Signature=%s",
algorithm, SECRET_ID, credential_scope, signed_headers, signature);
printf("%s\n", authorization);

char curlcmd[10240] = {0};
sprintf(curlcmd, "curl -X POST https://%s\n \
-H \"Authorization: %s\"\n \
-H \"Content-Type: application/json; charset=utf-8\"\n \
-H \"Host: %s\"\n \
-H \"X-TC-Action: %s\"\n \
-H \"X-TC-Timestamp: %s\"\n \
-H \"X-TC-Version: %s\"\n \
-H \"X-TC-Region: %s\"\n \
-d \'%s\'",
host, authorization, host, action, request_timestamp, version, region, payload);
printf("%s\n", curlcmd);
return 0;
}
```

其他语言

- Lua: GitHub
- Swift: GitHub
- Dart: GitHub
- Shell(Bash): GitHub

签名失败

存在以下签名失败的错误码，请根据实际情况处理。

错误码	错误描述
AuthFailure.SignatureExpire	签名过期。Timestamp 与服务器接收到请求的时间相差不得超过五分钟。
AuthFailure.SecretIdNotFound	密钥不存在。请到控制台查看密钥是否被禁用，是否少复制了字符或者多了字符。
AuthFailure.SignatureFailure	签名错误。可能是签名计算错误，或者签名与实际发送的内容不相符合，也有可能是密钥 SecretKey 错误导致的。
AuthFailure.TokenFailure	临时证书 Token 错误。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。

签名方法

最近更新时间：2024-12-25 02:03:42

签名方法 v1 简单易用，但是功能和安全性都不如签名方法 v3，推荐使用签名方法 v3。

首次接触，建议使用 API Explorer 中的“签名串生成”功能，选择签名版本为“API 3.0 签名 v1”，可以生成签名过程进行验证，并提供了部分编程语言的签名示例，也可直接生成 SDK 代码。推荐使用腾讯云 API 配套的 8 种常见的编程语言 SDK，已经封装了签名和请求过程，均已开源，支持 Python、Java、PHP、Go、NodeJS、.NET、C++、Ruby。

推荐使用 API Explorer　　点击调试

您可以通过 API Explorer 的【签名串生成】模块查看每个接口签名的生成过程。

腾讯云 API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。
签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往 云API密钥页面 申请，否则无法调用云 API 接口。

1. 申请安全凭证

在第一次使用云 API 之前，请前往 云 API 密钥页面 申请安全凭证。
安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- **用户必须严格保管安全凭证，避免泄露。**

申请安全凭证的具体步骤如下：

1. 登录 腾讯云管理中心控制台 。
2. 前往 云 API 密钥 的控制台页面
3. 在 云 API 密钥 页面，单击【新建密钥】即可以创建一对 SecretId/SecretKey。

注意：每个账号最多可以拥有两对 SecretId/SecretKey。

2. 生成签名串

有了安全凭证 SecretId 和 SecretKey 后，就可以生成签名串了。以下是使用签名方法 v1 生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKID********************************
- SecretKey: ********************************

注意：这里只是示例，请根据用户实际申请的 SecretId 和 SecretKey 进行后续操作！

以云服务器查看实例列表（DescribeInstances）请求为例，当用户调用这一接口时，其请求参数可能如下：

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥 ID	AKID********************************
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例 ID	ins-09dx96dg
Offset	偏移量	0
Limit	最大允许输出	20
Version	接口版本号	2017-03-12

这里只展示了部分公共参数和接口输入参数，用户可以根据实际需要添加其他参数，例如 Language 和 Token 公共参数。

2.1. 对参数排序

首先对所有请求参数按参数名的字典序（ ASCII 码）升序排序。注意：1）只按参数名进行排序，参数值保持对应即可，不参与比大小；2）按 ASCII 码比大小，如 InstanceIds.2 要排在 InstanceIds.12 后面，不是按字母表，也不是按数值。用户可以借助编程语言中的相关排序函数来实现这一功能，如 PHP 中的 ksort 函数。上述示例参数的排序结果如下:

```
{
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : 'AKID********************************',
'Timestamp' : 1465185768,
'Version': '2017-03-12',
}
```

使用其它程序设计语言开发时，可对上面示例中的参数进行排序，得到的结果一致即可。

2.2. 拼接请求字符串

此步骤生成请求字符串。
将把上一步排序好的请求参数格式化成“参数名称=参数值”的形式，如对 Action 参数，其参数名称为 "Action" ，参数值为 "DescribeInstances" ，因此格式化后就为 Action=DescribeInstances 。
注意：“参数值”为原始值而非 url 编码后的值。

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKID*******
************************&Timestamp=1465185768&Version=2017-03-12
```

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机:查看实例列表(DescribeInstances)的请求域名为：cvm.tencentcloudapi.com。实际的请求域名根据接口所属模块的不同而不同，详见各接口说明。
3. 请求路径: 当前版本云API的请求路径固定为 / 。
4. 请求字符串: 即上一步生成的请求字符串。

签名原文串的拼接规则为： 请求方法 + 请求主机 +请求路径 + ? + 请求字符串 。

示例的拼接结果为:

```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-gu
angzhou&SecretId=AKID********************************&Timestamp=1465185768&Version=2017-03-12
```

2.4. 生成签名串

此步骤生成签名串。
首先使用 HMAC-SHA1 算法对上一步中获得的**签名原文字符串**进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例:

```
$secretKey = '********************************';
$srcStr = 'GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&R
egion=ap-guangzhou&SecretId=AKID********************************&Timestamp=1465185768&Version=2017-03-12';
```

```
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为：

```
7RAM2xfNMO9EiVTNmPg06MRnCvQ=
```

使用其它程序设计语言开发时，可用上面示例中的原文进行签名验证，得到的签名串与例子中的一致即可。

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 7RAM2xfNMO9EiVTNmPg06MRnCvQ= ，最终得到的签名串请求参数（Signature）为：7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D，它将用于生成最终的请求 URL。

注意：如果用户的请求方法是 GET，或者请求方法为 POST 同时 Content-Type 为 application/x-www-form-urlencoded，则发送请求时所有请求参数的值均需要做 URL 编码，参数键和=符号不需要编码。非 ASCII 字符在 URL 编码前需要先以 UTF-8 进行编码。

注意：有些编程语言的网络库会自动为所有参数进行 urlencode，在这种情况下，就不需要对签名串进行 URL 编码了，否则两次 URL 编码会导致签名失败。

注意：其他参数值也需要进行编码，编码采用 RFC 3986。使用 %XY 对特殊字符例如汉字进行百分比编码，其中“X”和“Y”为十六进制字符（0-9 和大写字母 A-F），使用小写将引发错误。

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理。

错误代码	错误描述
AuthFailure.SignatureExpire	签名过期
AuthFailure.SecretIdNotFound	密钥不存在
AuthFailure.SignatureFailure	签名错误
AuthFailure.TokenFailure	token 错误
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）

5. 签名演示

在实际调用 API 3.0 时，推荐使用配套的腾讯云 SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java
- PHP
- Go
- NodeJS
- .NET
- C++
- Ruby

下面提供了不同产品的生成签名 demo，您可以找到对应的产品参考签名的生成：

- Signature Demo

为了更清楚的解释签名过程，下面以实际编程语言为例，将上述的签名过程具体实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK 。

最终输出的 url 可能为：`https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-`

guangzhou&SecretId=AKID********************************&Signature=7RAM2xfNMO9EiVTNmPg06MRnCvQ%3D&Timestamp=1465185768&Version=2017-03-12 。

注意：由于示例中的密钥是虚构的，时间戳也不是系统当前时间，因此如果将此 url 在浏览器中打开或者用 curl 等命令调用时会返回鉴权错误：签名过期。为了得到一个可以正常返回的 url ，需要修改示例中的 SecretId 和 SecretKey 为真实的密钥，并使用系统当前时间戳作为 Timestamp 。

注意：在下面的示例中，不同编程语言，甚至同一语言每次执行得到的 url 可能都有所不同，表现为参数的顺序不同，但这并不影响正确性。只要所有参数都在，且签名计算正确即可。

注意：以下代码仅适用于 API 3.0，不能直接用于其他的签名流程，请以对应的实际文档为准。

Java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TencentCloudAPIDemo {
private final static String CHARSET = "UTF-8";

public static String sign(String s, String key, String method) throws Exception {
Mac mac = Mac.getInstance(method);
SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAlgorithm());
mac.init(secretKeySpec);
byte[] hash = mac.doFinal(s.getBytes(CHARSET));
return DatatypeConverter.printBase64Binary(hash);
}

public static String getStringToSign(TreeMap<String, Object> params) {
StringBuilder s2s = new StringBuilder("GETcvm.tencentcloudapi.com/?");
// 签名时要求对参数进行字典排序，此处用TreeMap保证顺序
for (String k : params.keySet()) {
s2s.append(k).append("=").append(params.get(k).toString()).append("&");
}
return s2s.toString().substring(0, s2s.length() - 1);
}

public static String getUrl(TreeMap<String, Object> params) throws UnsupportedEncodingException {
StringBuilder url = new StringBuilder("https://cvm.tencentcloudapi.com/?");
// 实际请求的url中对参数顺序没有要求
for (String k : params.keySet()) {
// 需要对请求串进行urlencode，由于key都是英文字母，故此处仅对其value进行urlencode
url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).append("&");
}
return url.toString().substring(0, url.length() - 1);
}

public static void main(String[] args) throws Exception {
TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
// 实际调用时应当使用随机数，例如: params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
params.put("Nonce", 11886); // 公共参数
// 实际调用时应当使用系统当前时间，例如: params.put("Timestamp", System.currentTimeMillis() / 1000);
params.put("Timestamp", 1465185768); // 公共参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
params.put("SecretId", System.getenv("TENCENTCLOUD_SECRET_ID")); // 公共参数
```

```
params.put("Action", "DescribeInstances"); // 公共参数
params.put("Version", "2017-03-12"); // 公共参数
params.put("Region", "ap-guangzhou"); // 公共参数
params.put("Limit", 20); // 业务参数
params.put("Offset", 0); // 业务参数
params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
params.put("Signature", sign(getStringToSign(params), System.getenv("TENCENTCLOUD_SECRET_KEY"), "HmacSHA1")); // 公共参数
System.out.println(getUrl(params));
}
}
```

