容器服务（TKE）

产品文档



目录

容器服务（TKE） 6
- 产品简介 6
  - 产品概述 6
  - 产品优势 8
  - 产品架构 10
  - 应用场景 12
  - 产品功能 14
- 快速入门 16
  - 入门必读 16
  - 部署容器服务TKE 18
  - 入门示例 23
    - 创建简单的nginx服务 23
    - 编写HelloWorld程序 26
    - 单实例版WordPress 30
    - 使用CDB的wordpress 32
- 操作指南 37
  - 集群管理 37
    - 集群概述 37
    - 集群的托管模式说明 38
    - 集群生命周期 40
    - 创建集群 41
    - 集群扩缩容 44
    - 连接集群 45
    - 升级集群 50
    - 集群启用IPVS 55
    - 集群启用GPU调度 56
    - 选择集群规格 59
    - CVM容器集群网络 61
    - 设置同地域集群间互通 62
    - 设置跨地域集群间互通 65
    - 设置容器集群与IDC互通 68
    - 设置安全组服务 71
    - 自定义 Kubernetes 组件启动参数 75
  - 镜像仓库 76
    - 镜像仓库概述 76
    - 镜像仓库基本教程 77
    - 如何搭建私有镜像仓库 81
  - 运维中心 89
    - 日志采集 89
    - 采集容器日志到对应消费端 93
  - 节点管理 94
    - 节点概述 94
    - 节点生命周期 95
    - 新增节点 96
    - 移除节点 98
    - 驱逐或封锁节点 99
    - 设置节点的启动脚本 102
    - 设置节点Label 105
    - 创建伸缩组 108
    - 查看伸缩组 110
    - 调整伸缩组 111
    - 删除伸缩组 113
    - 查看伸缩组伸缩记录 114
  - Kubernetes对象管理 116
    - 概述 116
    - Namespaces 118
    - 工作负载 120
      - Deployment管理 120
      - StatefulSet管理 132
      - DaemonSet管理 138
      - Job管理 146
      - CronJob管理 151
      - 设置工作负载的资源限制 157
      - 设置工作负载的调度规则 160
      - 设置工作负载的健康检查 162
      - 设置工作负载的运行命令和参数 164
      - 使用 TCR 企业版实例内容器镜像创建工作负载 166
    - 自动伸缩 170
      - 自动伸缩基本操作 170
      - 自动伸缩指标说明 177
    - 服务 182
      - Service管理 182
      - Ingress管理 183
    - Service 管理 187
      - 概述 187
      - Service 基本功能 190
      - Service 负载均衡配置 194
      - Service 使用已有 CLB 200
      - Service 后端选择 203
      - Service 优雅停机 208
      - Pod 优雅删除 212
      - 使用 LoadBalancer 直连 Pod 模式 Service 214
      - 多 Service 复用 CLB 219
      - Service 扩展协议 221
      - Service Annotation 说明 224
    - 配置 227
      - ConfigMap管理 227
      - Secret管理 233
    - Ingress 管理 239

- Nginx 类型 Ingress ... 239
  - 概述 ... 239
  - 安装 Nginx-ingress 实例 ... 240
  - 使用 Nginx-ingress 对象接入集群外部流量 ... 241
- CLB 类型 Ingress ... 243
  - 概述 ... 243
  - Ingress 基本功能 ... 245
  - Ingress 使用已有 CLB ... 250
  - Ingress 证书配置 ... 252
  - Ingress Annotation 说明 ... 257
- Ingress Controllers 说明 ... 260
- 组件管理 ... 262
  - 扩展组件概述 ... 262
  - 扩展生命周期管理 ... 263
  - COS 说明 ... 264
  - CFS 说明 ... 265
  - CBS 说明 ... 267
    - CBS 简介 ... 267
    - 通过CBS避免云硬盘跨可用区挂载 ... 269
    - 在线扩容云硬盘 ... 271
    - 创建快照和使用快照来恢复卷 ... 274
  - P2P 说明 ... 279
  - GPU-Manager 说明 ... 282
  - HPC 说明 ... 285
  - Network Policy 说明 ... 289
  - Nginx-ingress 说明 ... 290
- 存储 ... 291
  - Volume管理 ... 291
  - PV&PVC管理 ... 294
  - StorageClass管理 ... 300
- 授权管理 ... 303
  - 概述 ... 303
  - 使用预设身份授权 ... 305
  - 自定义策略授权 ... 307
- 监控 ... 310
  - 监控概述 ... 310
  - 查看监控数据 ... 311
  - 监控及告警指标列表 ... 314
- HelmChart ... 319
  - 概述 ... 319
  - Helm实列 ... 320
  - Helm仓库 ... 322
- 健康检查 ... 325

- 最佳实践 ... 329
  - 构建简单web应用 ... 329
  - docker run 参数适配 ... 336
  - 单实例多容器实践 ... 339
- 访问管理 ... 340
  - 概述 ... 340
  - 服务授权相关角色权限说明 ... 341
  - TKE镜像仓库资源级权限设置 ... 342
  - TKE集群资源级权限接口列表 ... 349
  - 使用示例 ... 353
    - 配置子账号对TKE服务全读写或只读权限 ... 353
    - 配置子账号对单个TKE集群的管理权限 ... 355
- 购买容器集群 ... 360
  - 集群新增资源所属项目说明 ... 360
  - 容器服务安全组设置 ... 361
  - 容器服务节点公网IP说明 ... 364
  - 容器及节点网络设置 ... 366
  - 容器节点硬盘设置 ... 367
- 常见问题 ... 368
  - 集群类 ... 368
    - 集群相关 ... 368
    - 扩容缩容相关 ... 370
  - 服务常见问题 ... 373
  - 镜像仓库常见问题 ... 375
  - 远程终端常见问题 ... 376
  - 事件常见问题 ... 380
- 词汇表 ... 381
- API文档 ... 382
  - 镜像仓库服务（tcr） ... 382
    - 版本（2019-09-24） ... 382
      - API 概览 ... 382
      - 调用方式 ... 384
        - 接口签名v1 ... 384
        - 接口签名v3 ... 391
        - 请求结构 ... 400
        - 返回结果 ... 401
        - 公共参数 ... 404
      - 个人版相关接口 ... 406
        - 个人版镜像仓库批量删除Tag ... 406
        - 批量删除个人版仓库 ... 408
        - 创建应用更新触发器(功能已下线) ... 410
        - 创建个人版镜像版本清理策略 ... 412
        - 创建个人版命名空间 ... 414
        - 创建个人版镜像仓库 ... 416
        - 创建个人用户 ... 418
        - 删除个人版全局镜像版本自动清理策略 ... 420
        - 删除个人版镜像仓库Tag自动清理策略 ... 421
        - 删除个人版仓库tag ... 423

- 删除个人版命名空间 425
- 删除个人版镜像仓库 427
- 查询个人收藏仓库(功能下线中) 429
- 查询个人版中与指定tag镜像内容相同的tag列表 431
- 获取个人版全局镜像版本自动清理策略 433
- 获取个人版仓库自动清理策略 435
- 获取个人版镜像仓库tag列表 437
- 查询个人版命名空间信息 439
- 获取满足输入搜索条件的个人版镜像仓库 441
- 查询个人版所有仓库 443
- 查询个人版仓库信息 445
- 查询个人用户配额 447
- 复制个人版仓库镜像版本 449
- 设置个人版全局镜像版本自动清理策略 451
- 更新个人版仓库访问属性 453
- 更新个人版镜像仓库描述 455
- 修改个人用户登录密码 457
- 验证个人版命名空间是否存在 459
- 验证个人版仓库是否存在 461
- 数据结构 463
- 错误码 486

- 容器服务（tke） 489
  - 版本（2018-05-25） 489
    - API 概览 489
    - 调用方式 493
      - 接口签名v1 493
      - 接口签名v3 500
      - 请求结构 509
      - 返回结果 510
      - 公共参数 513
    - 伸缩组相关接口 515
      - 创建集群的伸缩组 515
      - 删除集群伸缩组 518
      - 查询集群层面的弹性伸缩配置 520
      - 查询集群关联的伸缩组列表 522
      - 修改集群伸缩组属性 524
      - 修改集群弹性伸缩组的属性 526
    - 其他接口 528
      - 获取镜像信息 528
      - 获取集群配额 530
      - 查询地域列表 532
      - 获取支持的集群版本信息 534
      - TKE APIServer代理转发接口 536
    - 监控相关接口 538
      - 关闭集群审计 538
      - 关闭集群事件存储 540
      - 启动集群审计 542
      - 启动集群审计 544
      - 获取集群的仪表盘ID 546
    - 网络相关接口 548
      - 添加容器子网（VPC-CNI网络类型的集群） 548
    - 节点相关接口 550
      - 添加已经存在的CVM实例到集群 550
      - 扩展集群节点 553
      - 删除集群中的节点 556
      - 获取集群节点列表 558
      - 查询集群节点信息 561
      - 查询已经存在的节点 564
      - 驱逐集群中的指定节点 568
    - 集群相关接口 570
      - 检查集群的CIDR是否冲突 570
      - 检查集群指定日志采集规则（旧版）是否已经存在 573
      - 采集当前TKE管控集群所使用的cpu总核数 575
      - 创建集群 577
      - 开启集群内网/外网访问 580
      - 添加虚拟节点至指定集群的指定虚拟节点池 582
      - 在指定集群内创建虚拟节点池 584
      - 创建集群日志收集规则(旧版) 586
      - 删除集群 588
      - 删除集群指定虚拟节点 590
      - 删除集群指定虚拟节点池 592
      - 删除Chart 594
      - 删除Chart指定版本 596
      - 删除日志收集规则(旧版) 598
      - 获取Chart的下载信息 600
      - 查询集群endpoints的信息 602
      - 查询集群内网/外网访问端口的状态 604
      - 获取集群的kubeconfig 606
      - 查询支持的集群等级 608
      - 查询集群的密钥信息 610
      - 获取集群Service列表 613
      - 获取集群指定虚拟节点池的节点列表 615
      - 获取集群的虚拟节点池列表 617
      - 查询集群列表 619
      - 获取chart上传cos信息 621
      - 获取chart列表 623
      - 获取chart详情 625
      - 获取chart版本列表 627
      - 驱逐集群指定虚拟节点 629
      - 启用指定集群的日志收集服务（旧版） 631
      - 获取集群指定日志收集规则的信息(旧版) 633

- 获取集群日志采集（旧版）启用状态 635
- 查询价格 637
- 查询集群日志收集列表(旧) 639
- 修改集群属性 641
- 修改虚拟节点池 644
- 查询当前TKE管控集群的cpu总核数是否超过限制 646
- 更新集群指定日志收集规则(旧版) 648
- 上传Chart 650
- 数据结构 652
- 错误码 689

产品简介
产品概述

产品介绍

容器服务平台是高度可扩展的高性能容器管理服务，您可以在托管的云服务器实例集群上轻松运行应用程序。使用该服务，您将无需安装、运维、扩展您的集群管理基础设施，只需进行简单的 API 调用，便可启动和停止 Docker 应用程序，查询集群的完整状态，以及使用各种云服务。您可以根据资源需求和可用性要求在集群中安排容器的置放，满足业务或应用程序的特定要求。
容器服务平台基于原生 Kubernetes 提供以容器为核心的解决方案，解决用户开发、测试及运维过程的环境问题、帮助用户降低成本，提高效率。容器服务平台完全兼容原生 Kubernetes API, 并扩展了 CBS、CLB 等 Kubernetes 插件，同时以私有网络为基础，实现了高可靠、高性能的网络方案。

名词解释

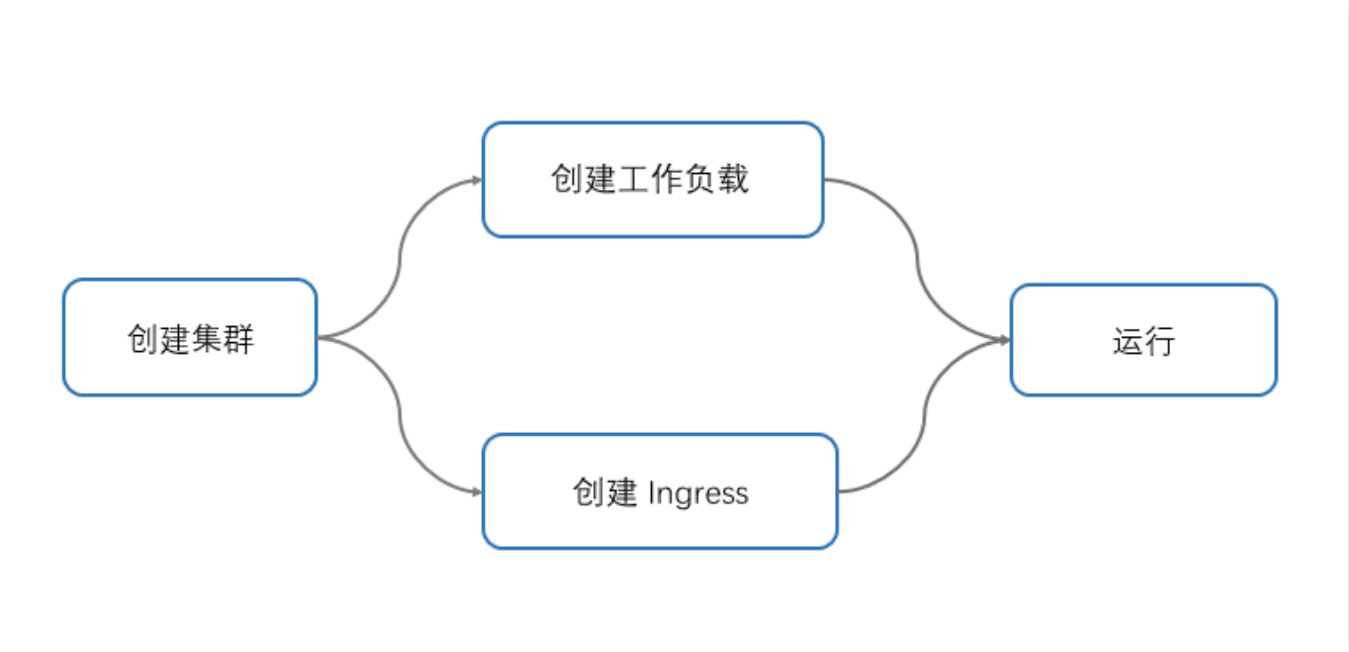
使用容器服务平台，会涉及到以下基本概念：

- 集群：是指容器运行所需云资源的集合，包含了若干台云服务器、负载均衡器等云资源。
- 实例（Pod）：由相关的一个或多个容器构成一个实例，这些容器共享相同的存储和网络空间。
- 工作负载：Kubernetes 资源对象，用于管理 Pod 副本的创建、调度以及整个生命周期的自动控制。
- Service：由多个相同配置的实例（Pod）和访问这些实例（Pod）的规则组成的微服务。
- Ingress：Ingress 是用于将外部 HTTP(S) 流量路由到服务（Service）的规则集合。
- Helm 应用：Helm 是管理 Kubernetes 应用程序的打包工具，提供了 Helm Chart 在指定集群内图形化的增删改查。
- 镜像仓库：用于存放 Docker 镜像，Docker 镜像用于部署容器服务。

使用流程

见下图，您只需要三步即可运行应用程序。

1. 创建集群
2. 创建工作负载或 创建 Ingress
3. 运行 Service 或 运行Ingress



相关服务

- 通过购买若干个云服务器组成容器服务集群，容器运行在云服务器中。有关更多信息，请参阅云服务器产品文档。
- 集群可以建立在私有网络下，集群内主机可以分配在不同可用区的子网下。有关更多信息，请参阅私有网络产品文档。
- 可以使用负载均衡，自动分配横跨多个云服务实例的客户端请求流量，转发至主机内容器。有关更多信息，请参阅负载均衡产品文档。
- 监控容器服务集群和容器实例的运行统计数据，可使用云监控。有关更多信息，请参阅云监控产品文档。

产品优势

编排优势

基于 Kubernetes

容器服务平台是基于 Kubernetes 实现的，Kubernetes(k8s) 是 Google 开源的容器集群管理系统。在 Docker 技术的基础上，为容器化的应用提供部署运行、资源调度、服务发现和动态伸缩等一系列完整功能，提高了大规模容器集群管理的便捷性。

Kubernetes 的优势

- Kubernetes 采用优雅的软件工程设计，通过模块化、微服务的方式，实现模块化设计，使得用户可以根据自己的使用场景；通过灵活插拔方式，采用自定义的网络、存储、调度、监控、日志等模块。
- Kubernetes 项目的社区秉承开源、开放的心态，可以支持容器、网络、存储实施方案。

容器服务平台TKE对比自建容器服务

优势	容器服务平台（TKE）	自建容器服务
简单易用	简化集群管理容器服务平台提供超大规模容器集群管理、资源调度、容器编排、代码构建，屏蔽了底层基础构架的差异，简化了分布式应用的管理和运维，您无需再操作集群管理软件或设计容错集群架构，因此也无需参与任何相关的管理或扩展工作。您只需启动容器集群，并指定想要运行的任务即可，容器服务平台帮您完成所有的集群管理工作，让您可以集中精力开发 Docker 化的应用程序。	自建容器管理基础设施通常涉及安装、操作、扩展自己的集群管理软件、配置管理系统和监控解决方案，管理复杂
灵活扩展	灵活集群托管，集成负载均衡您可以使用容器服务灵活安排长期运行的应用程序和批量作业。您还可以使用 API 获得最新的集群状态信息，以便集成您自己的自定义计划程序和第三方计划程序。容器服务平台与负载均衡集成，支持在多个容器之间分配流量。您只需指定容器配置和要使用的负载均衡器，容器服务管理程序将自动添加和删除。另外容器服务平台可以自动恢复运行状况不佳的容器，保证容器数量满足您的需求，以便为应用程序提供支持。	需要根据业务流量情况和健康情况人工确定容器服务的部署，可用性和可扩展性差
安全创新	资源高度隔离，服务高可用容器服务在您自己的云服务器实例中启动，不与其他客户共享计算资源。您的集群在私有网络中运行，因此您可以使用您自己的安全组和网络 ACL，这些功能为您提供了高隔离水平，并帮助您使用云服务器构建高安全水平的应用程序。容器服务采用分布式服务架构，保证服务的故障自动恢复、快速迁移；结合有状态服务后端的分布式存储，实现服务和数据的安全、高可用。	自建容器服务因其内核问题及 Namespace 不够完善，租户、设备、内核模块隔离性都比较差

优势	容器服务平台（TKE）	自建容器服务
高效	镜像快速部署，业务持续集成容器服务平台运行在您的私有网络中，高品质的 BGP 网络保证镜像极速上传下载，轻松支持海量容器秒级启动，极大程度降低了运行开销，使您的部署更加专注于业务运行。您可以在容器服务平台上部署业务，开发人员在 GitHub 或其他代码平台提交代码后，容器服务可立即进行构建、测试、打包集成，将集成的代码部署到预发布环境和现网环境上。	自建容器服务的网络无保证，因此无法保证使用镜像创建容器的效率
低成本	容器服务免费容器服务平台没有任何附加费用，您可以在容器中免费调用 API 构建您的集群管理程序。您只需为您创建的用于存储和运行应用程序的云服务资源（例如云服务器、云硬盘等）付费。	需要投入资金构建、安装、运维、扩展自己的集群管理基础设施，成本开销大

容器服务平台TKE监控与自建容器监控对比

容器服务平台监控为容器集群、服务、实例提供数据收集和数据展示功能。使用容器服务监控，您可以查看集群、节点、服务、实例、容器等近 30 个指标的监控统计数据，验证集群是否正常运行并创建相应告警，监控指标覆盖面广，并且在持续增加中。

优势	容器服务平台（TKE）	自建容器服务
指标完整	涉及到集群，服务，容器，pod 等近 30 个指标	指标不完整，很多需要开发
搭建成本低	创建集群时自带	人工搭建，成本高
运维成本低	平台助力运维，保证数据准确性	人工维护
存储成本低	每个指标免费保存 3 个月数据	根据存储大小计算
扩展性高	平台侧会不断完善和增加新的指标项	需要技术人员大量投入开发新指标
告警	有	无
问题排查手段	控制台可以方便查看容器 log， 并与 webshell 结合一键登录容器快速排查问题	需要手动登录到容器或者机器排查

产品架构

总体架构

本节介绍容器服务系统的设计和实现，产品架构如下图所示：



架构说明

1. 容器服务平台提供支持原生 Kubernetes 能力。
2. 提供了 Kubernetes 插件，帮助用户快速构建 Kubernetes 集群。
3. 容器服务平台在 Kubernetes 上层，提供了集群管理、应用管理、CI/CD 等进阶能力。

模块说明

1. 容器服务控制台和云 API：用户通过控制台、Kubectl 或 API 操作集群与服务。
2. 镜像服务 CCR 模块：TCloudFinanceZone提供的镜像服务模块，用户可以上传镜像或将镜像下载到本地。

3. 容器服务 TKE 模块：容器服务核心模块，包括集群的增删改查、服务的增删改查等。

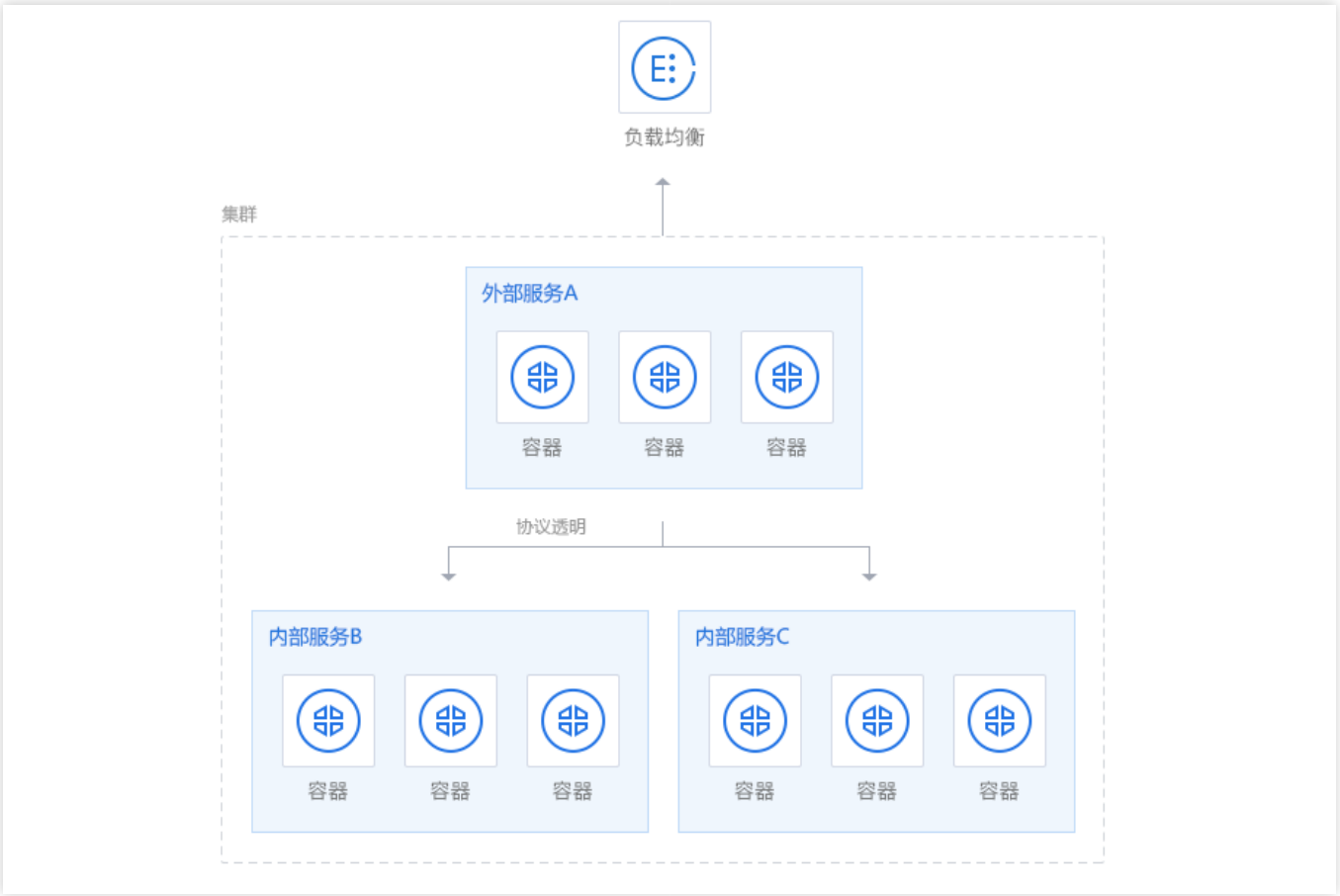
应用场景

微服务架构

微服务架构适用于构建复杂的应用，将单体式应用从不同纬度拆分成多个微服务，每个微服务的内容使用一个 docker 镜像管理。

容器服务平台部署微服务优势：

1. 简化了集群管理，无须安装、管理操作集群。
2. 无缝衔接了的计算、网络、存储、监控、安全能力，直接使用TCloudFinanceZone IAAS 能力。
3. 支持服务编排，服务粒度管理应用，简单易懂，资源高度隔离、服务高可用。



业务快速上云

个人或企业业务迁移到云平台上，可选择容器服务平台来简化上云配置，简化集群管理，提升业务交付速率。
容器服务平台让您一键单击创建服务，快速实现应用容器化部署，同时也可达到弹性扩缩、按需部署、高可用、易扩容、开发友好、降低人力成本的效果。


一键点击
Cloud
用户
容器服务
云

产品功能

产品功能

集群管理

通过容器服务平台可简单高效地管理您的容器集群，整个过程安全可信赖，并且能够无缝衔接计算、存储、网络能力。

模块	功能点
集群构成	支持 CVM 所有机型，可以新增和添加已有主机集群内主机支持跨可用区部署支持按量计费模式用户独占集群、VPC 安全隔离自定义集群网络，容器网络灵活配置
集群管理	- 支持节点升降配丰富的监控指标 - 支持自定义告警策略 - 自定义 Kubernetes 组件启动参数
Kubernetes 管理	- 支持 Kubernetes 多版本 - 版本升级功能Kubernetes 证书管理，Kubectl 直接操作集群控制台简单管理 Namespace - 资源对象浏览器，可以一键查看集群中所有资源对象 - 多种不同的扩展组件增强集群表现

应用管理

通过容器服务平台提供的应用管理功能，能够帮助您一键快速创建多个服务，部署不同环境应用。

模块	功能点
应用构成	- 支持 TKE 多种服务类型 - 支持 Kubernetes Deployment、DamentSet 等多种资源
应用管理	应用支持我的模板、模板市场快速创建支持更新应用实时对比查看应用内服务一键部署/停止
模板管理	支持我的模板

服务管理

服务管理为您提供高效的容器管理方案，支持服务的快速创建、快速扩缩容、负载均衡、服务发现、服务监控、健康检查等特性，您可以通过服务管理方便快捷的管理您的容器。

模块	功能点
服务部署	支持单实例多容器的服务部署支持多种服务访问方式支持服务内实例跨可用区部署支持设置亲和性和反亲和性调度

模块	功能点
服务管理	支持服务的滚动更新和快速更新支持服务的动态扩缩容支持远程登录到服务内容器
服务运维	支持查看服务详细的监控指标支持查看服务内容器的 stdout 和 stderr 日志支持设置服务告警策略支持设置存活检查和就绪检查两种健康检查方式容器异常自动恢复

配置项管理

配置项用来规定一些程序在启动时读入设定，提供了一种修改程序设置的方法， 针对不同的对象可以使用不同的配置项。

模块	功能点
配置项管理	配置项支持多版本支持可视化和 YAML 两种编辑形式
配置项使用	配置项以数据卷的形式挂载到容器目录配置项导入成环境变量配置项替代应用模板变量

镜像管理

镜像仓库包含了 dockerhub 官方镜像和用户私有镜像，镜像管理可以让您快速创建镜像、快速部署服务。

模块	功能点
镜像管理	支持创建私有镜像仓库支持查看和使用 DockerHub 镜像仓库（部署节点需要外网） 支持查看和使用 Hub 镜像仓库 支持管理多个镜像命名空间
镜像使用	提供高速的内网通道用于镜像创建服务支持公网上传下载镜像
CI/CD	支持设置私有镜像自动构建支持设置镜像的触发器

快速入门
入门必读

本文档旨在帮助您了解容器服务平台的基本知识，解答您在使用容器服务中可能遇到的问题，帮助您更快上手容器服务。

准备使用

我想在基础网络中使用 TKE，可以吗？
容器服务当前仅支持 VPC 网络，不支持基础网络。

简单试用

1. 我该如何使用容器服务？
   创建集群，创建服务两步即可使用，可以参考入门示例。
2. 是否可以选择已有的云主机加入集群？
   支持，可以创建集群完成后添加已有云主机。
3. 为什么我的服务一直在启动中？
   服务内容器若无持续运行的进程，会导致服务一直处于启动中, 更多服务启动的问题见事件常见问题。
4. 创建好的服务如何访问？
   不同的访问方式提供不同的访问入口，详情见服务访问方式。
5. 容器怎么访问公网？
   若容器所在主机有公网 IP 和带宽，则容器可直接访问外网，若容器所在主机无公网 IP 和带宽，则可以通过 NAT 网关访问外网。

部署业务

1. 我的业务需要配置很多文本或环境变量，该怎么管理？
   您可以通过 配置项 来管理配置文件。
2. 服务之间想互访该怎么办？
   集群下相同 Namespace 的服务可以直接相互访问，不同 Namespace 的服务需要通过 <service-name>.<namespace-name>.svc.cluster.local 的形式来访问
3. Ingress 与服务访问方式的“公网访问”有什么差异？
   Ingress 是将外部 HTTP(S) 流量路由到服务规则集合，与服务访问方式的公网访问无直接关系。
4. 我的业务是有状态的需要依赖于磁盘，可以吗？

可以通过挂载 CBS 数据卷的形式挂载数据盘到容器中。

5. 服务更新时业务会中断吗？

   服务有两种更新方式：滚动更新和快速更新。选择滚动更新方式，业务不会中断。

部署容器服务TKE

容器服务平台是高度可扩展的高性能容器管理服务，您可以在托管的云服务器实例集群上轻松运行应用程序。在本教程中，您将了解如何使用容器服务快速创建和管理容器集群，并在集群内快速、弹性地部署您的服务。

步骤 1：创建集群

首先您需要创建集群。集群是指容器运行所需云资源的集合，包含了若干台云服务器、负载均衡器等资源。

1. 登录TKE 控制台。
2. 单击左侧导航栏中的【集群】，在【集群管理】页单击【新建】。
3. 设置集群的基本信息后单击【下一步】。
    - 集群名称：您要创建的集群的名称。不超过60个字符。
    - CPU 架构：选择CPU架构。
    - Kubernetes 版本：选择 Kubernetes 版本。
    - 运行时组件：提供docker和containerd两种选择。
    - 集群 IP 类型：提供IPv4和IPv4/IPv6双栈。
    - 所在地域：建议您根据所在地理位置选择靠近的地域。可降低访问延迟，提高下载速度。
    - 集群网络：如现有的网络不合适，您可以去控制台 新建私有网络。
    - 容器网络插件：选择容器网络插件。
    - 容器网络：为集群内容器分配在容器网络地址范围内的 IP 地址，单节点 Pod 数量上限和集群内 Service 数量上限。
    - 操作系统：选择操作系统。
    - 集群描述：创建集群的相关信息。该信息将显示在 集群信息 页面。
    - 高级设置：
        - 可以开启“ipvs支持”：注意开启后将不支持关闭，适用于大规模场景下提供更优的转发性能。
        - 可以添加“标签”：为TKE集群配置标签，集群内创建的云服务的资源自动继承集群标签。
4. 选择机型（支持系统盘为云盘的所有机型）后单击【下一步】。
    - Node：
        - 立即部署: 直接在新建集群过程中新增Node节点
        - 暂不部署: 再集群新建完成之后再新增Node节点
    - 计费模式：按量计费。
    - Master&Etcd机型：
        - 可用区：选择当前服务可用的区域。
        - 节点网络：为集群内服务器分配在节点网络地址范围内的 IP 地址。参阅 容器及节点网络设置。
        - 机型：选择节点机型
        - 系统盘：SSD云硬盘固定为50GB，系统盘不支持更换介质，后端使用NVMe固态存储，适用于核心数据库业务、大型OLTP业务，以及对IO性能有极高的要求的业务
        - 数据盘：仅支持SSD云硬盘
        - 公网宽带：可选访不访问公网
        - 数量：选择节点数量， Master&Etcd机型至少选择三台。



- Node机型：
  和 Master&Etcd机型的选择完全一样。



5. 填写云主机配置后单击 【下一步】。
   - 数据盘挂载：将容器和镜像存储在数据盘，将容器和镜像存储在数据盘将自动格式化数据盘成ext4且自动挂载到您指定的挂载点，并将容器存储到挂载点的docker目录，仅对购买数据盘的节点生效
   - 安全组：安全组具有防火墙的功能，用于设置云服务器 CVM 的网络访问控制。
   - 登录方式：提供三种对应登录方式。
     - 设置密码：请根据提示设置对应密码。
     - 立即关联密钥：密钥对是通过一种算法生成的一对参数，是一种比常规密码更安全的登录云服务器的方式。详情参阅 SSH 密钥 。
     - 自动生成密码：自动生成的密码将通过站内信发送给您。
   - 安全加固：可选免费开通，安装组件免费开通DDoS防护、WAF和云镜主机防护
   - 云监控：可选免费开通，免费开通云产品监控、分析和实施告警，安装组件获取主机监控指标
   - 高级设置：
     - 自定义数据：用于启动时配置实例

- 封锁(cordon)：封锁节点后，将不接受新的Pod调度到该节点

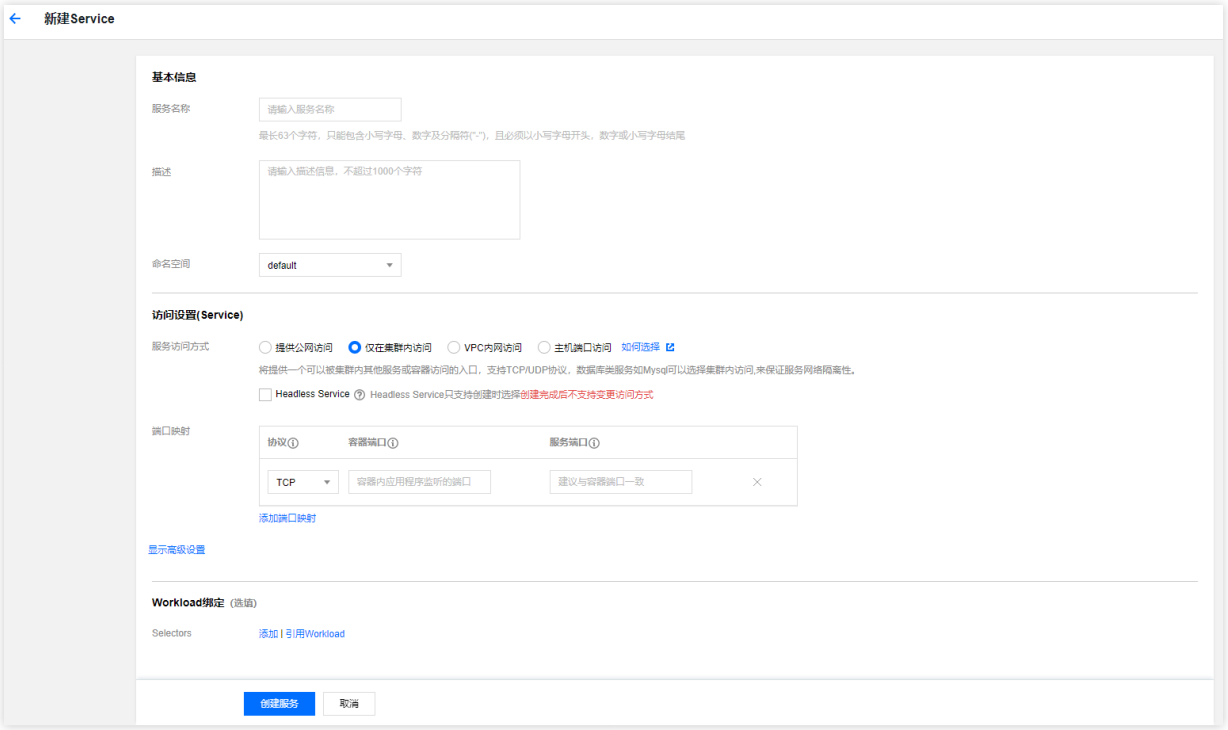
6. 确认配置信息后单击【完成】。
   创建完成的集群将出现在集群列表中。



步骤 2：创建服务

您现已创建了集群，接下来就是创建服务。服务是由多个相同配置的容器和访问这些容器的规则组成的微服务。

1. 单击集群列表中的一个集群，选择左侧导航栏中的【服务】 > 【Service】，在服务列表页的【新建】。
2. 基本信息设置。
   - 基本信息：
     - 服务名称：要创建的服务的名称。服务名称最长63个字符，只能包含小写字母、数字及分隔符(-)，且必须以小写字母开头，数字或小写字母结尾。
     - 描述：创建服务的相关信息。该信息将显示在 服务信息 页面。
     - 命名空间：选择服务创建的命名空间
   - 访问设置(Service)：
   - 服务访问方式：
     - 提供公网访问：自动创建传统型公网CLB以提供Internet访问入口，支持TCP/UDP协议，如web前台类服务可以选择公网访问。
     - Headless Service：该访问模式下可选，不创建用于集群内访问的ClusterIP，访问Service名称时返回后端Pods IP地址，用于适配自有的服务发现机制。解析域名时返回相应 Pod IP 而不是 Cluster IP
     - 仅在集群内访问：使用 Service 的 ClusterIP 模式，自动分配 Service 网段中的 IP，用于集群内访问。数据库类等服务如 MySQL 可以选择集群内访问，以保证服务网络隔离
     - VPC内访问：将提供一个可以被集群所在VPC下的其他资源访问的入口，支持TCP/UDP协议，需要被同一VPC下其他集群、云主机等访问的服务可以选择VPC内网访问的形式。
     - 主机端口访问：提供一个主机端口映射到容器的访问方式，支持 TCP、UDP、Ingress。可用于业务定制上层 LB 转发到 Node
     - 端口映射：输入负载要暴露的端口并指定通信协议类型
     - 高级设置：
       - Session Affinity: 会话保持，设置会话保持后，会根据请求IP把请求转发给这个IP之前访问过的Pod。默认None，按需使用
   - Workload绑定：
     - 添加：添加Workload的标签
     - 引用workload：直接引用Workload资源



3. 单击【创建服务】完成服务创建。
   创建完成的服务将出现在服务列表中。

步骤 3：删除资源

在本教程中，您启动了两种资源：集群和服务。在此步骤中，您将清除所有的资源以免产生不必要的费用。

删除集群

1. 单击左侧导航栏中的【集群】，在"集群管理"页单击操作列的【更多】>【删除】。

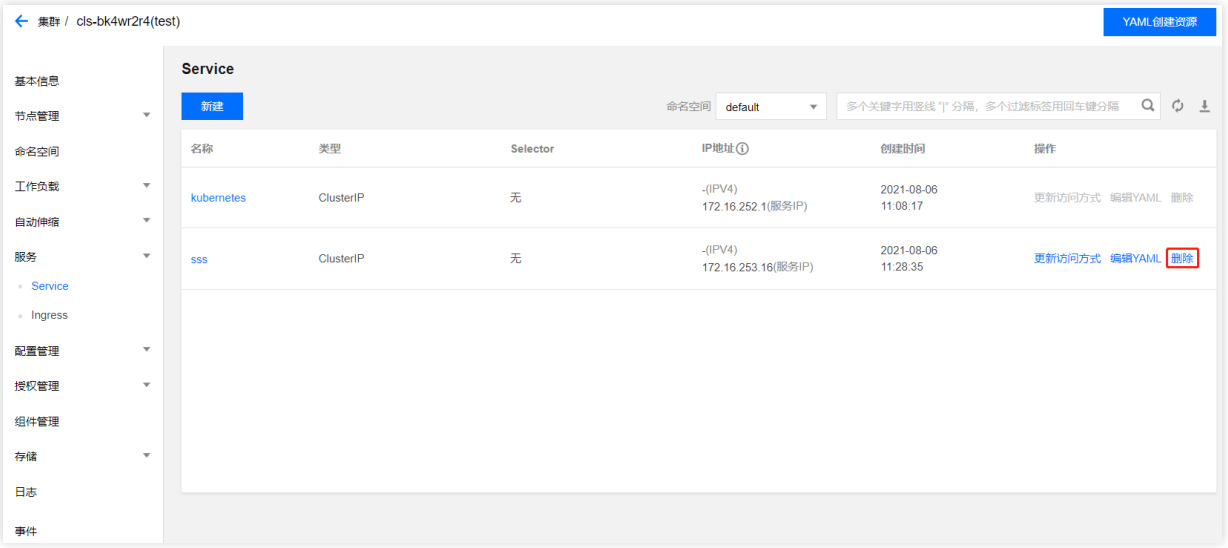


2. 单击【确定】。

> 注意：
> 集群在删除期间，无法对外提供服务，请提前做好准备，以免造成影响。

删除服务

1. 单击一个集群ID/名称进入一个集群，选择左侧导航栏中的【服务】 > 【Service】，单击服务列表右侧【删除】。



2. 单击【提交】。

更多

通过本教程，您已经了解如何在容器服务 TKE 中配置、部署和删除服务。使用容器服务 TKE，您将无需安装、运维、扩展您的集群管理基础设施，只需进行简单的 API 调用，便可启动和停止 Docker 应用程序，查询集群的完整状态，以及使用各种云服务。