Python

注意：如果是在 Python 2 环境中运行，需要先安装 requests 依赖包： `pip install requests` 。

```
# -*- coding: utf8 -*-
import base64
import hashlib
import hmac
import os
import time

import requests

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = os.environ.get("TENCENTCLOUD_SECRET_ID")
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secret_key = os.environ.get("TENCENTCLOUD_SECRET_KEY")

def get_string_to_sign(method, endpoint, params):
s = method + endpoint + "/?"
query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
return s + query_str

def sign_str(key, s, method):
hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
return base64.b64encode(hmac_str)

if __name__ == '__main__':
endpoint = "cvm.tencentcloudapi.com"
data = {
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : secret_id,
'Timestamp' : 1465185768, # int(time.time())
'Version': '2017-03-12'
}
s = get_string_to_sign("GET", endpoint, data)
data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
print(data["Signature"])
# 此处会实际调用，成功后可能产生计费
```

```
# resp = requests.get("https://" + endpoint, params=data)
# print(resp.url)
```

Golang

```
package main

import (
"bytes"
"crypto/hmac"
"crypto/sha1"
"encoding/base64"
"fmt"
"os"
"sort"
"strconv"
)

func main() {
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secretId := os.Getenv("TENCENTCLOUD_SECRET_ID")
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
secretKey := os.Getenv("TENCENTCLOUD_SECRET_KEY")
params := map[string]string{
"Nonce": "11886",
"Timestamp": strconv.Itoa(1465185768),
"Region": "ap-guangzhou",
"SecretId": secretId,
"Version": "2017-03-12",
"Action": "DescribeInstances",
"InstanceIds.0": "ins-09dx96dg",
"Limit": strconv.Itoa(20),
"Offset": strconv.Itoa(0),
}

var buf bytes.Buffer
buf.WriteString("GET")
buf.WriteString("cvm.tencentcloudapi.com")
buf.WriteString("/")
buf.WriteString("?")

// sort keys by ascii asc order
keys := make([]string, 0, len(params))
for k, _ := range params {
keys = append(keys, k)
}
sort.Strings(keys)

for i := range keys {
k := keys[i]
buf.WriteString(k)
buf.WriteString("=")
buf.WriteString(params[k])
buf.WriteString("&")
}
buf.Truncate(buf.Len() - 1)
```

```
hashed := hmac.New(sha1.New, []byte(secretKey))
hashed.Write(buf.Bytes())

fmt.Println(base64.StdEncoding.EncodeToString(hashed.Sum(nil)))
}
```

PHP

```
<?php
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
$secretId = getenv("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
$secretKey = getenv("TENCENTCLOUD_SECRET_KEY");
$param["Nonce"] = 11886;//rand();
$param["Timestamp"] = 1465185768;//time();
$param["Region"] = "ap-guangzhou";
$param["SecretId"] = $secretId;
$param["Version"] = "2017-03-12";
$param["Action"] = "DescribeInstances";
$param["InstanceIds.0"] = "ins-09dx96dg";
$param["Limit"] = 20;
$param["Offset"] = 0;

ksort($param);

$signStr = "GETcvm.tencentcloudapi.com/?";
foreach ( $param as $key => $value ) {
$signStr = $signStr . $key . "=" . $value . "&";
}
$signStr = substr($signStr, 0, -1);

$signature = base64_encode(hash_hmac("sha1", $signStr, $secretKey, true));
echo $signature.PHP_EOL;
// need to install and enable curl extension in php.ini
// $param["Signature"] = $signature;
// $url = "https://cvm.tencentcloudapi.com/?".http_build_query($param);
// echo $url.PHP_EOL;
// $ch = curl_init();
// curl_setopt($ch, CURLOPT_URL, $url);
// $output = curl_exec($ch);
// curl_close($ch);
// echo json_decode($output);
```

Ruby

```
# -*- coding: UTF-8 -*-
# require ruby>=2.3.0
require 'time'
require 'openssl'
require 'base64'

# 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
secret_id = ENV["TENCENTCLOUD_SECRET_ID"]
# 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
```

```
secret_key = ENV["TENCENTCLOUD_SECRET_KEY"]

method = 'GET'
endpoint = 'cvm.tencentcloudapi.com'
data = {
'Action' => 'DescribeInstances',
'InstanceIds.0' => 'ins-09dx96dg',
'Limit' => 20,
'Nonce' => 11886,
'Offset' => 0,
'Region' => 'ap-guangzhou',
'SecretId' => secret_id,
'Timestamp' => 1465185768, # Time.now.to_i
'Version' => '2017-03-12',
}
sign = method + endpoint + '/?'
params = []
data.sort.each do |item|
params << "#{item[0]}=#{item[1]}"
end
sign += params.join('&')
digest = OpenSSL::Digest.new('sha1')
data['Signature'] = Base64.encode64(OpenSSL::HMAC.digest(digest, secret_key, sign))
puts data['Signature']

# require 'net/http'
# uri = URI('https://' + endpoint)
# uri.query = URI.encode_www_form(data)
# p uri
# res = Net::HTTP.get_response(uri)
# puts res.body
```

DotNet

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Security.Cryptography;
using System.Text;

public class Application {
public static string Sign(string signKey, string secret)
{
string signRet = string.Empty;
using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
{
byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
signRet = Convert.ToBase64String(hash);
}
return signRet;
}
public static string MakeSignPlainText(SortedDictionary<string, string> requestParams, string requestMethod, string reques
tHost, string requestPath)
{
string retStr = "";
retStr += requestMethod;
```

```
retStr += requestHost;
retStr += requestPath;
retStr += "?";
string v = "";
foreach (string key in requestParams.Keys)
{
v += string.Format("{0}={1}&", key, requestParams[key]);
}
retStr += v.TrimEnd('&');
return retStr;
}

public static void Main(string[] args)
{
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
string SECRET_ID = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_ID");
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
string SECRET_KEY = Environment.GetEnvironmentVariable("TENCENTCLOUD_SECRET_KEY");

string endpoint = "cvm.tencentcloudapi.com";
string region = "ap-guangzhou";
string action = "DescribeInstances";
string version = "2017-03-12";
double RequestTimestamp = 1465185768; // 时间戳 2019-02-26 00:44:25,此参数作为示例，以实际为准
// long timestamp = ToTimestamp() / 1000;
// string requestTimestamp = timestamp.ToString();
Dictionary<string, string> param = new Dictionary<string, string>();
param.Add("Limit", "20");
param.Add("Offset", "0");
param.Add("InstanceIds.0", "ins-09dx96dg");
param.Add("Action", action);
param.Add("Nonce", "11886");
// param.Add("Nonce", Math.Abs(new Random().Next()).ToString());

param.Add("Timestamp", RequestTimestamp.ToString());
param.Add("Version", version);

param.Add("SecretId", SECRET_ID);
param.Add("Region", region);
SortedDictionary<string, string> headers = new SortedDictionary<string, string>(param, StringComparer.Ordinal);
string sigInParam = MakeSignPlainText(headers, "GET", endpoint, "/");
string sigOutParam = Sign(SECRET_KEY, sigInParam);
Console.WriteLine(sigOutParam);
}
}
```

NodeJS

```
const crypto = require('crypto');

function get_req_url(params, endpoint){
params['Signature'] = encodeURIComponent(params['Signature']);
const url_strParam = sort_params(params)
return "https://" + endpoint + "/?" + url_strParam.slice(1);
}
```

```
function formatSignString(reqMethod, endpoint, path, strParam){
let strSign = reqMethod + endpoint + path + "?" + strParam.slice(1);
return strSign;
}
function sha1(secretKey, strsign){
let signMethodMap = {'HmacSHA1': "sha1"};
let hmac = crypto.createHmac(signMethodMap['HmacSHA1'], secretKey || "");
return hmac.update(Buffer.from(strsign, 'utf8')).digest('base64')
}

function sort_params(params){
let strParam = "";
let keys = Object.keys(params);
keys.sort();
for (let k in keys) {
//k = k.replace(/_/g, '.');
strParam += ("&" + keys[k] + "=" + params[keys[k]]);
}
return strParam
}

function main(){
// 密钥参数
// 需要设置环境变量 TENCENTCLOUD_SECRET_ID，值为示例的 AKID********************************
const SECRET_ID = process.env.TENCENTCLOUD_SECRET_ID
// 需要设置环境变量 TENCENTCLOUD_SECRET_KEY，值为示例的 ********************************
const SECRET_KEY = process.env.TENCENTCLOUD_SECRET_KEY

const endpoint = "cvm.tencentcloudapi.com"
const Region = "ap-guangzhou"
const Version = "2017-03-12"
const Action = "DescribeInstances"
const Timestamp = 1465185768 // 时间戳 2016-06-06 12:02:48，此参数作为示例，以实际为准
// const Timestamp = Math.round(Date.now() / 1000)
const Nonce = 11886 // 随机正整数
//const nonce = Math.round(Math.random() * 65535)

let params = {};
params['Action'] = Action;
params['InstanceIds.0'] = 'ins-09dx96dg';
params['Limit'] = 20;
params['Offset'] = 0;
params['Nonce'] = Nonce;
params['Region'] = Region;
params['SecretId'] = SECRET_ID;
params['Timestamp'] = Timestamp;
params['Version'] = Version;

// 1. 对参数排序，并拼接请求字符串
strParam = sort_params(params)

// 2. 拼接签名原文字符串
const reqMethod = "GET";
const path = "/";
strSign = formatSignString(reqMethod, endpoint, path, strParam)
// console.log(strSign)
```

```
// 3. 生成签名串
params['Signature'] = sha1(SECRET_KEY, strSign)
console.log(params['Signature'])