入门示例
创建简单的nginx服务

本文档旨在帮助大家了解如何快速创建一个容器集群内的 Nginx 服务。

> 注意：
> 在创建 Nginx 服务之前，您必须拥有：
> 一个创建好的集群。有关如何创建集群的详细信息，参见创建集群。

创建 Nginx 服务

1. 登录 TKE 控制台。
2. 单击左侧导航栏中的【集群】，单击集群列表页的一个集群【ID/名称】。



3. 在集群页里面，选择【工作负载】 > 【Deployment】，在Deployment列表里面单击【新建】。



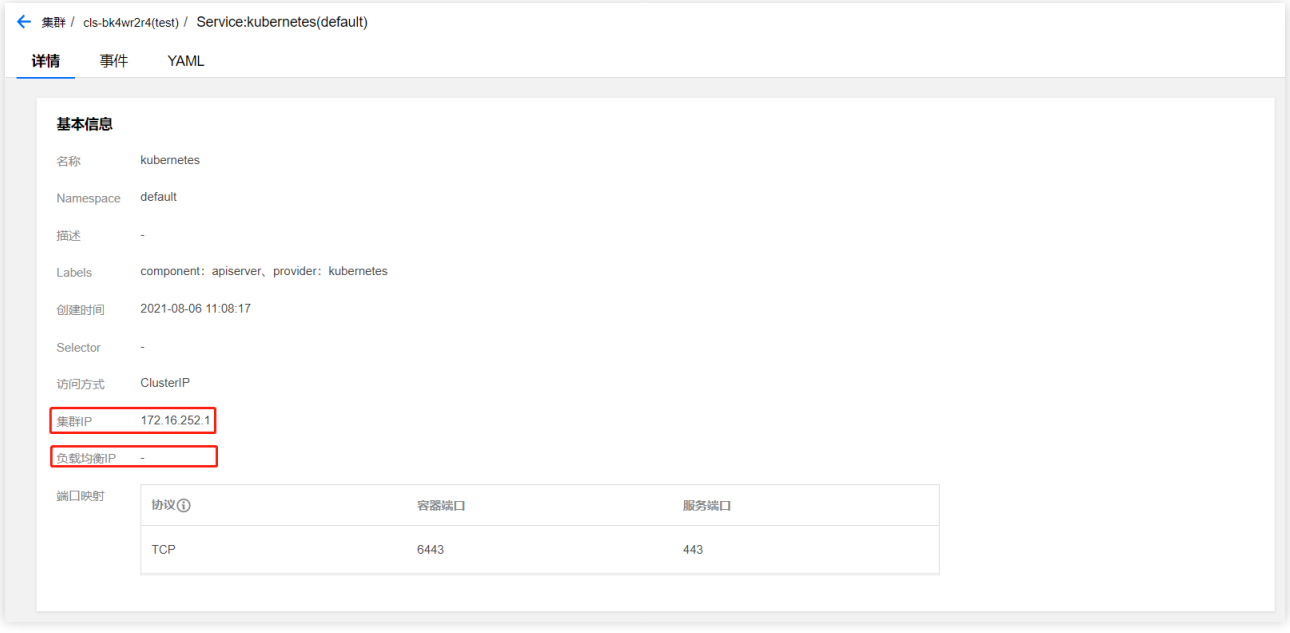
4. 在【新建Workload】页面里输入以下参数。
    - 工作负载名：输入自定义名称，这里以 nginx 为例。
    - **描述：**对此工作负载的信息描述。
    - 标签：默认值为工作负载名，支持新增标签变量。
    - **命名空间：**根据实际需求进行选择。默认为 default。
    - **类型：**选择工作负载的类型
    - 实例内容器
        - 名称：自定义，这里以 my-container 为例。
        - 镜像：根据实际需求进行选择，这里以 nginx 为例。
        - 镜像版本（Tag）：选择镜像版本（Tag）。
        - CPU/内存限制：配置该工作负载使用的CPU限制和内存限制。
        - GPU限制：配置该工作负载使用的最少GPU资源，请确保集群内已有足够的GPU资源。

        - 环境变量：配置该容器内的变量。
    - 实例数量：
        - 手动调节：直接设定实例数量。
        - 自动调节：满足任一设定条件，则自动调节实例（pod）数目。
    - 访问设置（Service）：
        - 勾选【启用】Service按钮，会新建一个与负载同名的Service。
        - 服务访问方式：为方便测试，这里选择 提供公网访问。
        - 端口映射：容器端口和服务端口都填80。
5. 单击【创建Workload】，完成 Nginx 服务的创建。

访问 Nginx 服务

提供两种方式访问 Nginx 服务。

- 通过负载均衡 IP 访问 Nginx 服务。
    i. 在【集群】页选择【服务】 > 【Service】，在Service列表里面单击刚刚新建的名为nginx的服务，进入nginx服务的详情页。



    ii. 可通过浏览器输入上图中的负载均衡 IP:服务端口来访问 Nginx 服务。
- 通过服务 名称访问 Nginx 服务。
  集群内的其他服务或容器可以直接通过服务名称访问。

进入 Nginx 服务器的默认欢迎页。


Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.

编写HelloWorld程序

本文档旨在帮助大家了解如何快速创建一个容器集群内的 Hello World 的 Node.js 版的服务。

第一步：编写代码制作镜像

编写应用程序

1. 创建一个 hellonode 的文件夹，加入 server.js 文件。

```
[root@VM_88_88_centos ~]# mkdir hellonode
[root@VM_88_88_centos ~]# cd hellonode/
[root@VM_88_88_centos hellonode]# vim server.js
[root@VM_88_88_centos hellonode]# ls
server.js
```

server.js 文件如下：

```
var http = require('http');
var handleRequest = function(request, response) {
  console.log('Received request for URL: ' + request.url);
  response.writeHead(200);
  response.end('Hello World!');
};
var www = http.createServer(handleRequest);
www.listen(8080);
```

2. 测试 Hello World 程序。

```
[root@VM_88_88_centos ~]# node server.js
```

打开新终端使用 curl 测试应用程序，或在浏览器以 IP 地址：端口的形式访问，端口为 8080 。

```
[root@VM_88_88_centos ~]# curl 127.0.0.1:8080
Hello World!
```

创建 Docker 镜像

构建 Docker 镜像更多详情见：如何构建 Docker 镜像。

1. 在 hellonode 文件夹下，创建 Dockerfile 文件：

```
FROM node:4.4
EXPOSE 8080
COPY server.js .
CMD node server.js
```

2. 通过 Docker build 命令构建镜像。

```
[root@VM_88_88_centos hellonode]# vim Dockerfile
[root@VM_88_88_centos hellonode]# ls
Dockerfile  server.js
[root@VM_88_88_centos hellonode]# docker build -t hello-node:v1 .
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM node:4.4
Trying to pull repository docker.io/library/node ...
4.4: Pulling from docker.io/library/node
......
......
Removing intermediate container 1e8d01dc319f
Successfully built 027232e62e3f
[root@VM_88_88_centos hellonode]# docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
hello-node                    v1                  027232e62e3f        54 minutes ago      647.4 MB
```

上传该镜像到 qcloud 镜像仓库

更多镜像操作详情见：镜像仓库基本教程。

```
[root@VM_3_224_centos hellonode]# sudo docker tag 027232e62e3f ccr.finance.cloud.tencent.com/test/helloworld:v1
[root@VM_3_224_centos hellonode]# sudo docker push ccr.finance.cloud.tencent.com/test/helloworld:v1
The push refers to a repository [ccr.finance.cloud.tencent.com/test/helloworld]
1b8da8805305: Pushed
20a6f9d228c0: Pushed
80c332ac5101: Pushed
04dc8c446a38: Pushed
1050aff7cfff: Pushed
66d8e5ee400c: Pushed
2f71b45e4e25: Pushed
v1: digest: sha256:38b194feeee09abf8ee45e7abca82b9fe494b18b953c771ce8ebefa387107be9 size: 1772
```

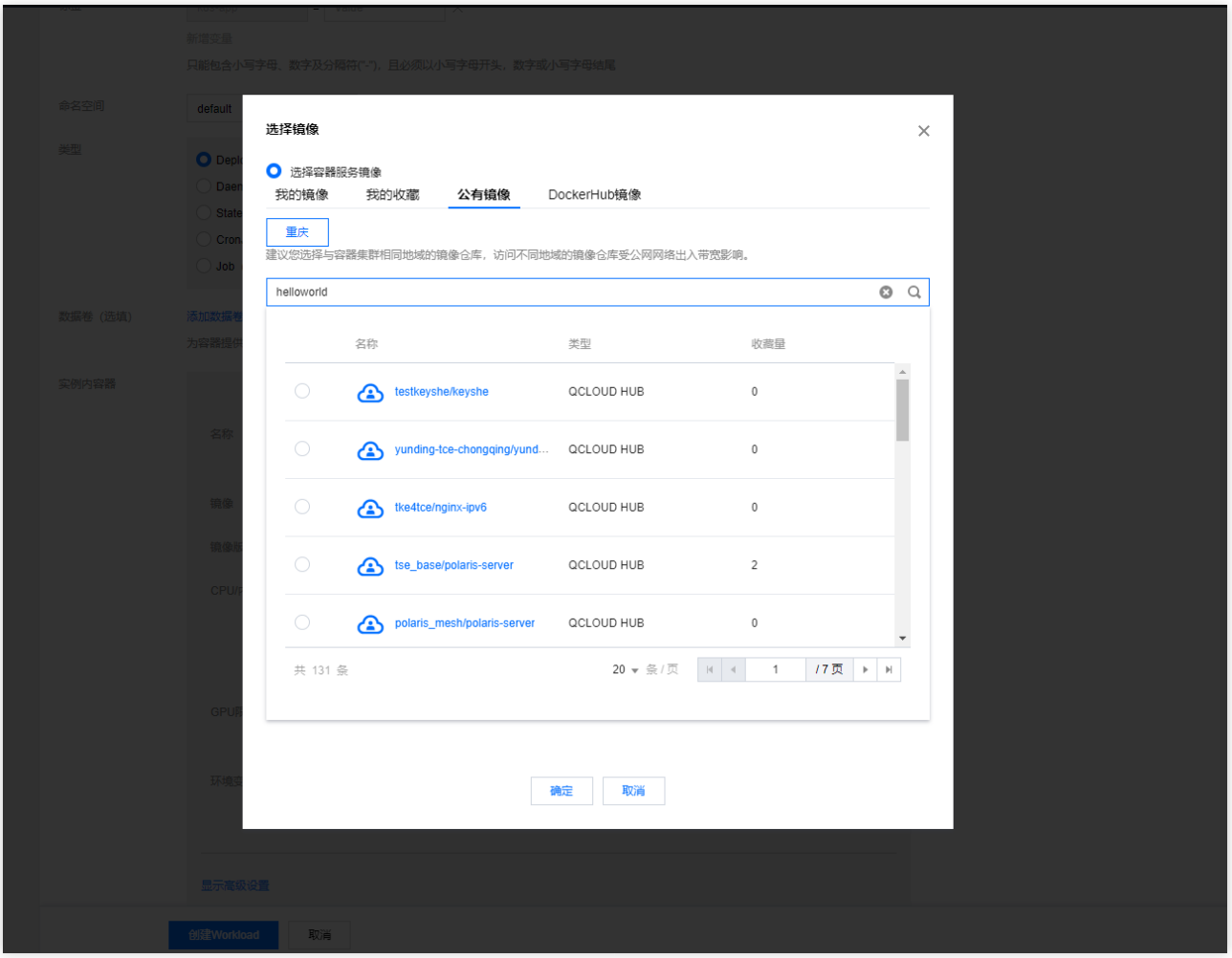
第二步：通过该镜像创建 Hello World 服务

> 注意：
> 在创建 helloworld 服务之前，您必须拥有: 一个创建好的集群。有关如何创建集群的详细信息，参见部署容器服务TKE 中步

> 骤1创建集群部分。

创建 helloworld 服务

1. 登录 TKE 控制台。
2. 单击左侧导航栏中的【集群】，单击集群列表页的一个集群【ID/名称】。
3. 在集群页选择【工作负载】 > 【Deployment】，在“Deployment”列表里面单击【新建】。
4. 在【新建Workload】页面输入以下参数。
    - 工作负载名：输入自定义名称，这里以 helloworld 为例
    - 实例内容器：
    - 名称：自定义，这里以 my-container 为例
    - 镜像：根据实际需求进行选择，这里以 helloworld 为例

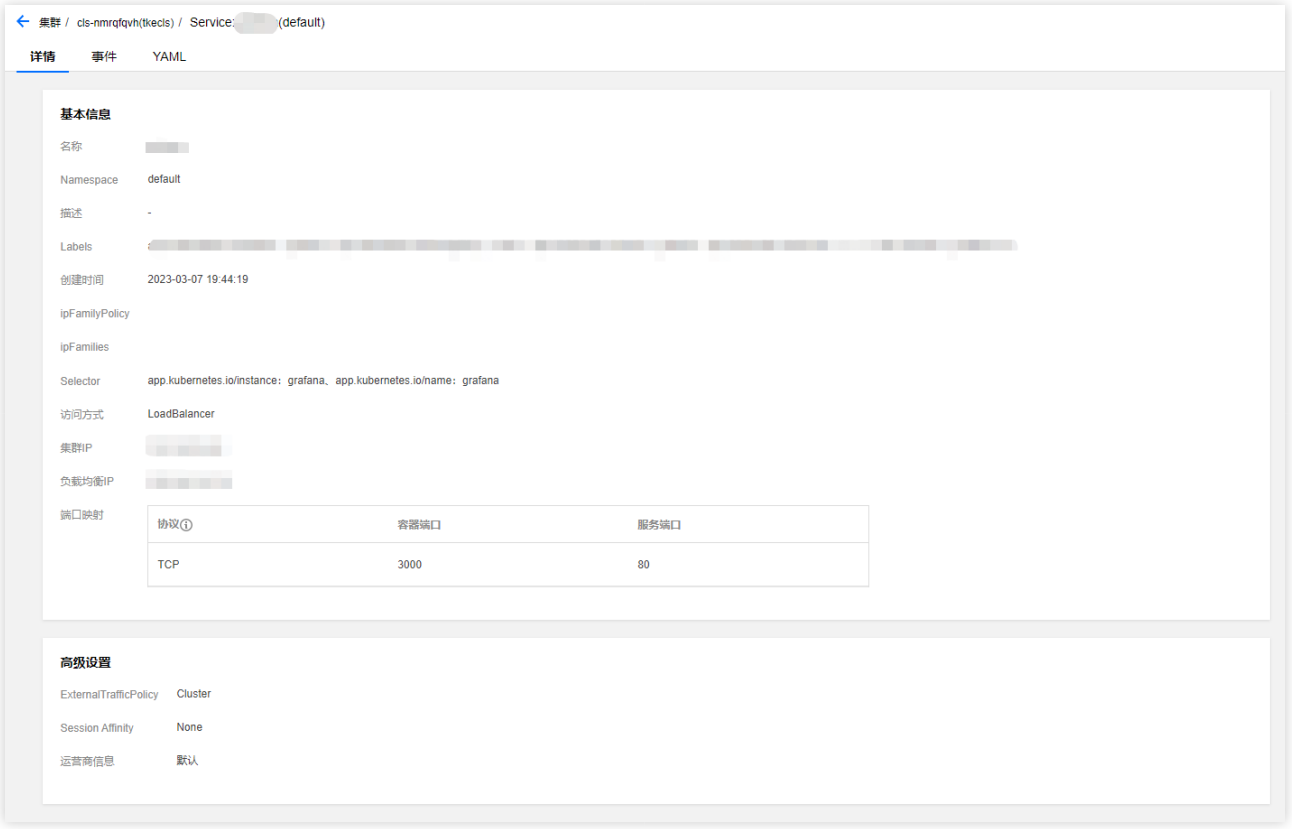


    - 访问设置（Service）：勾选【启用】Service按钮，会新建一个与负载同名的Service
    - 服务访问方式：为方便测试，这里选择 提供公网访问
    - 端口映射：容器端口和服务端口都填80
5. 单击创建Workload。

访问 helloworld 服务

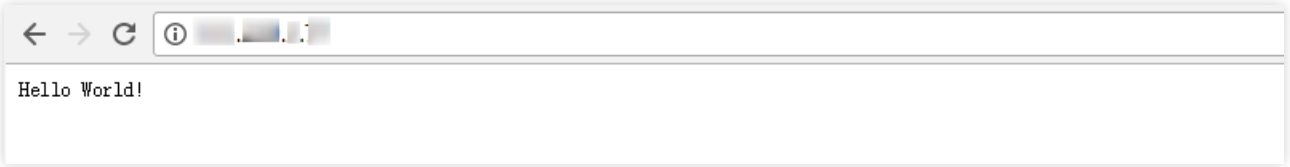
提供两种方式访问 helloworld 服务。

- 通过负载均衡 IP 访问 Hello World 服务
  i. 在集群页面选择【服务】 > 【Service】，在“Service”列表里单击刚刚新建的名为 helloworld 的服务，进入helloworld服务的详情页。



  ii. 在浏览器输入上图中的负载均衡IP:服务端口来访问 nginx 服务，

- 通过服务名称访问 Hello World 服务
  集群内的其他服务或容器可以直接通过服务名称访问。
  进入 Hello World 服务器的默认欢迎页。



单实例版WordPress

WordPress 是使用 PHP 语言开发的博客平台。用户可以在支持 PHP 和 MySQL 数据库的服务上架设属于自己的网站，也可以把 WordPress 当作一个内容管理系统来使用。
本文档旨在介绍如何使用 tutum/wordpress 镜像来创建一个公开访问的 WordPress 网站。

> 注意：
> 创建单实例版的 WordPress 仅供测试使用，该镜像中包含了 WordPress 所有的运行环境，直接拉取创建服务即可，但使用单实例版的 WordPress 不能保证数据的持久化存储，建议您使用自建的 MySQL 或使用TCloudFinanceZone 数据库 CDB 来保存您的数据。详情请参考 使用 CDB 的 WordPress。
> 在创建 WordPress 服务之前，您必须拥有:
>
> - 一个TCloudFinanceZone帐户。有关如何创建TCloudFinanceZone帐户，请在 [注册页面] 填写相关信息注册TCloudFinanceZone帐户。
> - 一个创建好的集群。有关如何创建集群的详细信息，参见 新建集群 。

创建 WordPress 服务

1. 登录 TKE 控制台。
2. 单击左侧导航栏中的【集群】，单击集群列表页的一个集群【ID/名称】。
3. 在【集群】页面，选择【工作负载】 > 【Deployment】，在Deployment列表里面单击【新建】。
4. 在"新建Workload"页面，根据实际情况，设置工作负载基本信息。
5. 镜像配置。
   - 名称：输入运行容器的名称，此处以 wordpress 为例。
   - 镜像：填写 tutum/wordpress 。
   - 版本（Tag）：填写 latest。
6. 设置访问设置（Service）。勾选【启用】Service，将容器端口和服务端口都设置为80 。

   > 服务所在集群的安全组需要放通节点网络及容器网络，同时需要放通30000-32768端口，否则可能会出现容器服务无法使用问题。详情参见 容器服务安全组设置



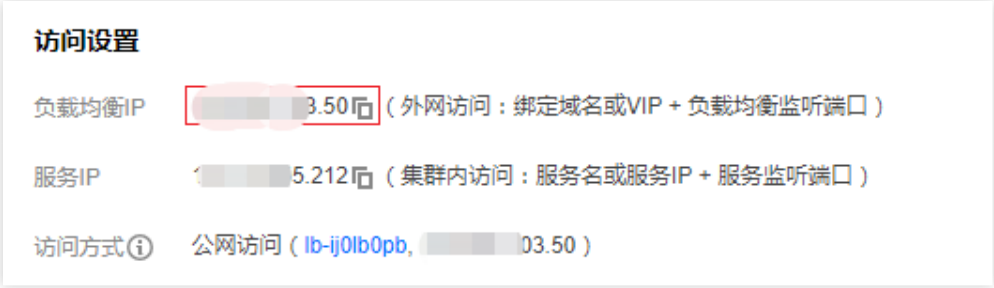
7. 单击创建 Workload。完成 WordPress 服务的创建。

> 其他选项保持为默认设置。

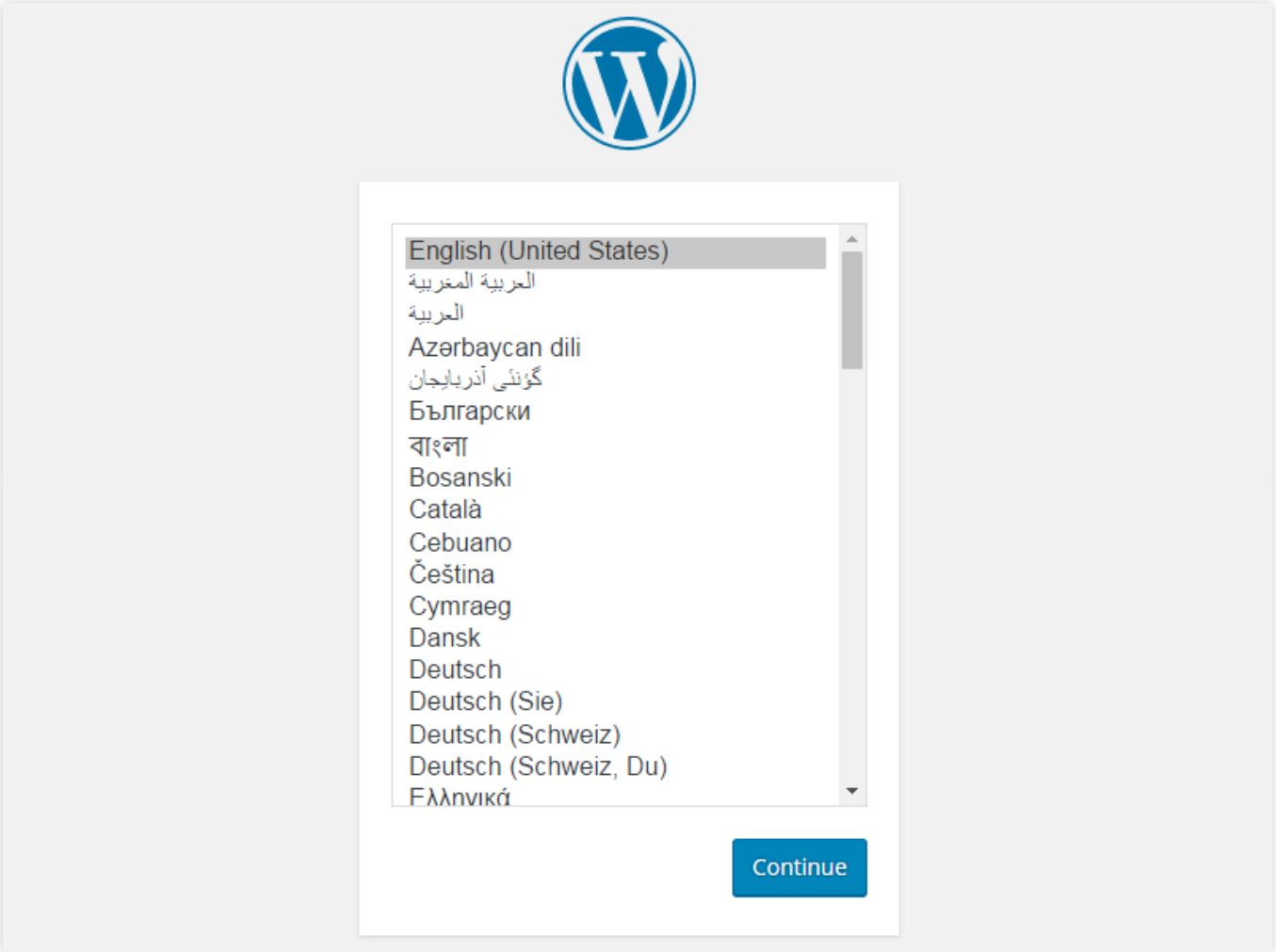
访问 WordPress 服务

提供以下方式访问 WordPress 服务。

- 通过负载均衡 IP来访问 WordPress 服务。单击服务页面的【服务信息】查看负载均衡 IP和负载均衡ID。



- 集群内的其他服务或容器可以直接通过服务名称访问。
  进入 WordPress 服务器的默认欢迎页。



若容器创建失败，可查看事件常见问题。

使用CDB的wordpress

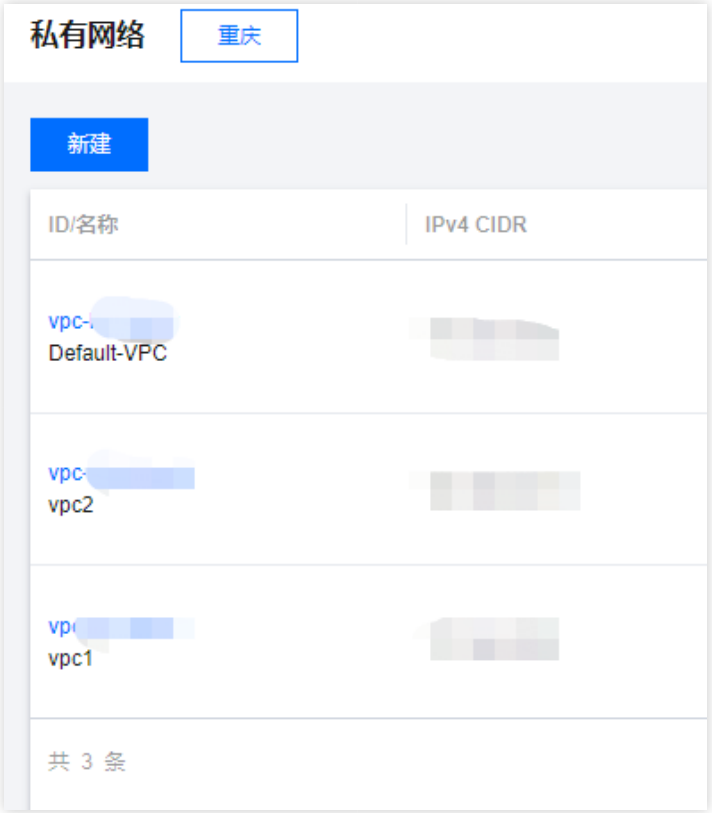
在单实例版 WordPress 示例中我们介绍了如何快速创建 WordPress 服务。单实例版 WordPress 的数据是写到同一个容器运行的 MySQL 数据库中，虽然这样的配置可以快速启动，但它也存在一个问题：如果容器因某种原因停止，数据库和存储类的文件将会丢失。
本文档旨在介绍如何设置 MySQL 数据库，它将在实例/容器重新启动后继续存在。通过使用 云数据库CDB 可以实现永久存储。

> 注意：
> 在创建使用 CDB 的 WordPress 服务之前，您必须拥有:
>
> - 一个TCloudFinanceZone账户。有关如何创建TCloudFinanceZone账户，请在 注册页面 填写相关信息注册TCloudFinanceZone账户。
> - 一个创建好的集群。有关如何创建集群的详细信息，参见 新建集群 。

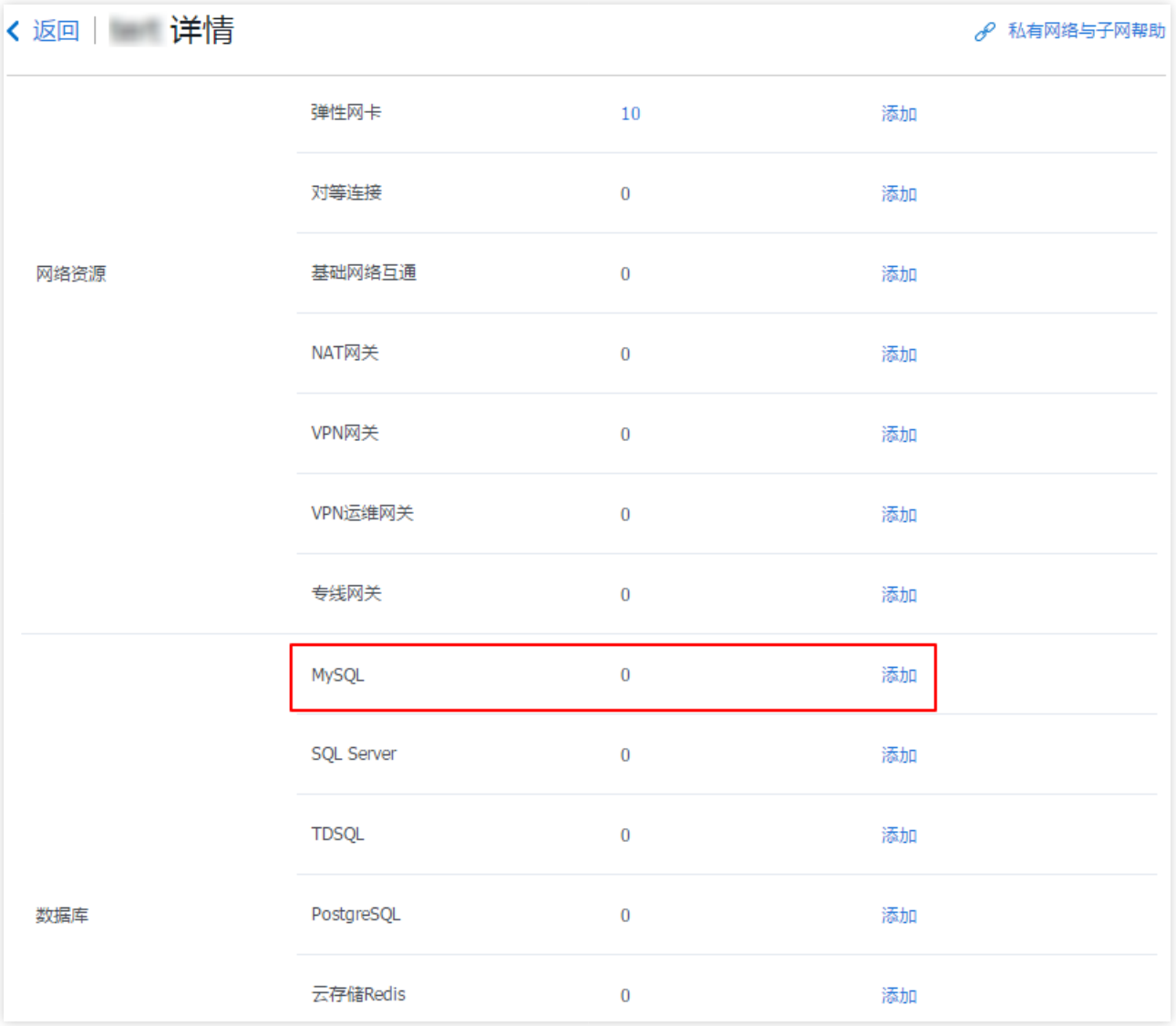
创建 WordPress 服务

第一步：创建云数据库 CDB

1. 登录私有网络控制台 。
2. 单击私有网络列表页的【ID/名称】（如：vpc-xxxxx）。



3. 在私有网络详情页，选择数据库目录下的 MySQL，单击右侧【添加】。



4. 选择购买配置，完成系列支付操作。相关详情请参见 数据库MySQL。
5. 购买的 MySQL 将出现在 MySQL 实例列表中。
6. 初始化 MySQL 实例。单击右侧 操作 列下的【初始化】。
7. 配置初始化相关参数，然后单击【确定】开始初始化。
   - 支持字符集：选择 MySQL 数据库支持的字符集。
   - 表名大小写敏感：表名是否大小写敏感，默认为是。
   - 自定义端口：数据库的访问端口，默认为 3306。
   - root账户密码：新创建的 MySQL 数据库的用户名默认为 root，此处用来设置此 root 账户的密码。
   - 确认密码：再次输入密码。
8. 目标 MySQL 实例的状态变为 运行中，说明已初始化成功。

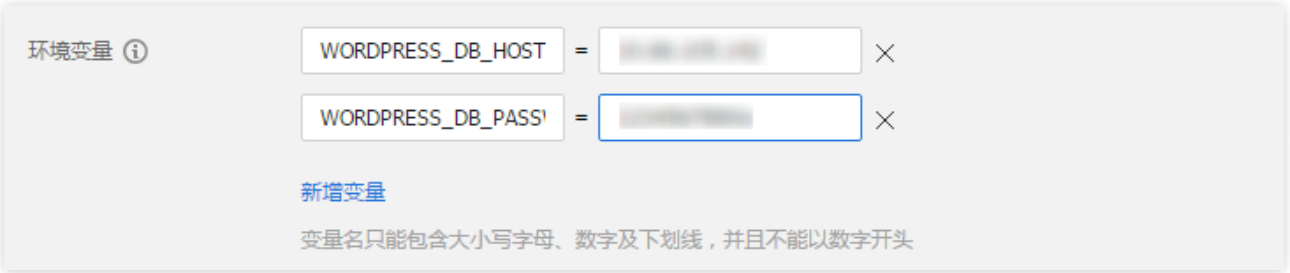
第二步：创建使用 CDB 的 WordPress 服务

1. 登录 TKE 控制台。
2. 单击左侧导航栏中的【集群】，单击集群列表页的一个集群【ID/名称】。
3. 在【集群】页面，选择【工作负载】 > 【Deployment】，在Deployment列表里面单击【新建】。
4. 在“新建Workload”页面，根据实际情况，设置工作负载基本信息。

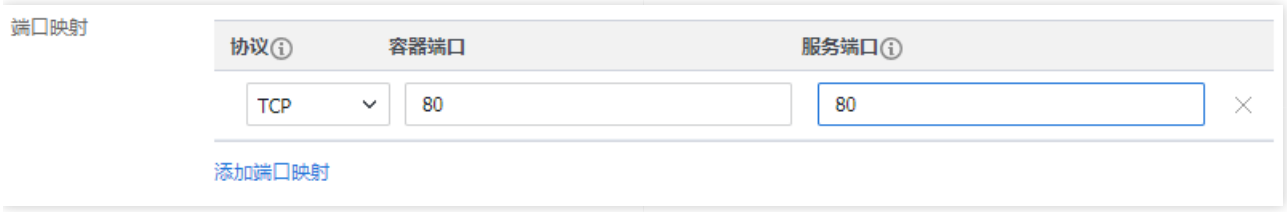
5. 镜像配置。
   - 名称：输入运行容器的名称，此处以 wordpress 为例。
   - 镜像：填写 wordpress 。
   - 版本（Tag）：填写 latest。



6. 单击运行容器下的【显示高级设置】，在弹出的下拉列表中，单击环境变量下的【新增变量】。依次填写：
   WORDPRESS_DB_HOST = 云数据库 MySQL 的地址
   WORDPRESS_DB_PASSWORD = 初始化时填写的密码



7. 设置端口映射。将容器端口和服务端口都设置为80。 > 注意：服务所在集群的安全组需要放通节点网络及容器网络，同时需要放通 30000-32768 端口，否则可能会出现容器服务无法使用问题。详情参见 容器服务安全组设置



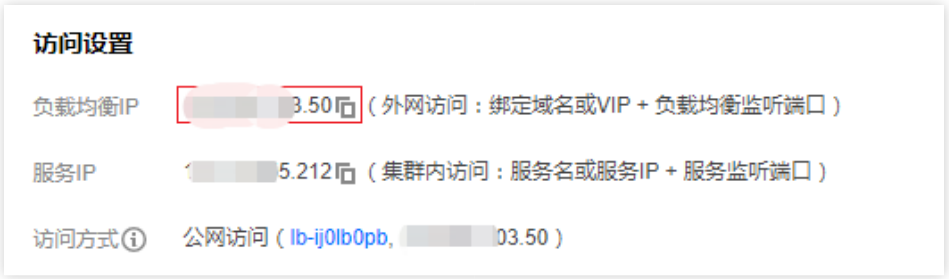
8. 单击 创建 Workload。完成 WordPress 服务的创建。 > 注意：其他选项保持为默认设置。

访问 WordPress 服务

提供三种方式访问 WordPress 服务。

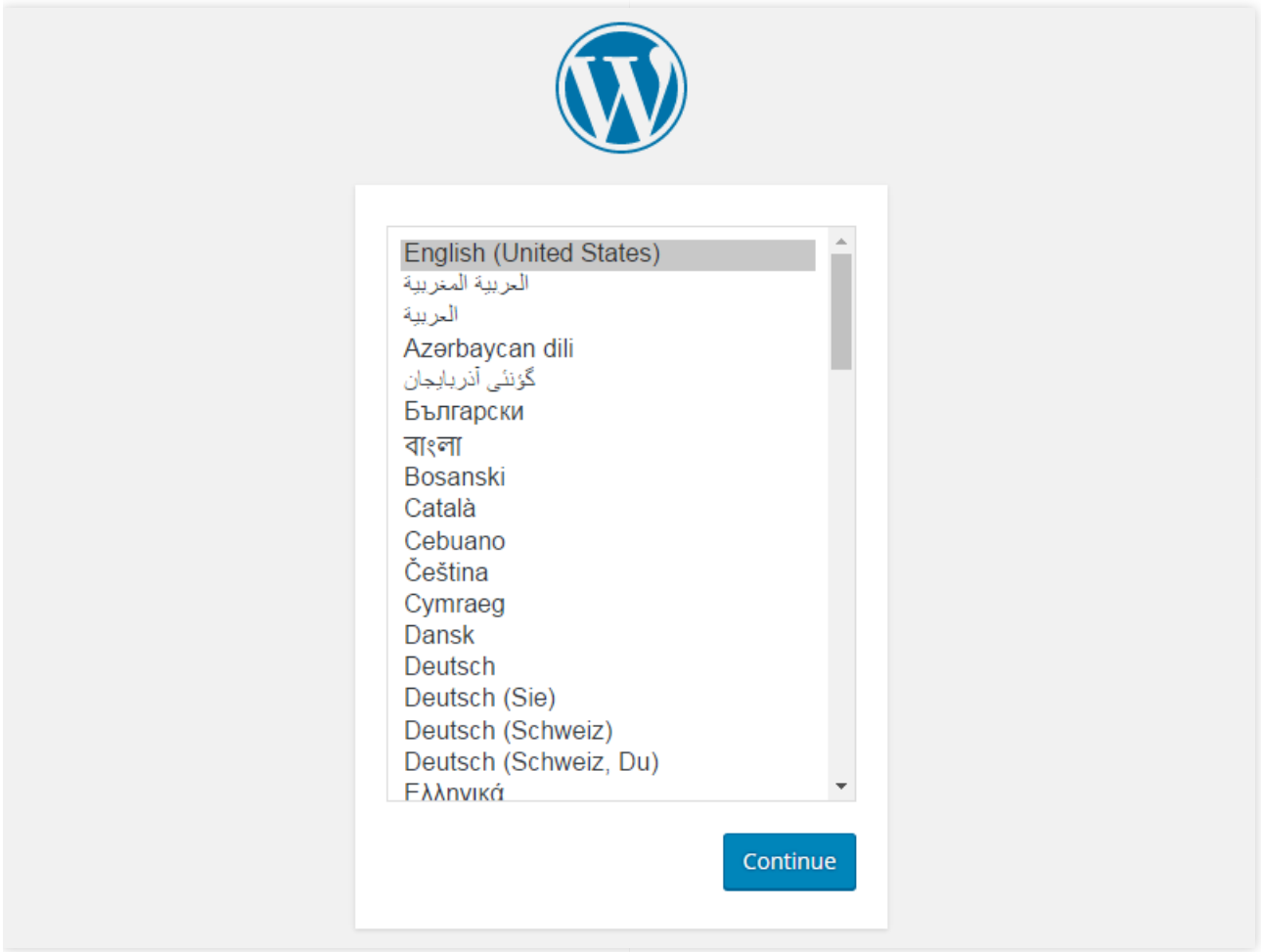
- 通过负载均衡 IP来访问 WordPress 服务。
  单击服务页面的【服务信息】查看负载均衡 IP 和负载均衡 ID。





- 通过 域名 来访问 WordPress 服务。
  在容器服务控制台左侧导航栏中，单击【负载均衡】，单击【TCP/UDP】，找到对应的负载均衡ID，复制域名访问服务。
- 集群内的其他服务或容器可以直接通过服务名称访问。

进入 WordPress 服务器的默认欢迎页。



若容器创建失败，可查看 事件常见问题。

操作指南
集群管理
集群概述

集群基本信息

集群是指容器运行所需云资源的集合，包含若干台云服务器、负载均衡器等资源。您可以在集群中运行您的应用程序。

集群架构

TKE 采用兼容标准的 Kubernetes 集群，包含以下组件：

- Master：用于管控集群的管理面节点。
- Etcd：保持整个集群的状态信息。
- Node：业务运行的工作节点。

集群类型

TKE 容器集群支持以下两种类型：

集群类型	描述
托管集群	Master、Etcd 云平台容器服务管理
独立集群	Master、Etcd 采用用户自有主机搭建

集群相关操作

- 创建集群
- 集群扩缩容
- 连接集群
- 升级集群
- CVM 容器集群网络

集群的托管模式说明

Master 托管模式

简介

容器服务 TKE 提供 Master、Etcd 全部托管的 Kubernetes 集群管理服务。
该模式下，Kubernetes 集群的 Master 和 Etcd 会集中管理和维护。您只需要购置集群，运行负载所需的工作节点即可，不需要关心集群的管理和维护。

Master 托管模式注意事项

该模式下，即使您删除集群的全部工作节点，集群仍会不断尝试运行您未删除的工作负载和服务，导致在此过程中可能会产生费用。如果您决定终止集群服务和费用产生，请直接删除该集群。

Master 独立部署模式

简介

TKE 也为您提供集群完全自主配置的 Master 独立部署模式。
选择该模式，Kubernetes 集群的 Master 和 Etcd 将会部署在您购置的 CVM 上。您拥有 Kubernetes 集群的所有管理和操作权限。

Master 独立部署模式注意事项

- 该模式下，Kubernetes 集群的 Master 和 Etcd 需要您额外购置资源部署。
- 如果您的集群规模较大，推荐选择高配机型。机型选择请参考：

集群规模	建议 Master 节点配置	建议节点数量
约100个节点	8核16GB SSD 系统盘	3台以上
约500个节点	16核32GB SSD 系统盘	3台以上
1000个节点以上	请根据实际需要合理配置	3台以上

购置限制说明

为了保证集群和服务的高可用性和提高集群性能，在独立部署模式下，设置以下限制：

- Master&Etcd 节点要求至少部署3台。
- Master&Etcd 节点需配置4核及以上的机型。
- Master&Etcd 节点选择 SSD 盘作为系统盘。

注意事项

为了保证集群的稳定性，以及发生异常后的恢复效率，建议如下：
在 Master 独立部署模式下：

- 请不要删除 Master 节点下支撑 Kubernetes 运行的核心组件。
- 请不要修改 Master 核心组件的配置参数。
- 请不要修改/删除集群内部的核心资源。
- 请不要修改/删除 Master 节点的相关证书文件（拓展名为 .crt，.key）。
  非必要情况下：
- 请不要修改任何节点的 docker 版本。
- 请不要修改任何节点操作系统的 kernel、nfs-utils 等相关组件。

> 说明
>
> - 核心组件：kube-APIserver，kube-scheduler，kube-controller-manager，tke-tools，systemd，cluster-contrainer-agent。
> - 核心组件配置参数：kube-APIserver 参数，kube-scheduler 参数，kube-controller-manager 参数。
> - 集群内部核心资源（包括但不限于）：hpa endpoint，master service account，kube-dns，auto-scaler，master cluster role， master cluster role binding。

集群生命周期

集群生命周期状态说明

状态	说明
创建中	集群正在创建，正在申请云资源。
规模调整中	集群的节点数量变更，添加节点或销毁节点中。
运行中	集群正常运行。
升级中	升级集群中。
删除中	集群在删除中。
异常	集群中存在异常，如节点网络不可达等。