// 4. 进行url编码并拼接请求url
// const req_url = get_req_url(params, endpoint)
// console.log(params['Signature'])
// console.log(req_url)
}
main()
```

返回结果

最近更新时间：2024-03-12 19:53:05

云 API 3.0 接口默认返回 JSON 数据，返回非 JSON 格式的接口会在文档中做出说明。返回 JSON 数据时最大限制为 50 MB，如果返回的数据超过最大限制，请求会失败并返回内部错误。请根据接口文档中给出的过滤功能（例如时间范围）或者分页功能，控制返回数据不要过大。

注意：目前只要请求被服务端正常处理了，响应的 HTTP 状态码均为200。例如返回的消息体里的错误码是签名失败，但 HTTP 状态码是200，而不是401。

正确返回结果

以云服务器的接口查看实例状态列表 (DescribeInstancesStatus) 2017-03-12 版本为例，若调用成功，其可能的返回如下为：

```
{
"Response": {
"TotalCount": 0,
"InstanceStatusSet": [],
"RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
}
}
```

- Response 及其内部的 RequestId 是固定的字段，无论请求成功与否，只要 API 处理了，则必定会返回。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系 腾讯云客服 或 提交工单，并提供该 ID 来解决问题。
- 除了固定的字段外，其余均为具体接口定义的字段，不同的接口所返回的字段参见接口文档中的定义。此例中的 TotalCount 和 InstanceStatusSet 均为 DescribeInstancesStatus 接口定义的字段，由于调用请求的用户暂时还没有云服务器实例，因此 TotalCount 在此情况下的返回值为 0， InstanceStatusSet 列表为空。

错误返回结果

若调用失败，其返回值示例如下为：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

- Error 的出现代表着该请求调用失败。Error 字段连同其内部的 Code 和 Message 字段在调用失败时是必定返回的。
- Code 表示具体出错的错误码，当请求出错时可以先根据该错误码在公共错误码和当前接口对应的错误码列表里面查找对应原因和解决方案。
- Message 显示出了这个错误发生的具体原因，随着业务发展或体验优化，此文本可能会经常保持变更或更新，用户不应依赖这个返回值。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系 腾讯云客服 或 提交工单，并提供该 ID 来解决问题。

公共错误码

返回结果中如果存在 Error 字段，则表示调用 API 接口失败。 Error 中的 Code 字段表示错误码，所有业务都可能出现的错误码为公共错误码。完整的错误码列表请参考本产品“API 文档”目录下的“错误码”页面。

参数类型

最近更新时间：2022-08-10 06:52:45

目前腾讯云 API 3.0 输入参数和输出参数支持如下几种数据格式：

- String: 字符串。
- Integer：整型，上限为无符号64位整数。SDK 3.0 不同编程语言支持的类型有所差异，建议以所使用编程语言的最大整型定义，例如 `Golang` 的 `uint64`。
- Boolean：布尔型。
- Float：浮点型。
- Double：双精度浮点型。
- Date：字符串，日期格式。例如：2022-01-01。
- Timestamp：字符串，时间格式。例如：2022-01-01 00:00:00。
- Timestamp ISO8601：ISO 8601 是由国际标准化组织（International Organization for Standardization，ISO）发布的关于日期和时间格式的国际标准，对应国标《GB/T 7408-2005数据元和交换格式信息交换日期和时间表示法》。建议以所使用编程语言的标准库进行格式解析。例如：2022-01-01T00:00:00+08:00。
- Binary：二进制内容，需要以特定协议请求和解析。

节点相关接口
删除节点

最近更新时间：2025-04-25 02:01:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(DeleteNodes)用于删除指定集群中一个或者多个计算节点或者登录节点。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteNodes。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
NodeIds.N	是	Array of String	节点ID。 示例值：["node-xxx"]

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除节点

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DeleteNodes
&ClusterId=hpc-5lyv94lq
&NodeIds.0=ins-jhf9c1gi
&NodeIds.1=ins-qmzlqk70
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "f4d90874-f565-463d-ba1d-c707517eeaa1"
```

```
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.NodeId	无法找到ID对应节点。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。
UnsupportedOperation.InvalidNodeRole	类型节点不支持当前操作。
UnsupportedOperation.NodeStatusNotSupport	节点状态不支持此操作。

添加节点

最近更新时间：2025-05-15 02:06:32

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(AddNodes)用于添加一个或者多个计算节点或者登录节点到指定集群。

默认接口请求频率限制：3次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：AddNodes。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
Placement	是	Placement	集群中实例所在的位置。 示例值：ap-guangzhou-1
ClusterId	是	String	集群ID。 示例值：hpc-xxx
VirtualPrivateCloud	是	VirtualPrivateCloud	私有网络相关信息配置。 示例值：vpc-xxx
Count	是	Integer	添加节点数量。 示例值：1
ImageId	否	String	指定有效的镜像ID，格式形如img-xxx。目前仅支持公有镜像和特定自定义镜像。 示例值：img-xxx
InstanceChargeType	否	String	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：PREPAID
InstanceChargePrepaid	否	InstanceChargePrepaid	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{ "Period": 36, "RenewFlag": "NOTIFY_AND_AUTO_RENEW" }
InstanceType	否	String	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S5.LARGE4
SystemDisk.N	否	Array of SystemDisk	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值：{ "DiskSize": 50 }
DataDisks.N	否	Array of DataDisk	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20

参数名称	必选	类型	描述
			块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值：[{ "DiskSize": 50, "DiskType": "CLOUD_PREMIUM" }]
InternetAccessible	否	InternetAccessible	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{ "PublicIpAssigned": "false" }
InstanceName	否	String	节点显示名称。 不指定节点显示名称则默认显示‘未命名’。 最多支持60个字符。 示例值：server_{R:3}
LoginSettings	否	LoginSettings	集群登录设置。 示例值：{ "KeyIds": ["skey-9tzxxb4j"] }
SecurityGroupIds.N	否	Array of String	集群中实例所属安全组。该参数可以通过调用 DescribeSecurityGroups 的返回值中的sgId字段来获取。若不指定该参数，则绑定默认安全组。 示例值：["sg-ajhn9qtq"]
ClientToken	否	String	用于保证请求幂等性的字符串。该字符串由客户生成，需保证不同请求之间唯一，最大值不超过64个ASCII字符。若不指定该参数，则无法保证请求的幂等性。 示例值：system-f3827db9-c58a-49cc-bf10-33fc1923a34a
QueueName	否	String	队列名称。不指定则为默认队列。 • SLURM默认队列为：compute。 示例值：compute
NodeRole	否	String	添加节点角色。默认值：Compute • Compute：计算节点。 • Login：登录节点。 示例值：Compute
DryRun	否	Boolean	是否只预检此次请求。 true：发送检查请求，不会创建实例。检查项包括是否填写了必需参数，请求格式，业务限制和云服务器库存。 如果检查不通过，则返回对应错误码； 如果检查通过，则返回RequestId. false（默认）：发送正常请求，通过检查后直接创建实例 示例值：False
NodeType	否	String	添加节点类型。默认取值：STATIC。 • STATIC：静态节点，不会参与弹性伸缩流程。 • DYNAMIC：弹性节点，会被弹性缩容的节点。管控节点和登录节点不支持此参数。 示例值：STATIC

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 添加节点

往集群ID为hpc-52nkfau6的集群的compute队列增加一个计算节点。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
```

```
X-TC-Action: AddNodes
<公共请求参数>

{
"Count": "1",
"VirtualPrivateCloud": {
"SubnetId": "subnet-r0zpktaa",
"VpcId": "vpc-r9jw2jzr"
},
"Placement": {
"Zone": "ap-guangzhou-2"
},
"ClusterId": "hpc-52nkfau6",
"ImageId": "img-3la7wgnt",
"InstanceChargeType": "SPOTPAID",
"InstanceType": "S2.SMALL2",
"NodeRole": "Compute",
"QueueName": "compute"
}
```

输出示例

```
{
"Response": {
"RequestId": "b2ac2379-6453-4eab-8f63-7ade00cb67b0"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
InternalError.CallCvm	cvm调用失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooSmall	参数值过小。
MissingParameter	缺少参数错误。
ResourceInsufficient	资源不足。
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.Queue	无法找到指定队列。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

查询指定集群节点列表

最近更新时间：2025-04-25 02:01:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (DescribeNodes) 用于查询指定集群节点概览信息列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeNodes。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
Filters.N	否	Array of Filter	• **queue-name** 按照【**队列名称**】进行过滤。队列名称形如：compute。 类型：String 必选：否 • **node-role** 按照【**节点角色**】进行过滤。节点角色形如：Manager。（Manager：管控节点。Compute：计算节点。Login：登录节点。ManagerBackup：备用管控节点。） 类型：String 必选：否 • **node-type** 按照【**节点类型**】进行过滤。节点类型形如：STATIC。(STATIC：静态节点。DYNAMIC：弹性节点。) 类型：String 必选：否 每次请求的 `Filters` 的上限为10， `Filter.Values` 的上限为5。
Offset	否	Integer	偏移量，默认为0。关于`Offset`的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于`Limit`的更进一步介绍请参考 API 简介中的相关小节。 示例值：20

3. 输出参数

参数名称	类型	描述
NodeSet	Array of NodeOverview	节点概览信息列表。
TotalCount	Integer	符合条件的节点数量。 示例值：1

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询指定集群节点概览信息。

查询指定集群节点概览信息，集群ID为hpc-ggaqjyax。

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DescribeNodes
&ClusterId=hpc-ggaqjyax
&Offset=0
&Limit=1
&<公共请求参数>
```

输出示例

```
{
"Response": {
"NodeSet": [
{
"InstanceId": "ins-j3n6kiae",
"Zone": "ap-chongqing-1",
"NodeState": "RUNNING",
"ImageId": "img-l8og963d",
"QueueName": "compute",
"NodeRole": "Compute",
"NodeType": "DYNAMIC"
}
],
"TotalCount": 3,
"RequestId": "6793673e-3bce-4fbb-adea-923a82267da2"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee

- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.InvalidFilterNotSupportedName	不支持指定过滤器的键。
InvalidParameterValue.LimitExceeded	参数值数量超过限制。
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooSmall	参数值过小。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
RequestLimitExceeded	请求的次数超过了频率限制。
UnknownParameter	未知参数错误。
UnsupportedOperation.ParameterTooLarge	参数值过大，不支持此操作。
UnsupportedOperation.ParameterTooSmall	参数值过小，不支持此操作。

存储选项相关接口
添加集群存储选项

最近更新时间：2025-04-25 02:01:32

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（AddClusterStorageOption）用于添加集群存储选项信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：AddClusterStorageOption。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
StorageOption	是	StorageOption	集群存储选项。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 添加集群存储选项

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: AddClusterStorageOption
<公共请求参数>

{
"ClusterId": "hpc-eq97tf4u",
"StorageOption": {
"CFSOptions": [
{
"LocalPath": "/data/",
"RemotePath": "172.30.3.90:/",
```

```
"Protocol": "NFS 4.0",
"StorageType": "SD"
},
{
"LocalPath": "/tmp/",
"RemotePath": "172.30.2.180@tcp0:/4fe1839b/cfs",
"Protocol": "TURBO",
"StorageType": "TB"
}
]
}
}
```

输出示例

```
{
"Response": {
"RequestId": "b2ac2379-6453-4eab-8f63-7ade00cb67b0"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

删除集群存储选项

最近更新时间：2025-04-25 02:01:32

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (DeleteClusterStorageOption) 用于删除集群存储选项信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteClusterStorageOption。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
LocalPath	是	String	本地挂载路径。 示例值：/data/

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除集群存储选项信息。

删除本地挂载点为/data/的存储选项信息。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DeleteClusterStorageOption
<公共请求参数>

{
"ClusterId": "hpc-5lyv94lq",
"LocalPath": "/data/"
}
```

输出示例

```
{
"Response": {
"RequestId": "2aeaea7e-9fc4-4c17-8a9b-50343b711003"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见 公共错误码。

查询集群存储选项

最近更新时间：2025-04-25 02:01:32

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (DescribeClusterStorageOption) 用于查询集群存储选项信息。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeClusterStorageOption。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq

3. 输出参数

参数名称	类型	描述
StorageOption	StorageOptionOverview	集群存储选项信息概览。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询集群存储选项

查询集群ID为hpc-5lyv94lq的集群存储选项信息。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeClusterStorageOption
<公共请求参数>

{
"ClusterId": "hpc-5lyv94lq"
}
```

输出示例

```
{
"Response": {
"StorageOption": {
"CFSOptions": [
{
"LocalPath": "/opt/",
"RemotePath": "172.30.3.90:/j25ey5tz/hpc-kjddsnfa/opt/",
"Protocol": "NFS 3.0",
"StorageType": "SD"
}
],
"GooseFSOptions": [
{
"LocalPath": "/data/",
"RemotePath": "/xxx/cfs/",
"Masters": [
"172.30.4.63:9202",
"172.30.0.128:9202"
]
}
]
},
"RequestId": "2aeaea7e-9fc4-4c17-8a9b-50343b712993"
}
}
```


5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
ResourceNotFound.ClusterId	集群不存在。

错误码	描述
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

集群相关接口
删除集群

最近更新时间：2025-04-25 02:01:30

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（DeleteCluster）用于删除一个指定的集群。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteCluster。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除集群

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DeleteCluster
&ClusterId=hpc-5lyv94lq
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "8b71050a-55c1-4f3b-bf66-5e4f8510a5a0"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
ResourceNotFound.ClusterId	集群不存在。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

查询集群列表

最近更新时间：2025-04-25 02:01:30

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（DescribeClusters）用于查询集群列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeClusters。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterIds.N	否	Array of String	集群ID列表。通过该参数可以指定需要查询信息的集群列表。 如果您不指定该参数，则返回Limit数量以内的集群信息。 示例值：['hpc-xxx']
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：20

3. 输出参数

参数名称	类型	描述
ClusterSet	Array of ClusterOverview	集群概览信息列表。 示例值：[{"ClusterId":"hpc-qrb7ybn0","ClusterName":"unnamed","ClusterStatus":"RUNNING"}]
TotalCount	Integer	集群数量。 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询集群列表

查询集群列表。

输入示例

```
https://thpc.tencentcloudapi.com/?Action=DescribeClusters
&Offset=0
&Limit=1
&<公共请求参数>
```

输出示例


```
{
"Response": {
"ClusterSet": [
{
"ClusterId": "hpc-qrb7ybn0",
"ClusterName": "unnamed",
"ClusterStatus": "RUNNING",
"Placement": {
"Zone": "ap-guangzhou-2"
},
"CreateTime": "2021-12-07T03:29:09Z",
"SchedulerType": "SGE",
"VpcId": "vpc-xxxxxxxx",
"ComputeNodeCount": 1,
"LoginNodeSet": [
{
"NodeId": "ins-jehc407m"
}
],
"LoginNodeCount": 1,
"ComputeNodeSet": [
{
"NodeId": "ins-jfhc307q"
}
],
"ManagerNodeCount": 1,
"ManagerNodeSet": [
{
"NodeId": "ins-cv6ygpkq"
}
]
}
],
"TotalCount": 1,
"RequestId": "8a39c8e3-129b-4628-b763-ef96caaa2f4d"
}
}
```


5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee

- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。

创建集群

最近更新时间：2025-05-15 02:06:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口 (CreateCluster) 用于创建并启动集群。

- 本接口为异步接口， 当创建集群请求下发成功后会返回一个集群ID和一个RequestId，此时创建集群操作并未立即完成。在此期间集群的状态将会处于“PENDING”或者“INITING”，集群创建结果可以通过调用 DescribeClusters 接口查询，如果集群状态(ClusterStatus)变为“RUNNING(运行中)”，则代表集群创建成功，“INIT_FAILED”代表集群创建失败。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：CreateCluster。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
Placement	是	Placement	集群中实例所在的位置。 示例值：{ "Zone": "ap-guangzhou-6" }
ManagerNode	否	ManagerNode	指定管理节点。 示例值：{"InstanceType": "S2.SMALL2"}
ManagerNodeCount	否	Integer	指定管理节点的数量。默认取值：1。取值范围：1~2。 示例值：1
ComputeNode	否	ComputeNode	指定计算节点。 示例值：{"InstanceType": "S2.SMALL2"}
ComputeNodeCount	否	Integer	指定计算节点的数量。默认取值：0。 示例值：1
SchedulerType	否	String	调度器类型。默认取值：SLURM。 • SLURM：SLURM调度器。 示例值：SLURM
ImageId	否	String	指定有效的镜像ID，格式形如img-xxx。目前支持部分公有镜像和自定义镜像。 示例值：img-xxx
VirtualPrivateCloud	否	VirtualPrivateCloud	私有网络相关信息配置。 示例值：{ "VpcId": "vpc-rhfaxx31", "SubnetId": "subnet-x7vxgqe" }
LoginSettings	否	LoginSettings	集群登录设置。 示例值：{ "KeyIds": ["skey-9tzxxb4j"] }
SecurityGroupIds.N	否	Array of String	集群中实例所属安全组。该参数可以通过调用 DescribeSecurityGroups 的返回值中的sgId字段来获 则绑定默认安全组。 示例值：["sg-ajhn9qtq"]
ClientToken	否	String	用于保证请求幂等性的字符串。该字符串由客户生成，需保证不同请求之间唯一，最大值不超过64个AS 数，则无法保证请求的幂等性。

参数名称	必选	类型	描述
			示例值：system-f3827db9-c58a-49cc-bf10-33fc1923a34a
DryRun	否	Boolean	是否只预检此次请求。 true：发送检查请求，不会创建实例。检查项包括是否填写了必需参数，请求格式，业务限制和云服务器如果检查不通过，则返回对应错误码； 如果检查通过，则返回RequestId. false（默认）：发送正常请求，通过检查后直接创建实例 示例值：False
AccountType	否	String	域名字服务类型。默认取值：NIS。 • NIS：NIS域名字服务。 示例值：NIS
ClusterName	否	String	集群显示名称。 示例值：cluster-test
StorageOption	否	StorageOption	集群存储选项 示例值：{"CFSOptions":[{"LocalPath":"/mnt/","RemotePath":"127.0.0.1@tcp0:/test/cfs/","Protocol":"TURBO","S
LoginNode	否	LoginNode	指定登录节点。 示例值：{"InstanceType": "S2.SMALL2"}
LoginNodeCount	否	Integer	指定登录节点的数量。默认取值：0。取值范围：0~10。 示例值：1
Tags.N	否	Array of Tag	创建集群时同时绑定的标签对说明。 示例值：[{"Key": "test", "Value": "test"}]
AutoScalingType	否	String	弹性伸缩类型。 • AS：集群自动扩缩容由弹性伸缩产品实现。 • THPC_AS：集群自动扩缩容由THPC产品内部实现。 示例值：THPC_AS

3. 输出参数

参数名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-5lyv94lq
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 创建集群

创建一个管控节点和两个计算节点的集群。调度器为：SLURM。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: CreateCluster
<公共请求参数>

{
"ManagerNodeCount": "1",
"Placement": {
```

```
"Zone": "ap-guangzhou-2"
},
"SchedulerType": "SLURM",
"ImageId": "img-3la7wgnt",
"ComputeNode": {
"InstanceChargeType": "SPOTPAID",
"InstanceType": "S2.SMALL2"
},
"ComputeNodeCount": "2",
"ManagerNode": {
"InstanceType": "S2.SMALL2"
}
}
```

输出示例

```
{
"Response": {
"ClusterId": "hpc-5lyv94lq",
"RequestId": "b2ac2379-6453-4eab-8f63-7ade00cb67b0"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.CallCvm	cvm调用失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue	参数取值错误。

错误码	描述
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.ParametersNotSupported	字段不支持此值。
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooShort	无效参数值。参数值太短。
InvalidParameterValue.TooSmall	参数值过小。
ResourceNotFound.ImageId	无法找到镜像ID。
UnsupportedOperation	操作不支持。