> 说明：
> 容器服务基于 Kubernetes 且为声明式服务。如果您已在容器服务中创建 CLB、CBS 盘等 IAAS 资源，现在不再需要使用 CLB 和 CBS，请在 TKE 控制台中删除对应的 Service 和 PersistentVolumeClaim 对象。如果您只在 CLB 控制台中删除 CLB 或者在 CBS 控制台中删除 CBS，容器服务会重新创建新的 CLB 和 CBS。

创建集群

操作场景

创建集群时，支持新增云服务器作为集群的初始节点，还支持通过选择已经存在的云服务器作为集群的初始节点。本文档指导您通过这两种方式创建集群。

操作系统说明

- 修改操作系统只影响后续新增的节点或重装的节点，对存量的节点操作系统无影响。
- 目前仅支持同类型的操作系统修改如：CentOS > CentOS 类的自定义镜像。

操作步骤

填写集群信息

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。



3. 在【集群管理】页单击【新建】。
4. 设置集群的基本信息，单击【下一步】。
    - 集群名称：您要创建的集群的名称。不超过60个字符。
    - **CPU架构：**根据需要选择对应架构。
    - Kubernetes 版本：选择 Kubernetes 版本。
    - 集群 IP 类型：提供IPv4和IPv4/IPv6双栈
    - **运行时组件：**提供docker和containerd两种选择。

- 所在地域：建议您根据所在地理位置选择靠近的地域。可降低访问延迟，提高下载速度。
- 集群网络：如现有的网络不合适，您可以去控制台 新建私有网络。
- 容器网络插件：选择容器网络插件。
- 容器网络：为集群内容器分配在容器网络地址范围内的 IP 地址，单节点Pod数量上限和集群内Service数量上限。
- 操作系统：选择操作系统。
- 集群描述：创建集群的相关信息。该信息将显示在 集群信息 页面。
- 高级设置：
  - 可以开启“ipvs支持”：注意开启后将不支持关闭，适用于大规模场景下提供更优的转发性能。
  - 可以添加“标签”：为TKE集群配置标签，集群内创建的云服务的资源自动继承集群标签。
  - 自定义参数：指定自定义参数来配置集群。详情请参见 自定义 Kubernetes 组件启动参数。

选择机型

1. 在“选择机型”中，选择部署模式和机型。
   主要参数信息如下：
   - Master：Master 的部署方法决定了您集群的管理模式。
   - Node：Node 配置的是集群运行服务真正使用的工作节点。您可以在创建集群时购置云服务器作为 Node 节点，也可以在集群创建完成后再添加 Node 节点。
   - Master&Etcd机型：根据如下要求，选择云服务器。
     - 可用区：您可以同时选择多个可用区部署您的 Master 或 Etcd，保证集群更高的可用性。
     - 节点网络：您可以同时选择多个子网的资源部署您的 Master 或 Etcd，保证集群更高的可用性。
     - 机型：选择大于 CPU 4核的机型，具体选择方案请参看 实例类型概述。
     - 系统盘：默认为“本地硬盘50G”，您可以根据需要进行配置。
     - 数据盘：Master 和 Etcd 因为不建议部署其他应用，默认不配置数据盘，您可以购置后再添加云盘。
     - 公网宽带：勾选分配免费公网IP，系统将分配公网 IP。
     - 数量：设置>=3台。
   - Node机型：当 “Node” 选择为 “立即部署” 时，可选。您可以选择已有的云服务器作为 Node 节点，也可以在集群创建完成后再添加 Node 节点。
     - 可用区：您可以同时选择多个可用区部署您的 Master 或 Node 节点，保证集群更高的可用性。
     - 节点网络：您可以同时选择多个子网的资源部署您的 Master 或 Node 节点，保证集群更高的可用性。
     - 机型：机型选择方案参看 实例类型概述。
     - 系统盘：默认为“本地硬盘50G”，您可以根据需要进行配置。
     - 数据盘：Node 可以在购置时根据需要配置数据盘。
     - 公网宽带：勾选分配免费公网IP，系统将分配公网 IP。
     - 数量：设置>=1台。
2. 单击【下一步】，配置云主机。

配置云服务器

1. 在 “云主机配置” 中，配置云服务器的其他配置。



主要参数信息如下：

- 数据盘挂载：默认勾选。
- 安全组：安全组具有防火墙的功能，用于设置云服务器的网络访问控制。参看容器服务安全组设置 。
- 登录方式：提供三种登录方式。
  - 设置密码：请根据提示设置对应密码。
  - 立即关联密钥：密钥对是通过算法生成的一对参数，是一种比常规密码更安全的登录云服务器的方式。详细参看 SSH 密钥。
  - 自动生成密码：自动生成的密码将通过站内信发送给您。
- 安全加固：可选免费开通，安装组件免费开通DDoS防护、WAF和云镜主机防护
- 云监控：可选免费开通，免费开通云产品监控、分析和实施告警，安装组件获取主机监控指标
- 高级设置：
  - 自定义数据：用于启动时配置实例
  - 封锁(cordon)：封锁节点后，将不接受新的Pod调度到该节点

2. 单击【下一步】，检查并确认配置信息。
3. 单击【完成】，即可完成创建。

集群扩缩容

操作场景

本文档指导您对集群进行扩缩容。TKE 容器集群支持以下扩缩容方法：

- 手动添加/移除节点
- 通过弹性伸缩自动添加/移除节点
- 扩容缩容节点

前提条件

已登录 TKE 控制台。

操作步骤

手动添加/移除节点

您可通过 新建节点方式进行手动添加节点，实现集群的手动扩容。通过 移除节点，实现集群的手动缩容。

新建节点

具体操作请参考新建节点指引。
创建过程中，您可以在“云主机配置”页面，配置云服务器，并对集群进行扩缩容。

移除节点

具体操作请参考 移除节点。

扩容缩容节点

可在 CVM 管理页面直接对 TKE 集群使用到的虚拟机节点进行扩容或缩容，由于该操作 TKE 集群无法感知配置变化，需要管理员登录节点，对节点的 kubelet 进行重启操作，以便让 TKE 集群感知配置的变化。

连接集群

操作场景

您可以通过 Kubernetes 命令行工具 Kubectl 从本地客户端机器连接到 TKE 集群。本文档指导您如何连接集群。

准备的软件

请根据操作系统的类型，选择获取 Kubectl 工具的方式：

> 说明：
> 根据实际需求，将命令行中的 *v1.8.13* 替换成业务所需的 Kubectl 版本。

- Mac OS X 系统
  执行以下命令，获取 Kubectl 工具：

  ```
  curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.8.13/bin/darwin/amd64/kubectl
  ```

- Linux 系统
  执行以下命令，获取 Kubectl 工具：

  ```
  curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.8.13/bin/linux/amd64/kubectl
  ```

- Windows 系统
  执行以下命令，获取 Kubectl 工具：

  ```
  curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.8.13/bin/windows/amd64/kubectl.exe
  ```

操作步骤

安装 Kubectl 工具

1. 参考 Installing and Setting up kubectl，安装 Kubectl 工具。 > 说明：

如果您已经安装 Kubectl 工具，请忽略本步骤。

2. 执行以下命令，添加执行权限。

```
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
```

3. 执行以下命令，测试安装结果。

```
kubectl version
```

如若输出类似以下版本信息，即表示安装成功。

```
Client Version: version.Info{Major:"1", Minor:"5",  GitVersion:"v1.5.2", GitCommit:"08e099554f3c31f6e6f07b448ab3ed78d0520507", GitTreeState:"clean", BuildDate:"2017-01-12T04:57:25Z", GoVersion:"go1.7.4", Compiler:"gc", Platform:"linux/amd64"}
```

获取集群账号密码以及证书信息

1. 登录 TKE 控制台。

2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。

3. 单击需要连接集群的【ID/名称】，进入该集群的管理页面。

4. 在左侧导航栏中，单击【基本信息】，进入“基本信息”页面。



5. 在“基本信息”中，单击【集群凭证】中的 【显示凭证】。

6. 在弹出的“集群凭证”窗口中，查看用户名、密码和证书信息。 > 说明：
   您可以根据实际需求，单击【复制】或【下载】将集群 CA 证书保存到本地。

7. 在弹出的“集群凭证”窗口中，获取访问入口。

- 集群内直接访问：“外网访问地址” 和 “内网访问地址” 保持默认值，无须进行任何配置，即可直接在集群内的主机上执行 Kubectl 命令。
- 获取公网访问入口：将 “外网访问地址” 设置为 “已开启”，可参考 设置 Kubectl 命令自动补全 直接使用外网访问地址进行访问。
- 获取 VPC 内网访问入口：将 “内网访问地址” 设置为 “已开启”，指定客户端主机的 hosts，用于支持域名解析。即，在 /etc/hosts 文件后追加内网返回的 IP 和域名。您可以手动设置，也可以参考以下代码进行设置。

```
sudo sed -i '$a **IP地址** **域名**' /etc/hosts
```

完成配置后，您可参考 设置 Kubectl 命令自动补全 使用内网访问地址域名进行访问。 > 说明：

若集群无可用节点（包括节点异常,已封锁等状态），内网访问将在集群内有可用节点时生效。

8. 单击【关闭】。

通过证书信息使用 Kubectl 操作集群

单次 Kubectl 操作请求，附带证书信息

> 说明：
> 该方法适用于单次 Kubectl 操作集群，无需将容器集群的证书信息保存到机器上。

请求方法
Kubectl 命令格式如下所示：

```
-s "域名信息" --username=用户名 --password=密码 --certificate-authority=证书路径
```

示例

```
kubectl get node -s "https://cls-66668888.ccs.abcd-cloud.com" --username=admin --password=6666o
9oIB2gHD88882quIfLMy6666 --certificate-authority=/etc/kubernetes/cluster-ca.crt
```

修改 Kubectl 配置文件，长期有效

> 说明：
> 该方法适用于长期通过 Kubectl 操作集群，只需配置一次，不修改文件即可长期有效。

1. 参考以下命令，修改 Kubectl 配置文件中的密码、证书信息。

```
kubectl config set-credentials default-admin --username=admin --password=6666o9oIB2gHD88
882quIfLMy6666
kubectl config set-cluster default-cluster --server=https://cls-66668888.ccs.abcd-cloud.com --cer
```

```
tificate-authority=/etc/kubernetes/cluster-ca.crt
kubectl config set-context default-system --cluster=default-cluster --user=default-admin
kubectl config use-context default-system
```

2. 配置完成后，执行以下命令，获取 node 节点信息。

```
kubectl get nodes
```

返回类似以下信息，即表示修改成功。

```
NAME        STATUS    AGE
10.0.0.61   Ready     10h
```

设置 Kubectl 命令自动补全

您可以通过执行以下命令，配置 Kubectl 自动补全，提高可使用性。

```
source <(kubectl completion bash)
```

升级集群

操作场景

容器服务 TKE 提供升级 Kubernetes 版本的功能，您可通过此功能对运行中的 Kubernetes 集群进行升级。升级的过程为：升级的前置检查、升级 Master 和升级 Node。

升级须知

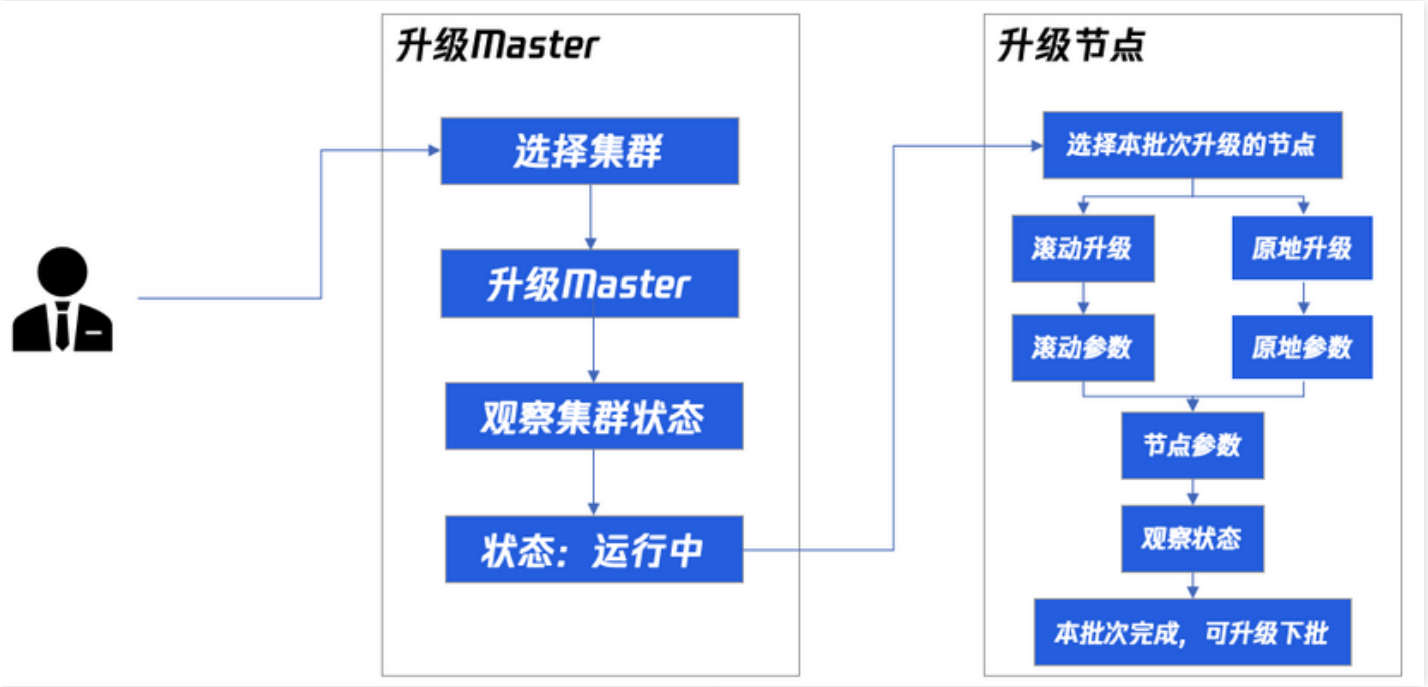
- 升级属于不可逆操作、请谨慎进行。
- 请在升级集群前，查看集群下状态是否均为健康状态。若集群不正常，您可以自行修复，也可以通过提交工单联系我们协助您进行修复。
- 升级顺序：升级集群时，必须先完成 Master 版本升级，再尽快完成 Node 版本升级，且升级过程中不建议对集群进行任何操作。
- 仅支持向上升级 TKE 提供的最近 Kubernetes 版本，不支持跨多个版本升级（例如1.8跳过1.10直接升级至1.12），且仅当集群内 Master 版本和 Node 版本一致时才可继续升级下一个版本。
- CSI-CFS 插件不兼容问题：关于 CSI 插件 COS CSI 和 CFS CSI，不同 Kubernetes 版本适配的 CSI 插件版本有以下差异，因此建议您：将集群升级到 TKE 1.14及以上版本时，在组件管理页面重新安装 CSI 插件（重建组件不影响已经在使用中的 COS 和 CFS 存储）。
  - Kubernetes 1.10 和 Kubernetes 1.12 版本适配的 CSI 插件版本是0.3。
  - Kubernetes 1.14 及以上版本适配的 CSI 插件版本是1.0。
- HPA 失效问题：在 Kubernetes 1.18版本之前，HPA 中所引用的 deployment 对象的 apiversion 可能是 extensions/v1beta1，而 Kubernetes 1.18版本之后，deployment 的 apiversion 只有 apps/v1 ，可能导致集群升级到 Kubernetes 1.18之后，HPA 会失效。
  如果您使用了 HPA 功能，建议在升级之前，执行如下命令，将 HPA 对象中的 apiVersion 切换到 apps/v1 。

```
kubectl patch hpa test -p '{"spec":{"scaleTargetRef":{"apiVersion":"apps/v1"}}}'
```

- Helm 应用失效问题：每个应用支持的 Kubernetes 的版本不太相同，包括通过应用市场安装的应用或是通过第三方安装的应用。建议在升级集群前，查看已安装在集群里的应用列表，确认其支持的集群版本范围。有些应用本身对高版本的 Kubernetes 有适配，此时可能需要升级应用的版本。有些应用可能还没有对高版本的 Kubernetes 适配，此时请谨慎升级集群。
- Nginx Ingress 版本问题：extensions/v1beta1 和 networking.k8s.io/v1beta1 API 版本的 Ingress 不在 v1.22 版本中继续提供。您集群里面 Nginx Ingress 的版本可能比较低，在升级 Kubernetes 版本到 v1.22 及以上版本时，在组件管理页面升级 Nginx Ingress 组件。

操作步骤

升级集群的两个步骤是 升级 Master Kubernetes 版本 和 升级 Node Kubernetes 版本。



升级 Master Kubernetes 版本

说明：目前已支持托管集群、独立集群 Master 版本升级，且升级需要花费5 - 10分钟，在此期间您将无法操作您的集群。

Master 大版本与小版本升级说明

目前 Master 升级已支持大版本升级（例如从1.14升级到1.16）、小版本升级（例如从1.14.3升级到1.14.6，或者从v1.18.4-tke.5升级到v1.18.4-tke.6）。
说明：

- 当大版本升级（例如1.12升级到1.14），若您已设置自定义参数，需要您重新设置新版本的自定义参数。原参数不保留。详情可参见 自定义 kubernetes 组件启动参数。
- 当小版本升级时，您已设置的自定义参数会被保留，无需重新设置。

注意事项

- 升级前，请详细阅读 升级须知。
- 1.7.8版本 TKE 集群，网络模式为 bridge，集群升级不会自动切换网络模式为 cni。
- 集群升级不会切换 kube-dns 为 core-dns。
- 创建集群时设置的部分特性（例如支持 ipvs），当集群 Master 版本升级到1.10和1.12后将不支持开通。
- 存量的集群升级后，若 Master 版本在1.10版本以上，Node 节点版本在1.8版本以下，PVC 功能将不可用。
- 升级 master 完成后，建议您尽快升级节点版本。

Master 升级技术原理

Master 节点升级分为3个步骤：前置组件升级、Master 节点组件升级、后置组件升级。

- 升级前置操作：将会升级前置依赖的组件，例如监控组件等，以防兼容性问题导致组件异常。
- Master 节点组件升级：将按组件顺序对所有 Master 的对应组件进行升级，所有 Master 的某个组件升级完成后再进行下一个组件的升级。
  TKE 会先升级 kube-apiserver，后升级 kube-controller-manager 和 kube-scheduler，最后升级 kubelet。具体步骤如下：
  - 重新生成 kube-apiserver 组件静态 Pod 对应的 yaml 文件内容。
  - 检查当前 kube-apiserver pod 是否健康，kubernetes 版本是否正常。
  - 同理，依次升级 kube-controller-manager 和 kube-scheduler。
  - 升级 kubelet，并检查所在 Master 节点是否 ready。
- 升级后置操作：
  - 按需升级后置依赖组件，如 kube-proxy（并将其滚动更新策略改为 on delete）、cluster-autoscaler 组件等。
  - 执行一些后置依赖组件相关的兼容性操作，防止兼容性问题导致组件异常。

Master 升级操作步骤

1. 登录容器服务控制台，选择左侧导航栏中的 集群。
2. 在集群管理页面，选择所需升级集群操作列的【升级Master Kubernetes版本】，进入升级Master页面。
3. 仔细阅读完升级须知后，单击【我已阅读并统一上述技术条款】。
4. 单击【下一步】，进入升级设置页面。
5. 选择升级目标版本后，单击【完成】，等待升级完成。
6. 您可以在集群管理页对应的集群状态处查看升级进度。

升级 节点 Kubernetes 版本

集群 Master Kubernetes 版本升级完成后，集群列表页将显示该集群节点有可用升级。

注意事项

- 升级前，请详细阅读升级须知。
- 当 Node 节点处于运行中时，可进行升级操作。

选择升级方式

升级 节点 Kubernetes 版本支持 重装滚动升级 和 原地滚动升级 两种升级方式。您可按需选择：

- 重装滚动升级：采用重装节点的方式升级节点版本。仅支持大版本升级，例如1.10可升级至1.12。
- 原地滚动升级：原地不重装，仅替换 Kubelet、kube-proxy 组件。

重装滚动升级执行原理

基于重装的节点升级采用滚动升级的方式，同一时间只会对一个节点进行升级，只有当前节点升级成功才会进行下个节点的升级。

- 升级前检查：对节点上的 Pod 进行驱逐前的检查。具体的升级前检查项如下：
  - 统计该节点所有工作负载的 Pod 个数，若驱逐节点后，任何工作负载的 Pod 数目变为0 ，则检查不通过，不能进行升级。
  - 以下系统控制面工作负载将被忽略：
    - l7-lb-controller
    - cbs-provisioner
    - hpa-metrics-server
    - service-controller
    - cluster-autoscaler
- 驱逐 Pod：首先将节点标记为不可调度，随后驱逐或者删除节点上所有 Pod。
- 移出节点：将节点从集群中移除。该步骤只进行基本的清理工作，不会删除节点在集群中的 Node 实例，所以节点的 label、taint 等属性都可保留。
- 重装节点：重装节点的操作系统，并重新安装新版本 kubelet。
- 升级后检查：检查节点是否 ready，是否为可调度的，并检查当前不可用 Pod 比例是否超过最大值。