查询集群活动历史记录

最近更新时间：2025-04-25 02:01:30

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口（DescribeClusterActivities）用于查询集群活动历史记录列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeClusterActivities。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。通过该参数指定需要查询活动历史记录的集群。 示例值：hpc-myd8fgsc
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：20

3. 输出参数

参数名称	类型	描述
ClusterActivitySet	Array of ClusterActivity	集群活动历史记录列表。
TotalCount	Integer	集群活动历史记录数量。 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询集群活动历史记录

根据集群ID查询集群活动历史记录。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeClusterActivities
<公共请求参数>
```

```
{
"ClusterId": "hpc-0yd8fqsc",
"Limit": "1",
"Offset": "0"
}
```

输出示例

```
{
"Response": {
"ClusterActivitySet": [
{
"ClusterId": "hpc-myd8fgsc",
"ActivityId": "cha-gvzj0zbd",
"ActivityType": "TerminateNodes",
"ActivityStatus": "SUCCESSFUL",
"ActivityStatusCode": "ActivitySuccess",
"ResultDetail": "Activity success.",
"Cause": "DeleteCluster",
"Description": "删除指定集群，销毁实例，销毁所有节点。",
"RelatedNodeActivitySet": [
{
"NodeInstanceId": "ins-1zll2of0",
"NodeActivityStatus": "SUCCESSFUL",
"NodeActivityStatusCode": "ActivitySuccess",
"NodeActivityStatusReason": "Activity success."
},
{
"NodeInstanceId": "ins-ig2bew40",
"NodeActivityStatus": "SUCCESSFUL",
"NodeActivityStatusCode": "ActivitySuccess",
"NodeActivityStatusReason": "Activity success."
}
],
"StartTime": "2021-11-01T02:17:20Z",
"EndTime": "2021-11-01T02:17:38Z"
}
],
"TotalCount": 1,
"RequestId": "7fa864e6-cf1a-4962-8aa9-f68abfa31a00"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee

- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。

弹性伸缩相关接口
查询弹性伸缩配置信息

最近更新时间：2025-04-25 02:01:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(DescribeAutoScalingConfiguration)用于查询集群弹性伸缩配置信息。本接口仅适用于弹性伸缩类型为THPC_AS的集群。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeAutoScalingConfiguration。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq

3. 输出参数

参数名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-rv7hahw2
ExpansionBusyTime	Integer	任务连续等待时间，队列的任务处于连续等待的时间。单位秒。 示例值：120
ShrinkIdleTime	Integer	节点连续空闲（未运行作业）时间，一个节点连续处于空闲状态时间。 示例值：300
QueueConfigs	Array of QueueConfigOverview	扩容队列配置概览列表。
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询扩缩容策略配置

查询集群ID为hpc-rv7hahw2的集群弹性伸缩配置信息。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeAutoScalingConfiguration
```

```
<公共请求参数>

{
"ClusterId": "hpc-rv7hahw2"
}
```

输出示例


```
{
"Response": {
"ClusterId": "hpc-rv7hahw2",
"ExpansionBusyTime": 120,
"ShrinkIdleTime": 300,
"QueueConfigs": [
{
"QueueName": "compute",
"MinSize": 0,
"MaxSize": 10,
"EnableAutoExpansion": false,
"EnableAutoShrink": false,
"ExpansionNodeConfigs": [
{
"InstanceType": "M5.SMALL8",
"Placement": {
"Zone": "ap-chongqing-1"
},
"InstanceChargeType": "POSTPAID_BY_HOUR",
"InstanceChargePrepaid": null,
"VirtualPrivateCloud": {
"VpcId": "vpc-r9jw2jzv",
"SubnetId": "subnet-r0zpktae"
},
"ImageId": "img-l8og963d",
"InternetAccessible": {
"InternetChargeType": "TRAFFIC_POSTPAID_BY_HOUR",
"InternetMaxBandwidthOut": 0
},
"SystemDisk": {
"DiskType": "CLOUD_BSSD",
"DiskSize": 50
},
"DataDisks": null
}
]
}
],
"RequestId": "935760b6-2a63-480d-9124-c5f5b9d4218b"
}
}
```


5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
ResourceNotFound.ClusterId	集群不存在。
UnsupportedOperation.AutoScalingType	弹性伸缩类型不支持此操作。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

设置弹性伸缩配置信息

最近更新时间：2025-04-25 02:01:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(SetAutoScalingConfiguration)用于为集群设置集群弹性伸缩配置信息。

默认接口请求频率限制：3次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：SetAutoScalingConfiguration。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
ExpansionBusyTime	否	Integer	任务连续等待时间，队列的任务处于连续等待的时间。单位秒。默认值120。 示例值：120
ShrinkIdleTime	否	Integer	节点连续空闲（未运行作业）时间，一个节点连续处于空闲状态时间。单位秒。默认值300。 示例值：300
QueueConfigs.N	否	Array of QueueConfig	扩容队列配置列表。 示例值：{"QueueName":"compute"}
DryRun	否	Boolean	是否只预检此次请求。 true：发送检查请求，不会绑定弹性伸缩组。检查项包括是否填写了必需参数，请求格式，业务限制。 如果检查不通过，则返回对应错误码； 如果检查通过，则返回RequestId。 false（默认）：发送正常请求，通过检查后直接绑定弹性伸缩组。 示例值：False

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 设置弹性伸缩配置

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
```

```
X-TC-Action: SetAutoScalingConfiguration
<公共请求参数>

{
"DryRun": "false",
"ShrinkIdleTime": "300",
"ExpansionBusyTime": "120",
"ClusterId": "hpc-5lyv94lq"
}
```

输出示例

```
{
"Response": {
"RequestId": "d1d90874-f565-463d-ba1d-c707517eece1"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InternalError.CallCAM	CAM服务调用失败。
InternalError.CallCvm	cvm调用失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.LimitExceeded	参数值数量超过限制。
InvalidParameterValue.TooSmall	参数值过小。
ResourceNotFound.ClusterId	集群不存在。
ResourceNotFound.ImageId	无法找到镜像ID。

错误码	描述
ResourceNotFound.Queue	无法找到指定队列。
UnsupportedOperation.AutoScalingType	弹性伸缩类型不支持此操作。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。
UnsupportedOperation.VpcIdConflict	vpc冲突，不支持当前操作。

绑定弹性伸缩组

最近更新时间：2025-04-25 02:01:32

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(BindAutoScalingGroup)用于为集群队列绑定弹性伸缩组

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：BindAutoScalingGroup。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
LaunchConfigurationId	是	String	弹性伸缩启动配置ID。 示例值：asc-jhf9c1gi
AutoScalingGroupId	是	String	弹性伸缩组ID。 示例值：asg-aesc6pcq
QueueName	否	String	队列名称。 示例值：all.q
ExpansionBusyTime	否	Integer	任务连续等待时间，队列的任务处于连续等待的时间。单位秒。默认值120。 示例值：120
ShrinkIdleTime	否	Integer	节点连续空闲（未运行作业）时间，一个节点连续处于空闲状态时间。单位秒。默认值300。 示例值：300
EnableAutoExpansion	否	Boolean	是否开启自动扩容，默认值true。 示例值：True
EnableAutoShrink	否	Boolean	是否开启自动缩容，默认值true。 示例值：True
DryRun	否	Boolean	是否只预检此次请求。 true：发送检查请求，不会绑定弹性伸缩组。检查项包括是否填写了必需参数，请求格式，业务限制。 如果检查不通过，则返回对应错误码； 如果检查通过，则返回RequestId。 false（默认）：发送正常请求，通过检查后直接绑定弹性伸缩组。 示例值：False

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 绑定弹性伸缩组

输入示例

```
https://thpc.tencentcloudapi.com/?Action=BindAutoScalingGroup
&ClusterId=hpc-5lyv94lq
&LaunchConfigurationId=asc-jhf9c1gi
&AutoScalingGroupId=asg-aesc6pcq
&ExpansionBusyTime=120
&ShrinkIdleTime=300
&DryRun=false
&<公共请求参数>
```

输出示例

```
{
"Response": {
"RequestId": "f4d90874-f565-463d-ba1d-c707517eeaa1"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.TooSmall	参数值过小。
ResourceNotFound.AutoScalingGroupId	无法找到弹性伸缩组ID。
ResourceNotFound.ClusterId	集群不存在。

错误码	描述
ResourceNotFound.LaunchConfigurationId	无法找到ID对应的弹性伸缩启动配置。
UnsupportedOperation.AutoScalingGroupAlreadyBinded	该伸缩组已绑定集群，请更换伸缩组。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

队列相关接口

查询队列列表

最近更新时间：2025-04-25 02:01:30

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(DescribeQueues)用于查询指定集群队列概览信息列表。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer 点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DescribeQueues。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-5lyv94lq
Offset	否	Integer	偏移量，默认为0。关于Offset的更进一步介绍请参考 API 简介中的相关小节。 示例值：0
Limit	否	Integer	返回数量，默认为20，最大值为100。关于Limit的更进一步介绍请参考 API 简介中的相关小节。 示例值：20

3. 输出参数

参数名称	类型	描述
QueueSet	Array of QueueOverview	队列概览信息列表。
TotalCount	Integer	符合条件的节点数量。 示例值：1
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 查询集群队列概览信息列表

查询集群队列概览信息列表。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeQueues
<公共请求参数>
```

```
{
"ClusterId": "hpc-prkxbd7c",
"Offset": 0,
"Limit": 20
}
```

输出示例

```
{
"Response": {
"QueueSet": [
{
"QueueName": "compute"
}
],
"TotalCount": 1,
"RequestId": "13ee84a5-7846-4682-8877-1c8f94962dd5"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.ParametersNotSupported	字段不支持此值。

错误码	描述
InvalidParameterValue.TooShort	无效参数值。参数值太短。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceNotFound	资源不存在。
UnknownParameter	未知参数错误。
UnsupportedOperation.ParameterTooLarge	参数值过大，不支持此操作。
UnsupportedOperation.ParameterTooSmall	参数值过小，不支持此操作。