重装滚动升级操作步骤（Node Kubernetes 版本）

1. 登录容器服务控制台，选择左侧导航栏中的 集群。
2. 在集群管理页面，选择所需升级集群操作列的【升级节点 Kubernetes版本】，进入升级节点页面。
3. 选择升级方式为【重装滚动升级】。
4. 仔细阅读完升级须知后，单击【我已阅读并同意上述技术条款】，并单击【下一步】。
5. 在节点选择步骤中，选择本批次需要升级的节点，单击【下一步】。
6. 在升级配置步骤中，配置并确认升级信息，单击【完成】即可开始升级。
7. 查看节点升级进度， 直至所有节点升级完成。

原地滚动升级执行原理

节点原地升级采用滚动升级的方式，同一时间只会对一个节点进行升级，只有当前节点升级成功才会进行下个节点的升级。
步骤描述如下：

- 组件更新：替换和重启节点上的 kubelet 和 kube-proxy 组件。
- 升级后检查：检查节点是否 ready，并检查当前不可用 Pod 比例是否超过最大值。

原地滚动升级操作步骤

1. 登录容器服务控制台，选择左侧导航栏中的 集群。
2. 在集群管理页面，选择所需升级集群操作列的【升级节点 Kubernetes版本】，进入升级节点页面。
3. 选择升级方式为【原地滚动升级】以及目标版本。
4. 仔细阅读完升级须知后，单击【我已阅读并同意上述技术条款】，并单击【下一步】。

5. 在节点选择步骤中，选择本批次需要升级的节点，单击【下一步】。
6. 在升级配置步骤中，确认信息并单击【完成】即可开始升级。
7. 查看节点升级进度， 直至所有节点升级完成。

集群启用IPVS

操作场景

默认情况下，Kube-proxy 使用 iptables 来实现 Service 到 Pod 之间的负载均衡。TKE 支持快速开启基于 IPVS 来承接流量并实现负载均衡。开启 IPVS 适用于大规模集群，可提供更好的可扩展性和性能。

注意事项

- 本功能仅在创建集群时开启，暂不支持对存量集群的修改。
- IPVS 开启针对全集群生效，建议不要手动修改集群内 IPVS 和 Iptables 混用。
- 集群开启 IPVS 后不可关闭。
- IPVS 仅针对 Kubernetes 1.10及以上版本的 TKE 集群生效。

操作步骤

1. 登录 TKE 控制台。
2. 参考创建集群，在【集群信息】页面中，将 “Kubernetes版本” 设置为高于1.10的 Kubernetes 版本，并单击【高级设置】，开启【ipvs 支持】。
3. 按照页面提示逐步操作，完成集群的创建。

集群启用GPU调度

操作场景

如果您的业务需要进行深度学习、高性能计算等场景，您可以使用容器服务支持 GPU 功能，通过该功能可以帮助您快速使用 GPU 容器。如需要使用 GPU 功能，需要有可以使用的 GPU CVM 资源。
启用 GPU 调度有以下两种方式：

- 在集群中添加 GPU 节点
  - 新建 GPU 云服务器
  - 添加已有 GPU 云服务器
- 创建 GPU 服务的容器
  - 通过控制台方式创建
  - 通过应用或 Kubectl 命令创建

前提条件

已登录 TKE 控制台。

注意事项

- 仅在集群 Kubernetes 版本大于1.8.*时，支持使用 GPU 调度。
- 容器之间不共享 GPU，每个容器均可以请求一个或多个 GPU。无法请求 GPU 的一小部分。
- 建议搭配亲和性调度来使用 GPU 功能。

操作步骤

在集群中添加 GPU 节点

添加 GPU 节点有以下两种方法：

- 新建 GPU 云服务器
- 添加已有 GPU 云服务器

新建 GPU 云服务器

1. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
2. 在需要创建 GPU 云服务器的集群行中，单击【新建节点】。

3. 在“选择机型”页面，将【实例族】设置为 “GPU机型”，并选择 GPU 计算型的实例类型。
4. 按照页面提示逐步操作，完成创建。

> 在进行 “云主机配置” 时，TKE 将自动根据选择的机型进行 GPU 的驱动安装等初始流程，您无需关心基础镜像。

添加已有 GPU 云服务器

1. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
2. 在需要添加已有 GPU 云服务器的集群行中，单击【添加已有节点】。
3. 在“选择节点”页面，勾选已有的 GPU 节点，单击【下一步】。
4. 按照页面提示逐步操作，完成添加。

> 在进行 “云主机配置” 时，TKE 将自动根据选择的机型进行 GPU 的驱动安装等初始流程，您无需关心基础镜像。

创建 GPU 服务的容器

创建 GPU 服务的容器有以下两种方法：

- 通过控制台方式创建
- 通过应用或 Kubectl 命令创建

通过控制台方式创建

1. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
2. 单击需要创建 Workload 的集群【ID/名称】，进入待创建 Workload 的集群管理页面。
3. 在 “工作负载” 下，任意选择 Workload 类型，进入对应的信息页面。例如，选择 【工作负载】 > 【DaemonSet】，进入 DaemonSet 信息页面。
4. 单击【新建】，进入“新建Workload”页面。
5. 根据页面信息，设置工作负载名、命名空间等信息。并在 “GPU限制” 中，设置 GPU 限制的数量。
6. 单击【创建Workload】，完成创建。

通过应用或 Kubectl 命令创建

您可以通过应用或 Kubectl 命令创建，在 YAML 文件中添加 GPU 字段。

模板内容

模板可以通过从UI导入服务或新增空服务并手动编写来创建多个服务的YAML描述，详情可查看 应用模板操作指引

服务名	操作
[redacted]	删除

内容

```
    metadata:
      creationTimestamp: null
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: test
        resources:
          limits:
            cpu: 500m
            memory: 1Gi
            nvidia.com/gpu: "1"
          requests:
            cpu: 250m
            memory: 256Mi
        securityContext:
          privileged: false
      serviceAccountName: ""
      volumes: null
```

新增空服务 从UI导入服务

选择集群规格

TKE 集群的可用性与集群 Pod、ConfigMap、CRD、Event 等资源的数量、以及各类资源的 Get/List 读操作 QPS、Patch/Delete/Create/Update 等写操作 QPS 密切相关，应尽量避免对资源数量较多的集群发起类 List 操作，避免把 TKE 集群当数据库使用，写入过多的 ConfigMap/CRD/EndPoints 等，影响集群可用性。
常见的类 List 操作如下（以 Pod 资源为例）：

- 带标签查询

  ```
  kubectl get pod -l app=nginx
  ```

- 指定 namespace 查询

  ```
  kubectl get pod -n default
  ```

- 查询整个集群的 pod 等

  ```
  kubectl get pod --all-namespaces
  ```

- 通过 client-go 发起的 List 请求

  ```
  k8sClient.CoreV1().Pods("").List(metav1.ListOptions{})
  ```

如您有类似查询集群全量资源的需求，建议使用 K8s 的 informer 机制通过本地 cache 查询。对于一些简单的场景，可以通过在 List 请求中增加 ResourceVersion 参数，在 kube-apiserver cache 中查询，如 `k8sClient.CoreV1().Pods("").List(metav1.ListOptions{ResourceVersion: "0"})`。需注意，即使从 kube-apiserver cache 查询，如果对大量资源频繁发起 List 请求，仍会对 kube-apiserver 内存造成较大压力，仅建议在请求频率较低时使用该方式。

推荐配置

请在选购集群时参考如下推荐配置，根据业务实际情况选择合适的集群规格，以免集群控制面组件负载过大导致集群不可用。
例如，如果您计划在一个集群中部署50个节点，但是计划部署2000个 Pod, 则应该选用最大管理节点规模为100（而非50）的集群规格。

> 说明
>
> - 节点指 Kubernetes Node, 包含 CVM 节点**。**

- Pod 包括所有 Namespace 下，所有状态的 Pod，但不包括系统组件相关 Pod（cni-agent 等）。
- CRD 不统计 TKE 系统组件产生的 CRD（如 VPC-CNI 相关组件产生的 CRD）。
- 最大其他 K8s 资源数量指除表格中单独列出的资源外，其他的 K8s 资源的最大数量。例如您购买了最大管理节点规模为L100的集群，为了保障集群的可用性，集群中的 ClusterRole、Service、Endpoint 等 K8s 资源的数量均不应该超过2500。
- 建议每种资源类型的所有对象总和不应超过800MiB，每个资源对象大小不超过100KB。

集群规格	最大管理节点数量	最大 Pod 数量（推荐）	最大 replicasets 数量	最大 ConfigMap 数量（推荐）	最大 CRD 数量 / 最大其他 K8s 资源数量（推荐）
L5	5	150	900	128	150
L20	20	600	3600	256	600
L50	50	1500	9000	512	1250
L100	100	3000	18000	1024	2500
L200	200	6000	36000	2048	5000
L500	500	15000	90000	4096	10000
L3000	3000	90000	540000	8192	50000
L5000	5000	150000	900000	10240	100000

CVM容器集群网络

集群网络与容器网络是集群的基本属性。通过设置集群网络和容器网络可以规划集群的网络划分。

- 集群网络：为集群内的主机分配在节点网络地址范围内的 IP 地址，您可以选择私有网络中的子网用于集群的节点网络。更多私有网络的介绍可参看 私有网络和子网 。
- 容器网络：为集群内的容器分配在容器网络地址范围内的 IP 地址，您可以自定义三大私有网段作为容器网络，根据您选择的集群内服务数量的上限，自动分配适当大小的 CIDR 段用于 kubernetes service；也可以根据您选择的每个节点的pod数量上限，自动为集群内每台云服务器分配一个适当大小的网段用于该主机分配 Pod 的 IP 地址。

集群网络与容器网络的关系

- 集群网络和容器网络网段不能重叠。
- 同一 VPC 内，不同集群的容器网络网段不能重叠。
- 容器网络和 VPC 路由重叠时，优先在容器网络内转发。

集群网络与TCloudFinanceZone其他资源通信

- 集群内容器与容器之间互通。
- 集群内容器与节点直接互通。
- 集群内容器与 云存储 redis、云数据库 Memcached 等资源同一 VPC 下内网互通。
- 设置同地域集群间互通。
- 设置跨地域集群间互通。
- 设置容器集群与 IDC 互通。

容器网络说明

- 容器 CIDR：集群内 Sevice、 Pod 等资源所在网段。

- Services 数量上限/集群：决定分配给 Sevice 的 CIDR 大小。

  > 容器服务 TKE 集群默认创建3个 Sevice：kubernetes、hpa-metrics-serviceube-dns，kube-dns，同时还有2个广播地址和网络号，因此用户可以使用的Services 数量上限/集群是 serviceMax - 5。

- Pod 数量上限/Node：决定分配给每个 Node 的 CIDR 的大小。 > 容器服务 TKE 集群默认创建2个 kube-dns 的 Pod 和1个 l7-lb-controller 的 Pod。
  对于一个 Node 上的 Pod，有三个地址不能分配分别是：网络号，广播地址和网关地址，因此 Node 最大的 Pod 数目 = podMax - 3。

设置同地域集群间互通

操作场景

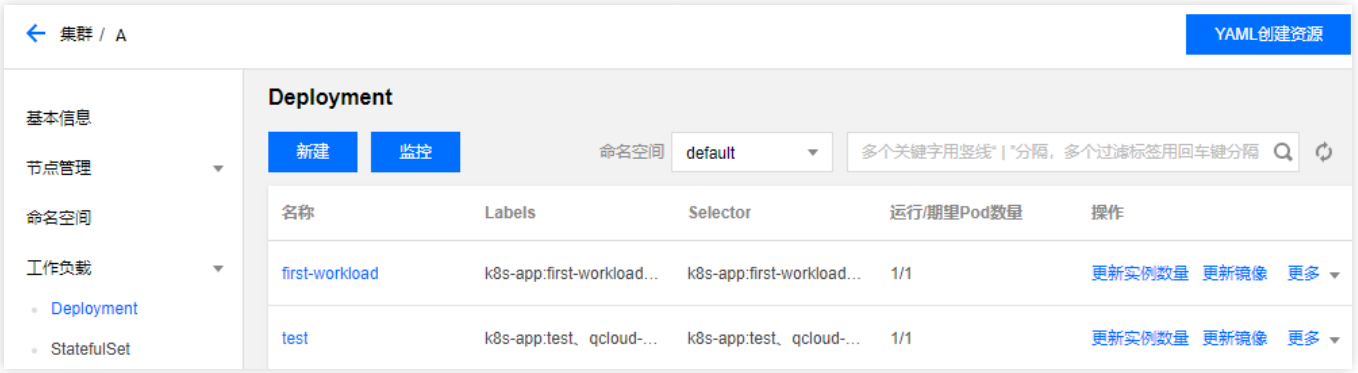
对等连接（Peering Connection）是一种大带宽、高质量的云上资源互通服务，可以打通云上的资源通信链路。您可以通过对等链接实现同地域不同 VPC 下的间的集群互通。关于如何建立对等连接，您可以参考 创建对等连接。

> - 本文档以已创建集群并已添加节点为例。关于如何创建集群，您可以参考 创建集群 进行创建。
> - 请先确保对等连接间成功建立，子机间能互通。若对等连接建立有问题，请排查控制台路由表项、CVM 安全组、子网 ACL 的设置是否有问题。
> - 暂时仅支持同地域对等连接间容器互通。若需要跨地域间容器互通，请提交工单咨询。

操作步骤

获取容器的基本信息

1. 登录TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要设置同地域集群间互通的集群【ID/名称】，进入该集群的管理页面。如下图所示：
   例如，进入 A 集群的管理页面。



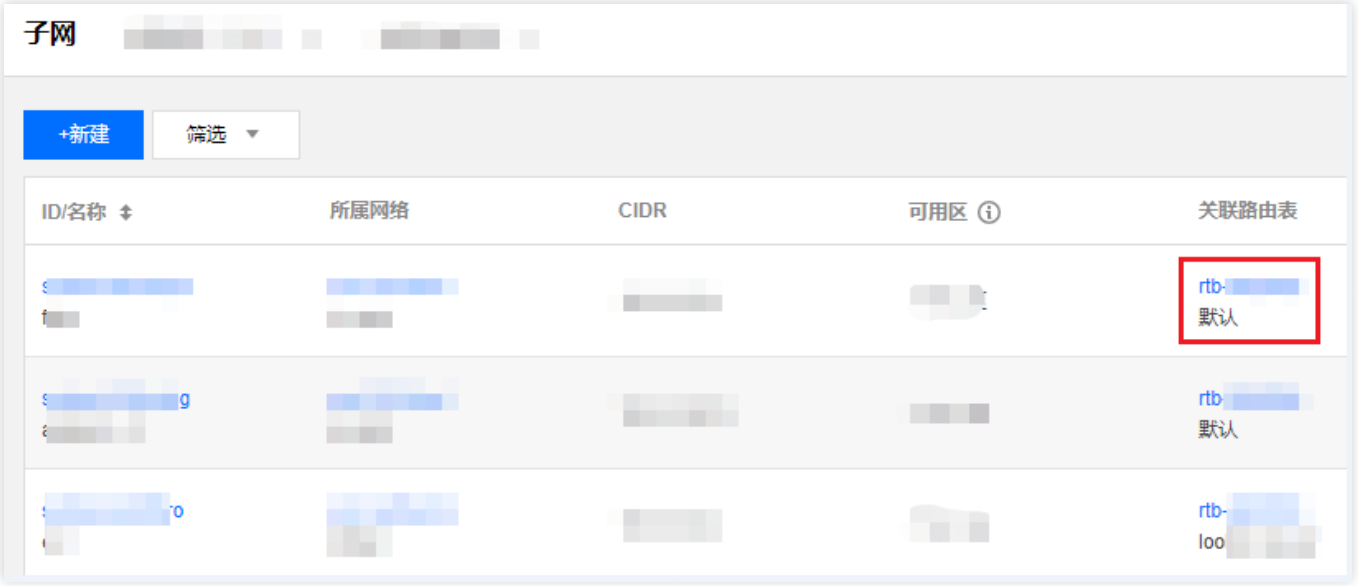
4. 在左侧导航栏中，选择【基本信息】，进入“基本信息”页面。
5. 记录 “基本信息” 中 “容器网络” 的网段和掩码。
6. 重复执行 步骤3 - 步骤5，记录另一个集群容器网络的网段和掩码。
   例如，记录 B 集群容器网络的网段和掩码。

配置路由表

1. 切换至私有网络控制台。

2. 在左侧导航栏中，单击【子网】，进入“子网”页面。

3. 单击对等连接本端指定子网的关联路由表。



4. 在关联路由表的基本信息页面，单击【新增路由策略】。

5. 在弹出的【新增路由】窗口中，设置路由信息。主要参数信息如下：

   - 目的端：输入 B 集群容器的网段。
   - 下一跳类型：选择“对等连接”。
   - 下一跳：选择已建立的对等连接。

6. 单击【创建】，完成本端路由表的配置。

7. 重复执行 步骤2 - 步骤6，完成对端路由表的配置。

预期结果

容器之间可以互通，容器的登录方法请参考 远程终端基本操作。

1. 登录集群 A 的容器，并在集群 A 的容器中访问集群 B 的容器。如下图所示：

```
[root@centos-ssh-8456f58d49-hv9k2 /]# ping 172.31.0.6
PING 172.31.0.6 (172.31.0.6) 56(84) bytes of data.
64 bytes from 172.31.0.6: icmp_seq=1 ttl=62 time=1.47 ms
64 bytes from 172.31.0.6: icmp_seq=2 ttl=62 time=1.29 ms
64 bytes from 172.31.0.6: icmp_seq=3 ttl=62 time=1.44 ms
```

2. 登录集群 B 的容器，并在集群 B 的容器中访问集群 A 的容器。如下图所示：

```
[root@centos-d999ccdd6-z42t4 /]# ping 192.168.0.11
PING 192.168.0.11 (192.168.0.11) 56(84) bytes of data.
64 bytes from 192.168.0.11: icmp_seq=1 ttl=62 time=1.40 ms
64 bytes from 192.168.0.11: icmp_seq=2 ttl=62 time=1.36 ms
64 bytes from 192.168.0.11: icmp_seq=3 ttl=62 time=1.41 ms
```

设置跨地域集群间互通

操作场景

对等连接（Peering Connection）是一种大带宽、高质量的云上资源互通服务，可以打通TCloudFinanceZone上的资源通信链路。关于如何建立对等连接，您可以参考 创建对等连接。您可以通过对等链接实现跨地域不同集群互通。

> - 本文档以已创建集群并已添加节点为例。关于如何创建集群，您可以参考 创建集群 进行创建。
> - 请先确保对等连接成功建立，子机间能互通。若对等连接建立有问题，请排查控制台路由表项、CVM 安全组、子网 ACL 的设置是否有问题。

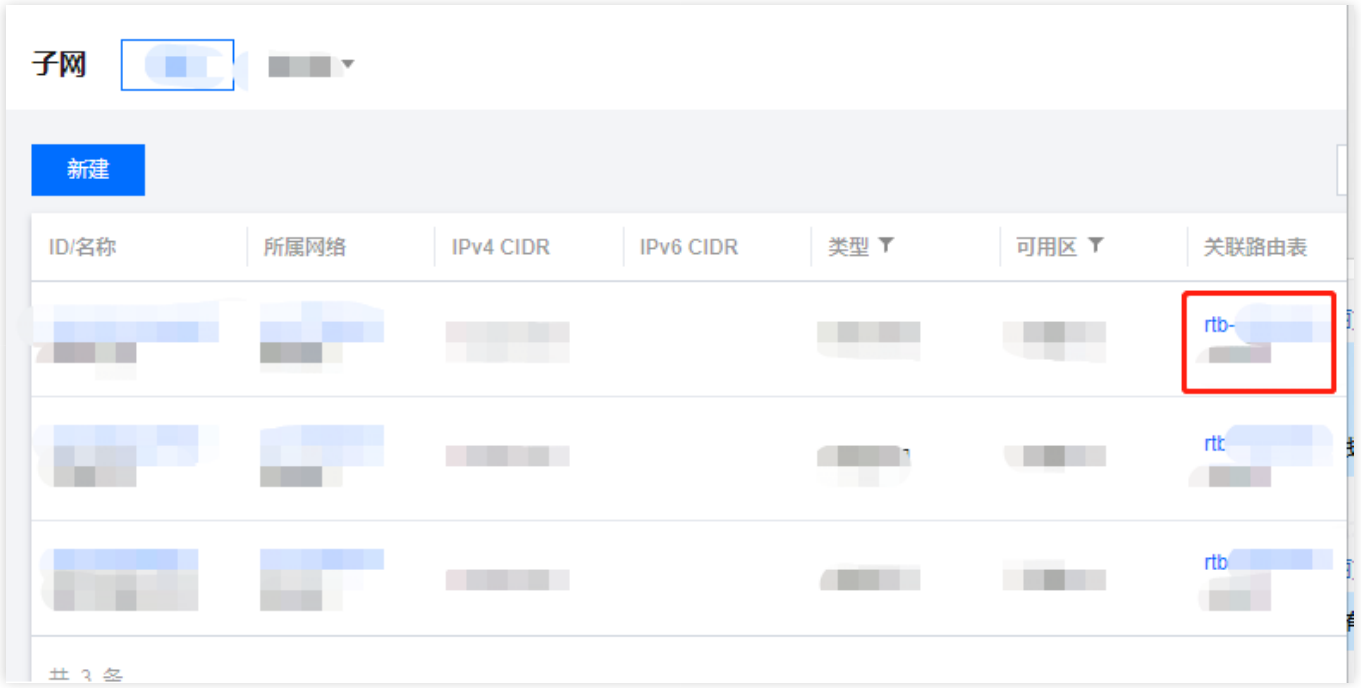
操作步骤

获取容器的基本信息

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要设置跨地域不同集群间互通的集群【ID/名称】，进入该集群的管理页面。
4. 在左侧导航栏中，选择【基本信息】，进入“基本信息”页面。
5. 记录 “基本信息” 中 “所在地域”、“节点网络” 和 “容器网络” 的信息。
6. 重复执行 步骤3 - 步骤5，记录另一个集群容器 “所在地域”、“节点网络” 和 “容器网络” 的信息。
   例如，记录 B 集群容器 “所在地域”、“节点网络” 和 “容器网络” 的信息。

配置路由表

1. 切换至私有网络控制台。
2. 在左侧导航栏中，单击【对等连接】，进入对等连接管理页面，并记录对等连接的【ID/名称】。
3. 在左侧导航栏中，单击【子网】，进入“子网”管理页面。
4. 单击对等连接本端指定子网的关联路由表。



5. 在关联路由表的“默认详情”页面，单击【新增路由策略】。
6. 在弹出的“新增路由”窗口中，设置路由信息。主要参数信息如下：
    - 目的端：输入 B 集群容器的网段。
    - 下一跳类型：选择 “对等连接”。
    - 下一跳：选择已建立的对等连接。
7. 单击【创建】，完成本端路由表的配置。
8. 重复执行 步骤3 - 步骤7，完成对端路由表的配置。

预期结果

容器之间可以互通，容器的登录方法请参考 远程终端基本操作。

1. 登录集群 A 的容器，并在集群 A 的容器中访问集群 B 的容器。如下图所示：

```
选中文字进行复制，按下Shift+Insert进行粘贴
[root@centos-sh-65d4dc775-csjd5 /]# ping 172.31.2.7
PING 172.31.2.7 (172.31.2.7) 56(84) bytes of data.
64 bytes from 172.31.2.7: icmp_seq=1 ttl=60 time=28.9 ms
64 bytes from 172.31.2.7: icmp_seq=2 ttl=60 time=28.7 ms
64 bytes from 172.31.2.7: icmp_seq=3 ttl=60 time=28.7 ms
64 bytes from 172.31.2.7: icmp_seq=4 ttl=60 time=28.8 ms
64 bytes from 172.31.2.7: icmp_seq=5 ttl=60 time=28.7 ms
^C
--- 172.31.2.7 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 28.706/28.810/28.953/0.202 ms
[root@centos-sh-65d4dc775-csjd5 /]#
```

2. 登录集群 B 的容器，并在集群 B 的容器中访问集群 A 的容器。如下图所示：

```
[root@centos-bj-bdcd88f45-w9tgz /]# ping 10.110.1.4
PING 10.110.1.4 (10.110.1.4) 56(84) bytes of data.
64 bytes from 10.110.1.4: icmp_seq=1 ttl=60 time=35.0 ms
64 bytes from 10.110.1.4: icmp_seq=2 ttl=60 time=35.0 ms
64 bytes from 10.110.1.4: icmp_seq=3 ttl=60 time=35.0 ms
64 bytes from 10.110.1.4: icmp_seq=4 ttl=60 time=35.0 ms
^C
--- 10.110.1.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 35.010/35.045/35.082/0.033 ms
[root@centos-bj-bdcd88f45-w9tgz /]#
```

设置容器集群与IDC互通

操作场景

目前容器集群与用户 IDC 互通主要通过两种方式：专线和 IPsec VPN。

> - 本文档以已创建集群并已添加节点为例。关于如何创建集群，您可以参考 创建集群 进行创建。
> - 请先确保容器服务所在的 VPC 和您 IDC 机房已通过专线或 VPN 成功连接。若通道未连接，您可以参考 VPN 通道未连通如何处理？。

操作步骤

通过专线方式互通

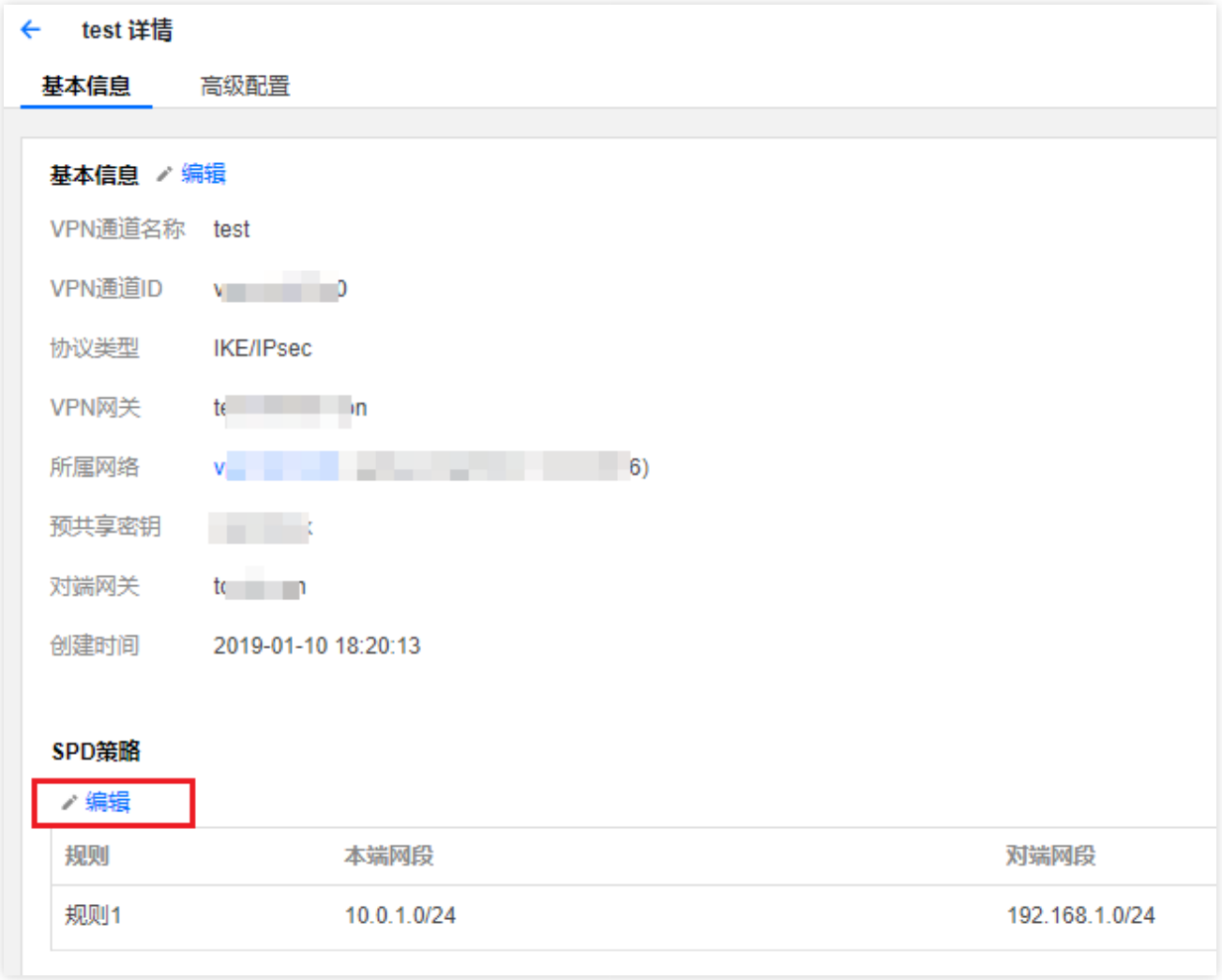
1. 参考 申请物理专线，申请物理专线。
2. 参考 申请通道，申请通道。
3. 参考 创建专线网关，创建专线网关。
4. 验证容器节点与 IDC 互通。 > 说明：
   执行此步骤时，请保证容器节点与 IDC 互通，验证通过。
5. 准备地域，appID，集群 ID，vpcID，专线网关 ID 信息，联系运维工程师协助配置打通容器网络。
6. 根据 IDC 使用的协议类型，选择操作方式。
   - 若 IDC 使用的是 BGP 协议，容器网段路由将自动同步。
   - 若是其他协议，需在 IDC 内配置访问容器网段下一跳路由到专线网关。
7. 验证容器与与 IDC 互通。

通过 VPN 方式互通

配置 SPD 策略

1. 登录私有网络控制台。
2. 在左侧导航栏中，单击【VPN链接】>【VPN通道】，进入“VPN 通道”管理页面。
3. 单击需要配置 SPD 策略的本端 VPN 通道的 ID/名称。

4. 在 VPN 通道的详情页面，单击【SPD策略】栏下的【编辑】，添加容器网段。



5. 单击【保存】。
6. 重复执行 步骤3 - 步骤5，配置对端 VPN 通道的 SPD 策略。

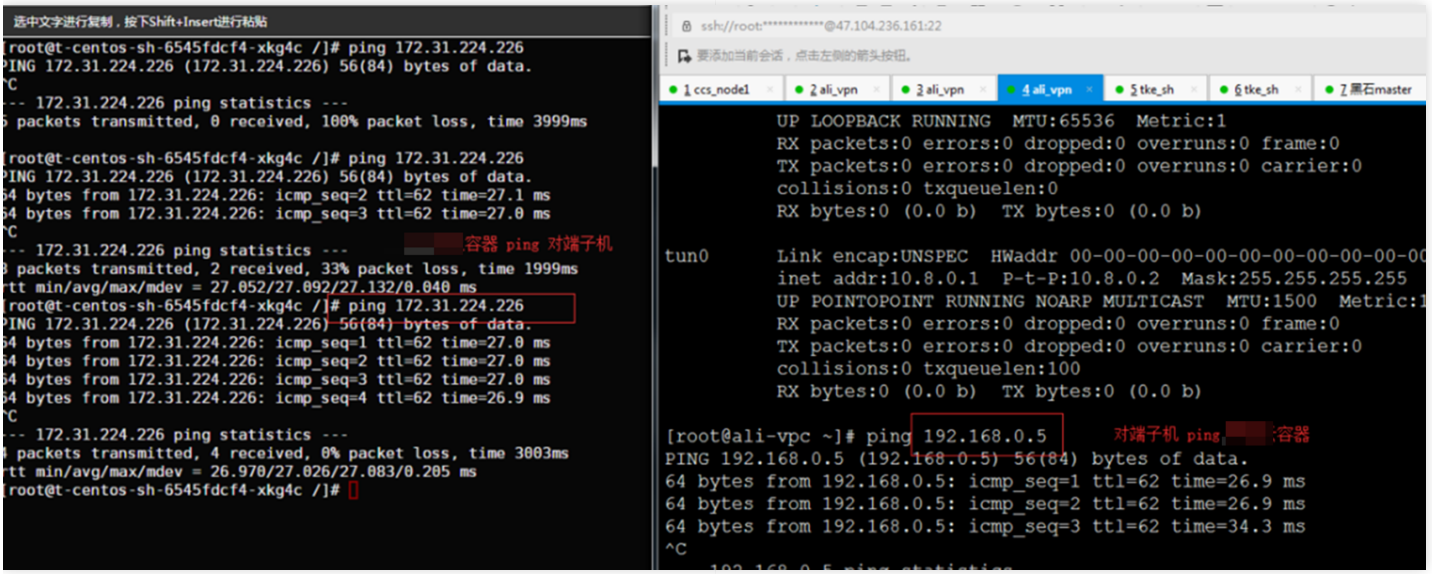
添加容器网段

> 说明：
> 一个子网只能绑定一个路由表，若关联多个路由表，将被替换成最后一个绑定的路由表。

1. 在左侧导航栏中，单击【路由表】，进入路由表管理页面。
2. 找到 设置同地域集群间互通 或者 设置跨地域集群间互通 时配置的路由表，单击该路由表的 ID/名称，进入路由表的详情页面。
3. 单击【新增路由策略】，追加容器网段。
4. 选择【关联子网】页签，单击【新增子网】，关联子机所在的子网。
5. 重复执行 步骤2 - 步骤4，在您对端的路由设备上，添加TCloudFinanceZone容器所在网段。

预期结果

容器和对端子机可以互通。



容器间与 VPN 对端子机已经实现互通。

> 说明：
> 如需云上容器与 IDC 机房通过 IPsec VPN 互通，主要需要设置 SPD策略和路由表。

设置安全组服务

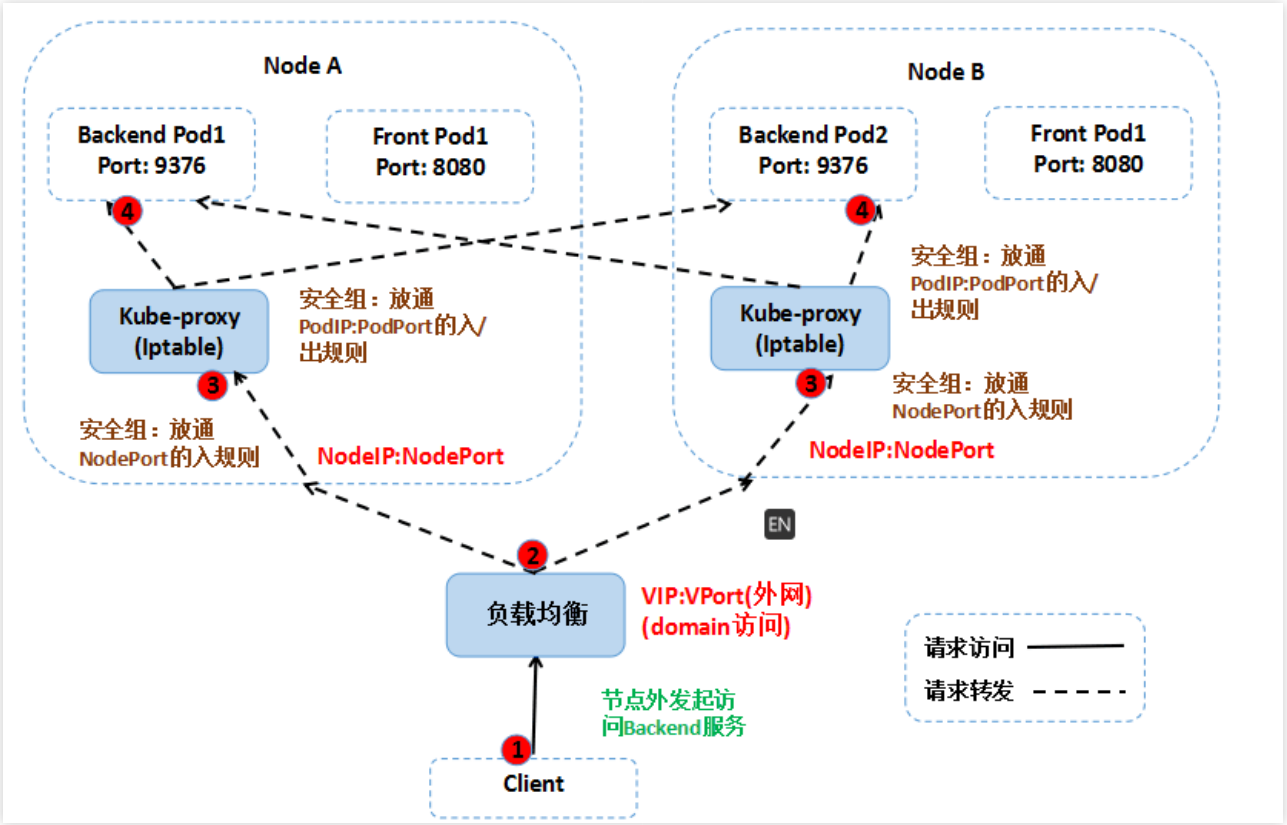
安全问题向来是一个大家非常关注的问题，云平台将安全性作为产品设计中的最高原则，严格要求产品做到安全隔离，容器服务同样非常看重这一点。云平台的基础网络可以提供充分的安全保障，容器服务选择了网络特性更丰富的云私有网络VPC来作为容器服务的底层网络，本文档主要介绍容器服务下使用安全组的最佳实践，帮助大家选择安全组策略。

安全组

安全组是一种有状态的包过滤功能的虚拟防火墙，它用于设置单台或多台云服务器的网络访问控制，是云平台提供的重要的网络安全隔离手段。更多安全组的介绍请参见安全组。

容器服务选择安全组的原则

- 由于在容器集群中，服务实例采用分布式的方式进行部署，不同的服务实例分布在集群的节点上。建议同一个集群下的主机绑定同一个安全组，集群的安全组不添加其他云服务器。
- 安全组只对外开放最小权限。
- 需放通以下容器服务使用规则：
  - 放通容器实例网络和集群节点网络
    当服务访问到达主机节点后，会通过 Kube-proxy 模块设置的 iptable 规则将请求进行转发到服务的任意一个实例。由于服务的实例有可能在另外的节点上，这时会出现跨节点访问。例如访问的目的 IP有服务实例IP、集群中其它的节点 IP、节点上集群 cbr0 网桥的 IP。这就需要在对端节点上放通容器实例网络和集群节点网络访问。
  - 同一 VPC 不同集群互访的情况，需要放通对应集群的容器网络和节点网络。
  - 需要 SSH 登录节点的放通22端口。
  - 放通节点30000 - 32768端口。
    在访问路径中，需要通过负载均衡器将数据包转发到容器集群的 NodeIP:NodePort 上。其中 NodeIP 为集群中任意一节点的主机 IP ，而 NodePort 是在创建服务时容器集群为服务默认分配的，NodePort 的范围为30000 - 32768。下图以外网访问服务为例：



容器服务默认安全组规则

节点默认安全组规则

集群节点间的正常通信需要放通部分端口，为避免绑定无效安全组造成客户创建集群失败，容器服务为您提供了默认安全组配置规则。如下表：

> 注意
>
> 若当前默认安全组不能满足业务需求，并且已创建绑定该安全组的集群时，您可参照管理安全组规则进行该集群安全组规则的查看、修改等操作。

入站规则

协议规则	端口	来源	策略	备注
ALL	ALL	容器网络 CIDR	允许	放通容器网络内 Pod 间通信
ALL	ALL	集群网络 CIDR	允许	放通集群网络内节点间通信
tcp	30000 - 32768	0.0.0.0/0	允许	放通 NodePort 访问（LoadBalancer 类型的 Service 需经过 NodePort 转发）

协议规则	端口	来源	策略	备注
udp	30000 - 32768	0.0.0.0/0	允许	放通 NodePort 访问（LoadBalancer 类型的 Service 需经过 NodePort 转发）
icmp	-	0.0.0.0/0	允许	放通 ICMP 协议，支持 Ping 操作

出站规则

协议规则	端口	来源	策略
ALL	ALL	0.0.0.0/0	允许

> 说明
>
> - 自定义出站规则时需放通节点网段和容器网段。
> - 容器节点配置该规则，可满足不同的访问方式访问集群中服务。
> - 集群中服务的访问方式，可参考 Service 管理 服务访问方式。

独立集群 Master 默认安全组规则

创建独立集群时，会默认为 Master 机型绑定 TKE 默认安全组，降低集群创建后 Master 与 Node 无法正常通信及 Service 无法正常访问的风险。默认安全组配置规则如下表：

> 说明
>
> 创建安全组的权限继承至 TKE 服务角色，详情请参见 服务授权相关角色权限说明。

入站规则

协议	端口	网段	策略	备注
ICMP	ALL	0.0.0.0/0	允许	支持 Ping 操作
TCP	30000 - 32768	集群网络 CIDR	允许	放通 NodePort 访问（LoadBalancer 类型的 Service 需经过 NodePort 转发）
UDP	30000 - 32768	集群网络 CIDR	允许	放通 NodePort 访问（LoadBalancer 类型的 Service 需经过 NodePort 转发）
TCP	60001,60002,10250,2380,2379,53,17443,50055,443,61678	集群网络 CIDR	允许	放通 API Server 通信
TCP	60001,60002,10250,2380,2379,53,17443	容器网络 CIDR	允许	放通 API Server 通信
TCP	30000 - 32768	容器网络 CIDR	允许	放通 NodePort 访问（LoadBalancer 类型的 Service 需经过

协议	端口	网段	策略	备注
				NodePort 转发）
UDP	30000 - 32768	容器网络 CIDR	允许	放通 NodePort 访问（LoadBalancer 类型的 Service 需经过 NodePort 转发）
UDP	53	容器网络 CIDR	允许	放通 CoreDNS 通信
UDP	53	集群网络 CIDR	允许	放通 CoreDNS 通信

出站规则

协议规则	端口	来源	策略
ALL	ALL	0.0.0.0/0	允许

自定义 Kubernetes 组件启动参数

操作场景

为方便对容器服务 TKE 集群中的 Kubernetes 组件参数进行设置与管理，我们开发了自定义 Kubernetes 组件参数功能。本文将介绍在集群中如何设置自定义 Kubernetes 组件参数。

注意事项

- 升级 Kubernetes 集群版本，由于 Kubernetes 跨版本后启动参数可能存在不兼容的情况，大版本升级不会保留您原集群版本的自定义 Kubernetes 组件参数，您需要重新设置自定义的 Kubernetes 的组件参数。

操作说明

创建集群设置自定义 Kubernetes 组件参数

1. 登录容器服务控制台，单击左侧导航栏中的【集群】。
2. 在“集群管理”页面，单击集群列表上方的【新建】。
   在“创建集群”页面，选择【高级设置】在展示的页面中可以自定义 Kubernetes 组件参数。

设置节点的自定义 Kubelet 参数

在“新建集群节点”页面、“添加已有节点”页面、“新增节点池”页面及“新增节点”页面均可设置节点的自定义 Kubelet 参数。

集群升级设置自定义 Kubernetes 组件参数

1. 登录容器服务控制台，选择左侧导航栏中的【集群】。
2. 在“集群管理”页面，选择需进行 Master Kubernetes 版本升级的集群 ID，进入集群详情页。
3. 在集群基本信息中，单击 Kubernetes 版本右侧的升级。同时设置 Kubernetes 组件启动参数。