删除队列

最近更新时间：2025-04-25 02:01:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(DeleteQueue)用于从指定集群删除队列。

- 本接口为目前只支持SchedulerType为SLURM的集群。
- 删除队列时，需要保证队列内不存在节点。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer　点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：DeleteQueue。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-prkxbd7c
QueueName	是	String	队列名称。 最多支持32个字符。 示例值：compute

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 删除队列

从集群删除指定队列。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DeleteQueue
<公共请求参数>

{
"ClusterId": "hpc-prkxbd7c",
```

```
"QueueName": "compute"
}
```

输出示例

```
{
"Response": {
"RequestId": "13ee84a5-7846-4682-8877-1c8f94962d10"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
FailedOperation	操作失败。
InternalError	内部错误。
InvalidParameter	参数错误。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue	参数取值错误。
InvalidParameterValue.LimitExceeded	参数值数量超过限制。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.ParametersNotSupported	字段不支持此值。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooShort	无效参数值。参数值太短。
InvalidParameterValue.TooSmall	参数值过小。

错误码	描述
InvalidParameterValue.ValueDuplicated	参数值重复。不支持此操作。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
ResourceInUse	资源被占用。
ResourceNotFound	资源不存在。
ResourceNotFound.Queue	无法找到指定队列。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。
UnsupportedOperation.QueueNotEmpty	队列内存在节点，不支持此操作。

添加队列

最近更新时间：2025-04-25 02:01:31

1. 接口描述

接口请求域名： thpc.tencentcloudapi.com 。

本接口(AddQueue)用于添加队列到指定集群。

- 本接口为目前只支持SchedulerType为SLURM的集群。
- 单个集群中队列数量上限为10个。

默认接口请求频率限制：20次/秒。

推荐使用 API Explorer

点击调试

API Explorer 提供了在线调用、签名验证、SDK 代码生成和快速检索接口等能力。您可查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：AddQueue。
Version	是	String	公共参数，本接口取值：2022-04-01。
Region	是	String	公共参数，详见产品支持的 地域列表。
ClusterId	是	String	集群ID。 示例值：hpc-prkxbd7c
QueueName	是	String	队列名称。 最多支持32个字符。 示例值：compute

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，由服务端生成，每次请求都会返回（若请求因其他原因未能抵达服务端，则该次请求不会获得 RequestId）。定位问题时需要提供该次请求的 RequestId。

4. 示例

示例1 添加队列

往指定集群添加队列。

输入示例

```
POST / HTTP/1.1
Host: thpc.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: AddQueue
<公共请求参数>

{
"ClusterId": "hpc-prkxbd7c",
```

```
"QueueName": "gpu"
}
```

输出示例

```
{
"Response": {
"RequestId": "13ee84a5-7846-4682-8877-1c8f94962dd6"
}
}
```

5. 开发者资源

腾讯云 API 平台

腾讯云 API 平台 是综合 API 文档、错误码、API Explorer 及 SDK 等资源的统一查询平台，方便您从同一入口查询及使用腾讯云提供的所有 API 服务。

API Inspector

用户可通过 API Inspector 查看控制台每一步操作关联的 API 调用情况，并自动生成各语言版本的 API 代码，也可前往 API Explorer 进行在线调试。

SDK

云 API 3.0 提供了配套的开发工具集（SDK），支持多种编程语言，能更方便的调用 API。

- Tencent Cloud SDK 3.0 for Python: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Java: GitHub, Gitee
- Tencent Cloud SDK 3.0 for PHP: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Go: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Node.js: GitHub, Gitee
- Tencent Cloud SDK 3.0 for .NET: GitHub, Gitee
- Tencent Cloud SDK 3.0 for C++: GitHub, Gitee
- Tencent Cloud SDK 3.0 for Ruby: GitHub, Gitee

命令行工具

- Tencent Cloud CLI 3.0

6. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码。

错误码	描述
AuthFailure	CAM签名/鉴权错误。
FailedOperation	操作失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooSmall	参数值过小。
InvalidParameterValue.ValueDuplicated	参数值重复。不支持此操作。
LimitExceeded	超过配额限制。
LimitExceeded.QueueNumLimit	队列数量达到上限。
OperationDenied	操作被拒绝。
RequestLimitExceeded	请求的次数超过了频率限制。

错误码	描述
ResourceNotFound	资源不存在。
ResourceNotFound.ClusterId	集群不存在。
UnsupportedOperation	操作不支持。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。

数据结构

最近更新时间：2025-05-14 02:04:21

CFSOption

描述CFS文件系统版本和挂载信息

被如下接口引用：AddClusterStorageOption, CreateCluster。

名称	类型	必选	描述
LocalPath	String	是	文件系统本地挂载路径。 示例值：/data/
RemotePath	String	是	文件系统远程挂载ip及路径。 示例值：172.0.0.1:/
Protocol	String	否	文件系统协议类型，默认值NFS 3.0。 • NFS 3.0。 • NFS 4.0。 • TURBO。 示例值：NFS 3.0
StorageType	String	否	文件系统存储类型，默认值SD；其中 SD 为通用标准型标准型存储， HP为通用性能型存储， TB为turbo标准型， TP 为turbo性能型。 示例值：SD

CFSOptionOverview

CFS存储选项概览信息。

被如下接口引用：DescribeClusterStorageOption。

名称	类型	描述
LocalPath	String	文件系统本地挂载路径。 示例值：/data/
RemotePath	String	文件系统远程挂载ip及路径。 示例值：172.0.0.1:/
Protocol	String	文件系统协议类型。 • NFS 3.0。 • NFS 4.0。 • TURBO。 示例值：NFS 3.0
StorageType	String	文件系统存储类型，默认值SD；其中 SD 为通用标准型标准型存储， HP为通用性能型存储， TB为turbo标准型， TP 为turbo性能型。 示例值：SD

ClusterActivity

符合条件的集群活动信息。

被如下接口引用：DescribeClusterActivities。

名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-myd8fgsc
ActivityId	String	集群活动ID。 示例值：cha-gvzj0zbd
ActivityType	String	集群活动类型。取值范围： • CreateAndAddNodes：创建实例并添加进集群

名称	类型	描述
		• RemoveNodesFromCluster：从集群移除实例 • TerminateNodes：销毁实例 • MountStorageOption：增加挂载选项并进行挂载 • UmountStorageOption：删除集群挂载存储选项并解挂载 示例值：TerminateNodes
ActivityStatus	String	集群活动状态。取值范围： • PENDING：等待运行 • RUNNING：运行中 • SUCCESSFUL：活动成功 • PARTIALLY_SUCCESSFUL：活动部分成功 • FAILED：活动失败 示例值：SUCCESSFUL
ActivityStatusCode	String	集群活动状态码。 示例值：ActivitySuccess
ResultDetail	String	集群活动结果详情。 注意：此字段可能返回 null，表示取不到有效值。 示例值：Activity success.
Cause	String	集群活动起因。 示例值：DeleteCluster
Description	String	集群活动描述。 示例值：删除指定集群，销毁实例，销毁所有节点。
RelatedNodeActivitySet	Array of NodeActivity	集群活动相关节点活动集合。 示例值：[{"NodeInstanceId":"ins-1zll2of0","NodeActivityStatus":"SUCCESSFUL"}]
StartTime	Timestamp ISO8601	集群活动开始时间。 示例值：2021-11-01T02:17:20Z
EndTime	Timestamp ISO8601	集群活动结束时间。 示例值：2021-11-01T02:17:38Z

ClusterOverview

集群概览信息。

被如下接口引用：DescribeClusters。

名称	类型	描述
ClusterId	String	集群ID。 示例值：hpc-2mv1i3d8
ClusterStatus	String	集群状态。取值范围： • PENDING：创建中 • INITING：初始化中 • INIT_FAILED：初始化失败 • RUNNING：运行中 • TERMINATING：销毁中

名称	类型	描述
		示例值：RUNNING
ClusterName	String	集群名称。 示例值：hpc-test
Placement	Placement	集群位置信息。 示例值：{ "Zone": "ap-guangzhou-6" }
CreateTime	Timestamp ISO8601	集群创建时间。 示例值：2024-12-18T13:46:34Z
SchedulerType	String	集群调度器。 示例值：SLURM
ComputeNodeCount	Integer	计算节点数量。 示例值：1
ComputeNodeSet	Array of ComputeNodeOverview	计算节点概览。 示例值：[{"NodeId":"ins-riredadfj6"},{"NodeId":"ins-mdafdsai"},{"NodeId":"ins-5icofadac"},{"NodeId":"ins-icfdadfy"}]
ManagerNodeCount	Integer	管控节点数量。 示例值：1
ManagerNodeSet	Array of ManagerNodeOverview	管控节点概览。 示例值：[{"NodeId":"node-rirfdafa"}]
LoginNodeSet	Array of LoginNodeOverview	登录节点概览。 示例值：[{"NodeId":"ins-riredadfj6"},{"NodeId":"ins-mdafdsai"},{"NodeId":"ins-5icofadac"},{"NodeId":"ins-icfdadfy"}]
LoginNodeCount	Integer	登录节点数量。 示例值：1
VpcId	String	集群所属私有网络ID。 示例值：vpc-r9fr2jzy

ComputeNode

计算节点信息。

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
SystemDisk	SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值：

名称	类型	必选	描述
			{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
InternetAccessible	InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}
InstanceName	String	否	节点显示名称。 • 不指定节点显示名称则默认显示‘未命名’。 最多支持60个字符。 示例值：未命名

ComputeNodeOverview

计算节点概览。

被如下接口引用：DescribeClusters。

名称	类型	描述
NodeId	String	计算节点ID。 示例值：ins-jfhc307q

DataDisk

描述了数据盘的信息

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
DiskSize	Integer	是	数据盘大小，单位：GB。最小调整步长为10G，不同数据盘类型取值范围不同，具体限制详见：存储概述。默认值为0，表示不购买数据盘。更多限制详见产品文档。 示例值：0
DiskType	String	否	数据盘类型。数据盘类型限制详见存储概述。取值范围： • LOCAL_BASIC：本地硬盘 • LOCAL_SSD：本地SSD硬盘 • LOCAL_NVME：本地NVME硬盘，与InstanceType强相关，不支持指定 • LOCAL_PRO：本地HDD硬盘，与InstanceType强相关，不支持指定 • CLOUD_BASIC：普通云硬盘 • CLOUD_PREMIUM：高性能云硬盘 • CLOUD_SSD：SSD云硬盘 • CLOUD_HSSD：增强型SSD云硬盘 • CLOUD_TSSD：极速型SSD云硬盘 默认取值：LOCAL_BASIC。 示例值：LOCAL_BASIC

ExpansionNodeConfig

弹性扩容节点配置信息。

被如下接口引用：SetAutoScalingConfiguration。

名称	类型	必选	描述
Placement	Placement	是	扩容实例所在的位置。 示例值：{"Zone": "ap-guangzhou-1"}
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
VirtualPrivateCloud	VirtualPrivateCloud	否	私有网络相关信息配置。 示例值：{"VpcId": "vpc-xxx", "SubnetId": "subnet-xxx"}

ExpansionNodeConfigOverview

扩容节点配置信息概览。

被如下接口引用：DescribeAutoScalingConfiguration。

名称	类型	描述
InstanceType	String	节点机型。 注意：此字段可能返回 null，表示取不到有效值。 示例值：S2.SMALL2
Placement	Placement	扩容实例所在的位置。 注意：此字段可能返回 null，表示取不到有效值。
InstanceChargeType	String	节点计费类型。 注意：此字段可能返回 null，表示取不到有效值。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 注意：此字段可能返回 null，表示取不到有效值。
VirtualPrivateCloud	VirtualPrivateCloud	私有网络相关信息配置。 注意：此字段可能返回 null，表示取不到有效值。
ImageId	String	指定有效的镜像ID，格式形如`img-xxx`。 注意：此字段可能返回 null，表示取不到有效值。 示例值：img-3la7wgnt
InternetAccessible	InternetAccessible	公网带宽相关信息设置。 注意：此字段可能返回 null，表示取不到有效值。
SystemDisk	SystemDisk	节点系统盘配置信息。 注意：此字段可能返回 null，表示取不到有效值。
DataDisks	Array of DataDisk	节点数据盘配置信息。 注意：此字段可能返回 null，表示取不到有效值。

Filter

描述键值对过滤器，用于条件过滤查询。例如过滤ID、名称、状态等

- 若存在多个Filter时，Filter间的关系为逻辑与（AND）关系。
- 若同一个Filter存在多个Values，同一Filter下Values间的关系为逻辑或（OR）关系。

被如下接口引用：DescribeNodes。

名称	类型	必选	描述
Name	String	是	需要过滤的字段。 示例值：fiter-name
Values	Array of String	是	字段的过滤值。 示例值：fiter-value

GooseFSOption

描述GooseFS挂载信息

被如下接口引用：AddClusterStorageOption, CreateCluster。

名称	类型	必选	描述
LocalPath	String	是	文件系统本地挂载路径。 示例值：/data
RemotePath	String	是	文件系统远程挂载路径。 示例值：/data/goosefs-fuse
Masters	Array of String	是	文件系统master的ip和端口。 示例值：["172.16.0.2:55533","172.16.0.3:55533","172.16.0.4:55533"]

GooseFSOptionOverview

GooseFS存储选项概览信息。

被如下接口引用：DescribeClusterStorageOption。

名称	类型	描述
LocalPath	String	文件系统本地挂载路径。 示例值：/data/
RemotePath	String	文件系统远程挂载路径。 示例值：/data/goosefs-fuse/
Masters	Array of String	文件系统master的ip和端口。 示例值：["172.16.0.2:55533","172.16.0.3:55533","172.16.0.4:55533"]

InstanceChargePrepaid

描述了实例的计费模式

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
Period	Integer	是	购买实例的时长，单位：月。取值范围：1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 36, 48, 60。 示例值：1
RenewFlag	String	否	自动续费标识。取值范围： NOTIFY_AND_AUTO_RENEW：通知过期且自动续费 NOTIFY_AND_MANUAL_RENEW：通知过期不自动续费 DISABLE_NOTIFY_AND_MANUAL_RENEW：不通知过期不自动续费

名称	类型	必选	描述
			默认取值：NOTIFY_AND_MANUAL_RENEW。若该参数指定为NOTIFY_AND_AUTO_RENEW，在账户余额充足的情况下，实例到期后将按月自动续费。 示例值：NOTIFY_AND_AUTO_RENEW

InternetAccessible

描述了实例的公网可访问性，声明了实例的公网使用计费模式，最大带宽等

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
InternetChargeType	String	否	网络计费类型。取值范围： BANDWIDTH_PREPAID：预付费按带宽结算 TRAFFIC_POSTPAID_BY_HOUR：流量按小时后付费 BANDWIDTH_POSTPAID_BY_HOUR：带宽按小时后付费 BANDWIDTH_PACKAGE：带宽包用户 默认取值：非带宽包用户默认与子机付费类型保持一致。 示例值：TRAFFIC_POSTPAID_BY_HOUR
InternetMaxBandwidthOut	Integer	否	公网出带宽上限，单位：Mbps。默认值：0Mbps。不同机型带宽上限范围不一致，具体限制详见购买网络带宽。 示例值：0

LoginNode

登录节点信息。

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
SystemDisk	Array of SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值： {"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
InternetAccessible	Array of InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}
InstanceName	String	否	节点显示名称。

名称	类型	必选	描述
			• 不指定节点显示名称则默认显示‘未命名’。最多支持60个字符。 示例值：未命名

LoginNodeOverview

登录节点概览。

被如下接口引用：DescribeClusters。

名称	类型	描述
NodeId	String	登录节点ID。 示例值：ins-jfhc307q

LoginSettings

描述了实例登录相关配置与信息。

被如下接口引用：AddNodes, CreateCluster。

名称	类型	必选	描述
Password	String	否	实例登录密码。不同操作系统类型密码复杂度限制不一样，具体如下： • Linux实例密码必须8到30位，至少包括两项[a-z]，[A-Z]、[0-9] 和 [( ) ` ~ ! @ # $ % ^ & * - + = | { } [ ] : ; ' , . ? / ]中的特殊符号。  • Windows实例密码必须12到30位，至少包括三项[a-z]，[A-Z]，[0-9] 和 [ ( ) ` ~ ! @ # $ % ^ & * - + = | { } [ ] : ; ' , . ? /] 中的特殊符号。 若不指定该参数，则由系统随机生成密码，并通过站内信方式通知到用户。 示例值：test@1253456

ManagerNode

管控节点信息

被如下接口引用：CreateCluster。

名称	类型	必选	描述
InstanceChargeType	String	否	节点计费类型。 • PREPAID：预付费，即包年包月 • POSTPAID_BY_HOUR：按小时后付费 • SPOTPAID：竞价付费 默认值：POSTPAID_BY_HOUR。 示例值：POSTPAID_BY_HOUR
InstanceChargePrepaid	InstanceChargePrepaid	否	预付费模式，即包年包月相关参数设置。通过该参数可以指定包年包月节点的购买时长、是否设置自动续费等属性。若指定节点的付费模式为预付费则该参数必传。 示例值：{"Period":1,"RenewFlag":"NOTIFY_AND_AUTO_RENEW"}
InstanceType	String	否	节点机型。不同实例机型指定了不同的资源规格。 • 具体取值可通过调用接口DescribeInstanceTypeConfigs来获得最新的规格表或参见实例规格描述。 示例值：S2.SMALL2
SystemDisk	SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值： {"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}

名称	类型	必选	描述
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值： [{"DiskSize":50,"DiskType":"CLOUD_PREMIUM","DiskBackupQuota":1}]
InternetAccessible	InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{"PublicIpAssigned":true,"InternetMaxBandwidthOut":5}
InstanceName	String	否	节点显示名称。 • 不指定节点显示名称则默认显示‘未命名’。 • 购买多个节点，如果指定模式串{R:x}，表示生成数字[x, x+n-1]，其中n表示购买节点的数量，例如server_{R:3}，购买1个时，节点显示名称为server_3；购买2个时，节点显示名称分别为server_3，server_4。支持指定多个模式串{R:x}。购买多个节点，如果不指定模式串，则在节点显示名称添加后缀1、2...n，其中n表示购买节点的数量，例如server_，购买2个时，节点显示名称分别为server_1，server_2。 • 最多支持60个字符（包含模式串）。 示例值：未命名

ManagerNodeOverview

管控节点概览。

被如下接口引用：DescribeClusters。

名称	类型	描述
NodeId	String	管控节点ID。 示例值：ins-jfhc307q

NodeActivity

节点活动信息。

被如下接口引用：DescribeClusterActivities。

名称	类型	描述
NodeInstanceId	String	节点活动所在的实例ID。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ins-1zll2of0
NodeActivityStatus	String	节点活动状态。取值范围： • RUNNING：运行中 • SUCCESSFUL：活动成功 • FAILED：活动失败 示例值：SUCCESSFUL
NodeActivityStatusCode	String	节点活动状态码。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ActivitySuccess
NodeActivityStatusReason	String	节点活动状态原因。 注意：此字段可能返回 null，表示取不到有效值。 示例值：Activity success.

NodeOverview

节点概览信息。

被如下接口引用：DescribeNodes。

名称	类型	描述
InstanceId	String	节点实例ID。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ins-f5ew7k5w
Zone	String	节点所在可用区信息。 注意：此字段可能返回 null，表示取不到有效值。 示例值：ap-chongqing-1
NodeState	String	节点状态。 • SUBMITTED：已完成提交。 • CREATING：创建中。 • CREATED：完成创建。 • INITING：初始化中。 • INIT_FAILED：初始化失败。 • RUNNING：运行中。 • DELETING：销毁中。 注意：此字段可能返回 null，表示取不到有效值。 示例值：RUNNING
ImageId	String	镜像ID。 注意：此字段可能返回 null，表示取不到有效值。 示例值：img-l8og963d
QueueName	String	节点所属队列名称。 注意：此字段可能返回 null，表示取不到有效值。 示例值：compute
NodeRole	String	节点角色。 • Manager：管控节点。 • Compute：计算节点。 • Login：登录节点。 • ManagerBackup：备用管控节点。 注意：此字段可能返回 null，表示取不到有效值。 示例值：Compute
NodeType	String	节点类型。 • STATIC：静态节点。 • DYNAMIC：弹性节点。 注意：此字段可能返回 null，表示取不到有效值。 示例值：STATIC

Placement

描述了实例的抽象位置

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, DescribeClusters, SetAutoScalingConfiguration。

名称	类型	必选	描述
Zone	String	是	实例所属的可用区名称。该参数可以通过调用 DescribeZones 的返回值中的Zone字段来获取。 示例值：ap-guangzhou-1

QueueConfig

扩容队列配置。

被如下接口引用：SetAutoScalingConfiguration。

名称	类型	必选	描述
QueueName	String	是	队列名称。 示例值：compute
MinSize	Integer	否	队列中弹性节点数量最小值。取值范围0～200。 示例值：0

名称	类型	必选	描述
MaxSize	Integer	否	队列中弹性节点数量最大值。取值范围0~200。 示例值：10
EnableAutoExpansion	Boolean	否	是否开启自动扩容。 示例值：False
EnableAutoShrink	Boolean	否	是否开启自动缩容。 示例值：False
ImageId	String	否	指定有效的镜像ID，格式形如img-xxx。目前仅支持公有镜和特定自定义镜像。 示例值：ins-jfhc307q
SystemDisk	SystemDisk	否	节点系统盘配置信息。若不指定该参数，则按照系统默认值进行分配。 示例值：{ "DiskSize": 50 }
DataDisks	Array of DataDisk	否	节点数据盘配置信息。若不指定该参数，则默认不购买数据盘。支持购买的时候指定21块数据盘，其中最多包含1块LOCAL_BASIC数据盘或者LOCAL_SSD数据盘，最多包含20块CLOUD_BASIC数据盘、CLOUD_PREMIUM数据盘或者CLOUD_SSD数据盘。 示例值：[{ "DiskSize": 50, "DiskType": "CLOUD_PREMIUM" }]
InternetAccessible	InternetAccessible	否	公网带宽相关信息设置。若不指定该参数，则默认公网带宽为0Mbps。 示例值：{ "PublicIpAssigned": "false" }
ExpansionNodeConfigs	Array of ExpansionNodeConfig	否	扩容节点配置信息。 示例值：{"Placement": "ap-guangzhou-6"}

QueueConfigOverview

扩容队列配置概览。

被如下接口引用：DescribeAutoScalingConfiguration。

名称	类型	描述
QueueName	String	队列名称。 示例值：compute
MinSize	Integer	队列中弹性节点数量最小值。取值范围0~200。 示例值：1
MaxSize	Integer	队列中弹性节点数量最大值。取值范围0~200。 示例值：5
EnableAutoExpansion	Boolean	是否开启自动扩容。 示例值：False
EnableAutoShrink	Boolean	是否开启自动缩容。 示例值：False
ExpansionNodeConfigs	Array of ExpansionNodeConfigOverview	扩容节点配置信息。 示例值： [{"InstanceType":"M5.SMALL8","InstanceChargeType":"POSTPAID_BY_HOU

QueueOverview

队列信息概览。

被如下接口引用：DescribeQueues。

名称	类型	描述
QueueName	String	队列名称。 示例值：compute

StorageOption

描述集群文件系统选项

被如下接口引用：AddClusterStorageOption, CreateCluster。

名称	类型	必选	描述
CFSOptions	Array of CFSOption	否	集群挂载CFS文件系统选项。
GooseFSOptions	Array of GooseFSOption	否	集群挂载GooseFS文件系统选项。

StorageOptionOverview

集群存储选项概览信息。

被如下接口引用：DescribeClusterStorageOption。

名称	类型	描述
CFSOptions	Array of CFSOptionOverview	CFS存储选项概览信息列表。
GooseFSOptions	Array of GooseFSOptionOverview	GooseFS存储选项概览信息列表。

SystemDisk

描述了操作系统所在块设备即系统盘的信息

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
DiskType	String	否	系统盘类型。系统盘类型限制详见存储概述。取值范围： LOCAL_BASIC：本地硬盘 LOCAL_SSD：本地SSD硬盘 CLOUD_BASIC：普通云硬盘 CLOUD_SSD：SSD云硬盘 CLOUD_PREMIUM：高性能云硬盘 默认取值：当前有库存的硬盘类型。 示例值：CLOUD_BASIC
DiskSize	Integer	否	系统盘大小，单位：GB。默认值为 50 示例值：50

Tag

标签键值对。

被如下接口引用：CreateCluster。

名称	类型	必选	描述
Key	String	是	标签键 示例值：env
Value	String	是	标签值 示例值：product

VirtualPrivateCloud

描述了VPC相关信息

被如下接口引用：AddNodes, CreateCluster, DescribeAutoScalingConfiguration, SetAutoScalingConfiguration。

名称	类型	必选	描述
VpcId	String	是	私有网络ID，形如vpc-xxx。有效的VpcId可通过登录控制台查询；也可以调用接口 DescribeVpcEx ，从接口返回中的unVpcId字段获取。若在创建子机时VpcId与SubnetId同时传入DEFAULT，则强制使用默认vpc网络。

名称	类型	必选	描述
			示例值：vpc-fjdkanf
SubnetId	String	是	私有网络子网ID，形如subnet-xxx。有效的私有网络子网ID可通过登录控制台查询；也可以调用接口 DescribeSubnets ，从接口返回中的unSubnetId字段获取。若在创建子机时SubnetId与VpcId同时传入DEFAULT，则强制使用默认vpc网络。 示例值：subnet-jkfndae

错误码

最近更新时间：2024-12-20 01:58:02

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

Error 中的 Code 表示错误码，Message 表示该错误的具体信息。

错误码列表

公共错误码

错误码	说明
ActionOffline	接口已下线。
AuthFailure.InvalidAuthorization	请求头部的 `Authorization` 不符合腾讯云标准。
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。
AuthFailure.MFAFailure	MFA 错误。
AuthFailure.SecretIdNotFound	密钥不存在。请在 控制台 检查密钥是否已被删除或者禁用，如状态正常，请检查密钥是否填写正确，注意前后不得有空格。
AuthFailure.SignatureExpire	签名过期。Timestamp 和服务器时间相差不得超过五分钟，请检查本地时间是否和标准时间同步。
AuthFailure.SignatureFailure	签名错误。签名计算错误，请对照调用方式中的签名方法文档检查签名计算过程。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未授权。请参考 CAM 文档对鉴权的说明。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误（包括参数格式、类型等错误）。
InvalidParameterValue	参数取值错误。
InvalidRequest	请求 body 的 multipart 格式错误。
IpInBlacklist	IP 地址在黑名单中。
IpNotInWhitelist	IP 地址不在白名单中。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数。
NoSuchProduct	产品不存在

错误码	说明
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
RequestLimitExceeded.GlobalRegionUinLimitExceeded	主账号超过频率限制。
RequestLimitExceeded.IPLimitExceeded	IP 限频。
RequestLimitExceeded.UinLimitExceeded	主账号限频。
RequestSizeLimitExceeded	请求包超过限制大小。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
ResponseSizeLimitExceeded	返回包超过限制大小。
ServiceUnavailable	当前服务暂时不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误，用户多传未定义的参数会导致错误。
UnsupportedOperation	操作不支持。
UnsupportedProtocol	http(s) 请求协议错误，只支持 GET 和 POST 请求。
UnsupportedRegion	接口不支持所传地域。

业务错误码

错误码	说明
AuthFailure	CAM签名/鉴权错误。
InternalError.CallCAM	CAM服务调用失败。
InternalError.CallCvm	cvm调用失败。
InvalidParameter.Malformed	参数格式有误。
InvalidParameterValue.InvalidFilterNotSupportedName	不支持指定过滤器的键。
InvalidParameterValue.LimitExceeded	参数值数量超过限制。
InvalidParameterValue.NotSupported	不支持该参数值。
InvalidParameterValue.ParametersNotSupported	字段不支持此值。
InvalidParameterValue.TooLarge	参数值过大。
InvalidParameterValue.TooLong	参数长度过长。
InvalidParameterValue.TooShort	无效参数值。参数值太短。
InvalidParameterValue.TooSmall	参数值过小。
InvalidParameterValue.ValueDuplicated	参数值重复。不支持此操作。
LimitExceeded.QueueNumLimit	队列数量达到上限。
OperationDenied	操作被拒绝。
ResourceNotFound.AutoScalingGroupId	无法找到弹性伸缩组ID。
ResourceNotFound.ClusterId	集群不存在。

错误码	说明
ResourceNotFound.ImageId	无法找到镜像ID。
ResourceNotFound.LaunchConfigurationId	无法找到ID对应的弹性伸缩启动配置。
ResourceNotFound.NodeId	无法找到ID对应节点。
ResourceNotFound.Queue	无法找到指定队列。
UnsupportedOperation.AutoScalingGroupAlreadyBinded	该伸缩组已绑定集群，请更换伸缩组。
UnsupportedOperation.AutoScalingType	弹性伸缩类型不支持此操作。
UnsupportedOperation.ClusterStatusNotSupport	该集群当前状态不支持该操作。
UnsupportedOperation.InvalidNodeRole	类型节点不支持当前操作。
UnsupportedOperation.NodeStatusNotSupport	节点状态不支持此操作。
UnsupportedOperation.ParameterTooLarge	参数值过大，不支持此操作。
UnsupportedOperation.ParameterTooSmall	参数值过小，不支持此操作。
UnsupportedOperation.QueueNotEmpty	队列内存在节点，不支持此操作。
UnsupportedOperation.VpcIdConflict	vpc冲突，不支持当前操作。