镜像仓库
镜像仓库概述

镜像仓库概述

镜像仓库用于存放 Docker 镜像，Docker 镜像用于部署容器服务，每个镜像有特定的唯一标识（镜像的 Registry 地址+镜像名称+镜像 Tag）。

镜像类型

目前镜像支持公共镜像和用户私有镜像。

使用帮助

- 镜像仓库基本操作
- 如何搭建私有镜像仓库

镜像仓库基本教程

操作场景

镜像仓库用于存放 Docker 镜像，Docker 镜像用于部署容器服务，每个镜像有特定的唯一标识（镜像的 Registry 地址+镜像名称+镜像 Tag）。镜像支持 Docker Hub 官方镜像和用户私有镜像。

操作步骤

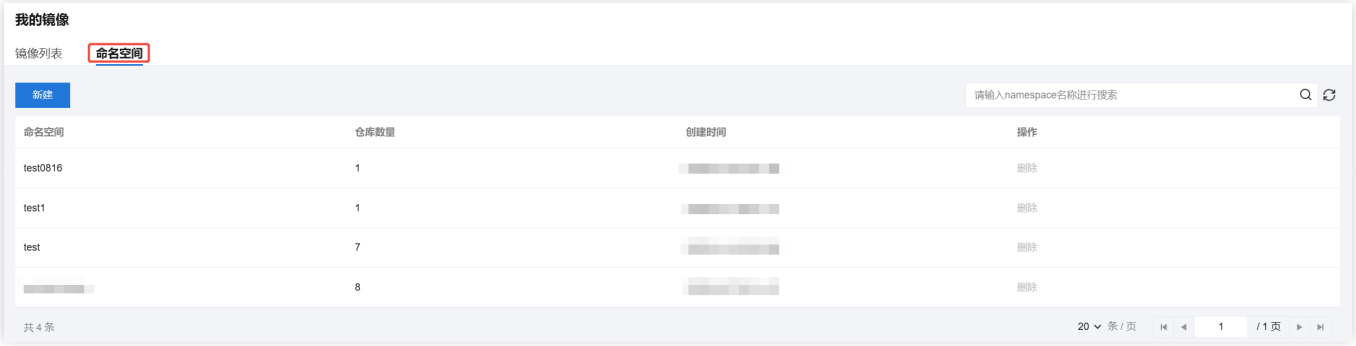
开通镜像仓库

> 说明：
> 首次使用镜像仓库的用户，需要先开通镜像仓库。

1. 登录TKE 控制台。
2. 选择左侧导航栏中的【镜像仓库】>【我的镜像】。
3. 根据以下提示填写相关信息后单击【开通】进行初始化。
    - 用户名：默认是当前用户的账号，是您登录到云平台Docker 镜像仓库的身份。
    - 密码：是您登录到云平台 Docker 镜像仓库的凭证。

创建命名空间

1. 选择左侧导航栏中的【镜像仓库】>【我的镜像】，进入“我的镜像”页面。
2. 在“我的镜像”页面中，选择【命名空间】页签并单击【新建】。

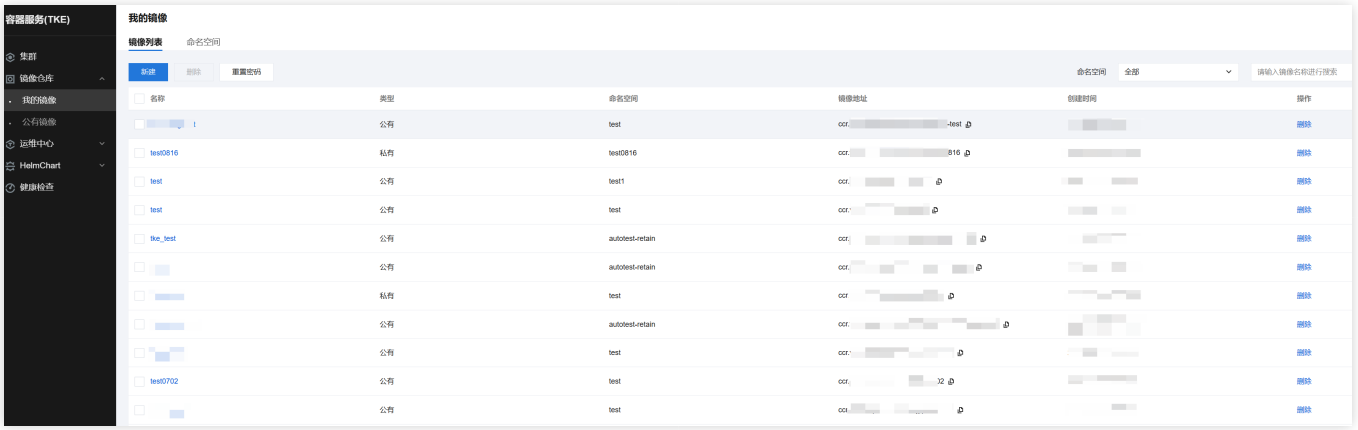


3. 在弹出的【新建命名空间】窗口中，输入命名空间名并单击【提交】。

> 说明：
> 命名空间名称全局唯一，若您希望使用的命名空间名称已被其他用户使用，请尝试其他适用的命名空间名称。

创建镜像

1. 选择左侧导航栏中的【镜像仓库】>【我的镜像】，进入“我的镜像”页面。
2. 在“我的镜像”页面，单击镜像列表页上方的【新建】。



3. 输入镜像名称和描述，然后【提交】。

> 说明：
> 命名空间将用于分类容器镜像，也是您创建的私人镜像地址的前缀，本文以 tkefiletest 为例。

推送镜像到镜像仓库

登录到TCloudFinanceZone registry

1. 在终端替换以下命令中的相关信息并执行，登录TCloudFinanceZone registry。

```
$ sudo docker login --username=[username] ccr.finance.cloud.tencent.com
```

username：TCloudFinanceZone账号，开通时已注册。
2. 输入密码后即登录完成。

上传镜像

根据以下提示替换命令中的相关信息并执行，上传镜像。

```
$ sudo docker tag [ImageId] ccr.finance.cloud.tencent.com/[namespace]/[ImageName]:[镜像版本号]
$ sudo docker push ccr.finance.cloud.tencent.com/[namespace]/[ImageName]:[镜像版本号]
```

- ImageId 和镜像版本号：根据已有镜像信息补充。
- namespace：开通镜像仓库时填写的命名空间。
- ImageName：在控制台创建的镜像名称。

下载镜像

1. 执行以下命令登录到镜像仓库，需输入在开通镜像仓库中已设置的密码。

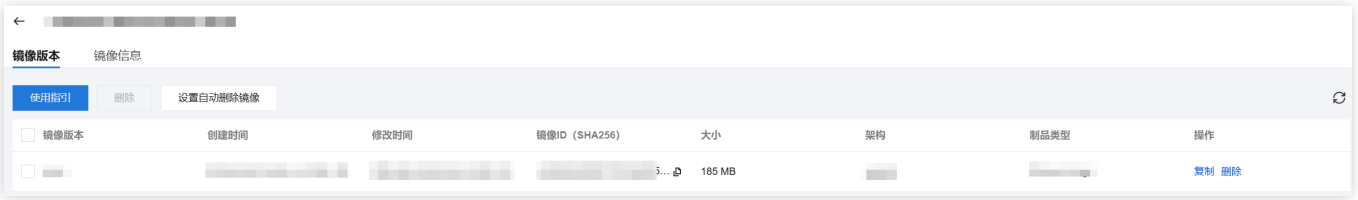
```
$ sudo docker login --username=[username] ccr.finance.cloud.tencent.com
```

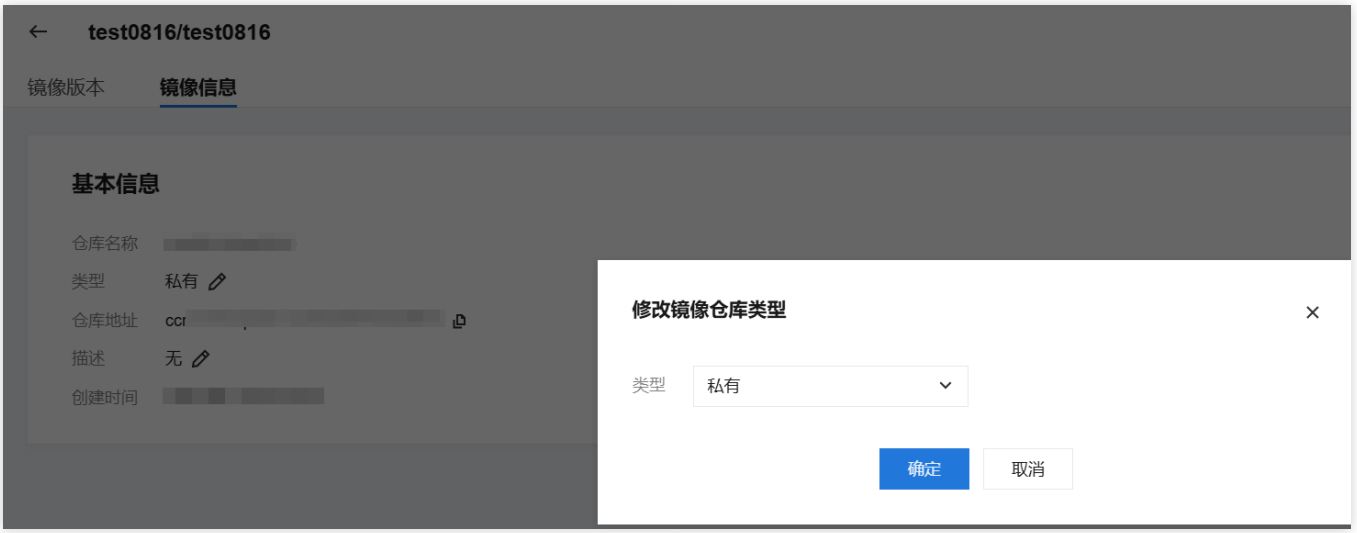
2. 替换命令中的相关信息并执行，下载镜像。

```
$ sudo docker pull ccr.finance.cloud.tencent.com/[namespace]/[ImageName]:[镜像版本号]
```

管理镜像

1. 选择左侧导航栏中的【镜像仓库】>【我的镜像】，进入“我的镜像”页面。
2. 单击需配置的镜像名称，进入详情页后，根据需要对镜像进行操作。
   在镜像版本页签可对镜像版本进行管理；
   在镜像信息页签可对镜像类型和描述进行编辑。





删除镜像

1. 选择左侧导航栏中的【镜像仓库】>【我的镜像】，进入“我的镜像”页面。
2. 选择需删除镜像所在行右侧【删除】。
3. 在弹出的“删除镜像仓库”窗口中，单击【确定】即可删除该镜像所有版本。



> 说明：如果需要删除镜像的某个版本，可以通过单击镜像名称，进入“镜像版本”页面后，根据需要选择对应镜像版本进行删除操作。

如何搭建私有镜像仓库

本文档介绍如何通过 Docker Compose 搭建一个简单的 registry 环境。使用 DockerHub 官方镜像，registry 镜像版本 为 registry:2.5.0 ， Nginx 镜像版本为 nginx:1.11.5 。这里主要介绍 registry 环境的搭建及使用，更详细的企业级 registry 服务器的搭建可参阅开源的 Harbor。

registry 概述

registry 是 Docker 的镜像存储服务，DockerHub 上的 registry 镜像见 Registry 官方镜像，更多详细信息请转至 GitHub 查看最新源码 。

搭建 registry

1. 在服务器上执行如下命令安装 Docker，这里选择TCloudFinanceZone（Ubuntu Server 14.04.1 LTS 64位）镜像来创建服务器。

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```

2. 安装 Docker Compose。
   Docker Compose 是一个定义及运行多个 Docker 容器的工具。
   使用 Docker Compose 只需要在一个配置文件中定义多个 Docker 容器，再使用一条命令将多个容器启动，Docker Compose 会通过解析容器间的依赖关系，按先后顺序启动所定义的容器。有关 Docker Compose 详情请转至 GitHub 了解。

   ```
   curl -L https://github.com/docker/compose/releases/download/1.8.0/docker-compose-$(uname -s)-$(uname -m) > /usr/local/bin/docker-compose
   chmod a+x /usr/local/bin/docker-compose
   ```

3. 启动 registry 服务，此例中包含 Nginx 和 registry 两个容器，涉及的配置文件请参见附录。

   ```
   docker-compose up -d
   ```

   - 停止服务。

   ```
   docker-compose stop
   ```

   - 重启服务。

docker-compose restart

- 下线服务。

docker-compose down

镜像基本操作

上传镜像

1. 因为搭建的 registry 服务使用 HTTP 协议，所以 Docker 启动参数需要配置 --insecure-registry localhost 选项，修改 /etc/default/docker 文件如下：

   DOCKER_OPTS="--insecure-registry localhost"

2. 重启 Docker。

   service docker restart

3. 拉取上传镜像 docker pull；docker tag；docker push（tag 默认为 latest）。

   ```
   docker pull hello-world
   docker tag hello-world localhost/library/hello-world
   docker push localhost/library/hello-world
   ```

下载镜像

```
docker pull localhost/library/hello-world
```

删除镜像

```
docker rmi localhost/library/hello-world
```

获取镜像

获取镜像仓库列表

```
# curl http://localhost/v2/_catalog
{"repositories":["library/hello-world"]}
```

未上传镜像前的输出如下：

```
# curl http://localhost/v2/_catalog
{"repositories":[]}
```

获取镜像 tag 列表

```
# curl -X GET http://localhost/v2/library/hello-world/tags/list
{"name":"library/hello-world","tags":["latest"]}
```

获取镜像 manifests 信息

```
# curl -H "Accept: application/vnd.docker.distribution.manifest.v2+json" -X GET http://localhost/v2/library/hello-world/manifests/latest
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
   "config": {
      "mediaType": "application/vnd.docker.container.image.v1+json",
      "size": 1473,
      "digest": "sha256:c54a2cc56cbb2f04003c1cd4507e118af7c0d340fe7e2720f70976c4b75237dc"
   },
   "layers": [
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 974,
         "digest": "sha256:c04b14da8d1441880ed3fe6106fb2cc6fa1c9661846ac0266b8a5ec8edf37b7c"
      }
   ]
}
```

其中 c54a2cc56cbb2f04003c1cd4507e118af7c0d340fe7e2720f70976c4b75237dc 即为执行 docker images 时显示的 IMAGE ID。
layers 表示了镜像的层次关系，可以通过 layers 中的 digest 来拉取 blob，详情见下面获取镜像 blob。

获取镜像 blob

在上面获取 hello-world:latest 镜像的 manifests 信息中只有一个 layer，以此为例来说明如何获取镜像 blob。拉取的结果显示获取的 blob 与文件 sha256 是一致的。执行 docker pull 实际上就是先获取到镜像的 manifests 信息，再拉取 blob。

```
# curl -s -X GET http://localhost/v2/library/hello-world/blobs/sha256:c04b14da8d1441880ed3fe6106fb2cc6fa1c9661846ac0266b8a5ec8edf37b7c -o hello-world.blob
# ls -l hello-world.blob
-rw-r--r-- 1 root root 974 Nov 23 09:56 hello-world.blob
# sha256sum hello-world.blob
c04b14da8d1441880ed3fe6106fb2cc6fa1c9661846ac0266b8a5ec8edf37b7c  hello-world.blob
```

删除镜像

删除镜像（ soft delete ）

首先通过 curl -i 参数获取到镜像的 Docker-Content-Digest ，registry 2.3 版本及以后的版本必须在 header 中指定 Accept: application/vnd.docker.distribution.manifest.v2+json ，否则默认返回的是 schema1 的 digest，与 schema2 的 digest 不同，使用不指定上述头信息返回的 digest 删除时会返回 404。

```
# curl -i -H "Accept: application/vnd.docker.distribution.manifest.v2+json" -X GET http://localhost/v2/library/hello-world/manifests/latest

HTTP/1.1 200 OK
Server: nginx/1.11.5
Date: Wed, 23 Nov 2016 02:17:51 GMT
Content-Type: application/vnd.docker.distribution.manifest.v2+json
Content-Length: 524
Connection: keep-alive
Docker-Content-Digest: sha256:a18ed77532f6d6781500db650194e0f9396ba5f05f8b50d4046b294ae5f83aa4
Docker-Distribution-Api-Version: registry/2.0
Etag: "sha256:a18ed77532f6d6781500db650194e0f9396ba5f05f8b50d4046b294ae5f83aa4"

{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
   "config": {
      "mediaType": "application/vnd.docker.container.image.v1+json",
      "size": 1473,
      "digest": "sha256:c54a2cc56cbb2f04003c1cd4507e118af7c0d340fe7e2720f70976c4b75237dc"
   },
   "layers": [
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 974,
         "digest": "sha256:c04b14da8d1441880ed3fe6106fb2cc6fa1c9661846ac0266b8a5ec8edf37b7c"
      }
```

```
    ]
}
```

根据上一步返回的 Docker-Content-Digest 删除，返回 202 表示删除成功。

```
# curl -k -v -s -X DELETE http://localhost/v2/library/hello-world/manifests/sha256:a18ed77532f6d6781500db650194e0f9396ba5f05f8b50d4046b294ae5f83aa4

* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> DELETE /v2/library/hello-world/manifests/sha256:a18ed77532f6d6781500db650194e0f9396ba5f05f8b50d4046b294ae5f83aa4 HTTP/1.1
> User-Agent: curl/7.35.0
> Host: localhost
> Accept: */*
>
< **HTTP/1.1 202 Accepted**
* Server nginx/1.11.5 is not blacklisted
< Server: nginx/1.11.5
< Date: Wed, 23 Nov 2016 02:29:59 GMT
< Content-Type: text/plain; charset=utf-8
< Content-Length: 0
< Connection: keep-alive
< Docker-Distribution-Api-Version: registry/2.0
<
* Connection #0 to host localhost left intact
```

确认结果：

```
# curl -X GET http://localhost/v2/library/hello-world/tags/list
{"name":"library/hello-world","tags":null}
```

删除镜像（ hard delete ）

在上一步中，只是删除了镜像的 manifests 信息，引用的 blob 还在占用磁盘空间，执行如下命令可以查看可以删除的 blob。

```
docker exec -it myregistry_registry_1 /bin/registry garbage-collect --dry-run /etc/registry/config.yml
```

要删除 blob，释放磁盘空间，需要执行下面的命令。
注意：
在执行下面的命令时 registry 必须是只读模式（只读模式可在 registry 配置文件中设置），否则可能会导致数据不一致。

```
docker exec -it myregistry_registry_1 /bin/registry garbage-collect /etc/registry/config.yml
```

附录

目录结构

```
.
|-- config
|   |-- nginx
|   |   `-- nginx.conf
|   `-- registry
|       `-- config.yml
`-- docker-compose.yml
```

nginx.conf

```
worker_processes auto;

events {
  worker_connections 1024;
  use epoll;
  multi_accept on;
}

http {
  tcp_nodelay on;

  # this is necessary for us to be able to disable request buffering in all cases
  proxy_http_version 1.1;

  upstream registry {
    server registry:5000;
  }

  server {
    listen 80;

    # disable any limits to avoid HTTP 413 for large image uploads
    client_max_body_size 0;

    location /v1/ {
      return 404;
    }
```

```
    location /v2/ {
      proxy_pass http://registry/v2/;
      proxy_set_header Host $http_host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

      # When setting up Harbor behind other proxy, such as an Nginx instance, remove the below line if
the proxy already has similar settings.
      proxy_set_header X-Forwarded-Proto $scheme;

      proxy_buffering off;
      proxy_request_buffering off;

    }
  }
}
```

config.yml

```
version: 0.1
log:
  level: debug
  fields:
    service: registry
storage:
    cache:
        layerinfo: inmemory
    filesystem:
        rootdirectory: /var/lib/registry
    maintenance:
        uploadpurging:
            enabled: false
        readonly:
            enabled: false
    delete:
        enabled: true
http:
    addr: :5000
    secret: yoursecret
```

docker-comose.yaml

```
version: '2'
services:
```

```
registry:
  image: library/registry:2.5.0
  restart: always
  volumes:
    - /data/registry:/var/lib/registry
    - ./config/registry/:/etc/registry/
  environment:
    - GODEBUG=netdns=cgo
  command:
    ["serve", "/etc/registry/config.yml"]
proxy:
  image: library/nginx:1.11.5
  restart: always
  volumes:
    - ./config/nginx:/etc/nginx
  ports:
    - 80:80
    - 443:443
  depends_on:
    - registry
```

运维中心
日志采集

操作场景

日志采集功能是容器服务 TKE 为用户提供的集群内日志采集工具，可以将集群内服务或集群节点特定路径文件的日志发送至日志服务（CLS）。日志采集功能适用于需要对 Kubernetes 集群内服务日志进行存储和分析的用户。
日志采集功能需要为每个集群手动开启并配置采集规则。日志采集功能开启后，日志采集 Agent 会在集群内以 DaemonSet 的形式运行，并根据用户通过日志采集规则配置的采集源、CLS 日志主题和日志解析方式，从采集源进行日志采集，将日志内容发送到 CLS 并存储。您可根据以下操作开启日志采集功能：

- 开启日志采集
- 采集容器标准输出日志
- 采集容器文件日志
- 采集节点文件日志

前提条件

- 请在开启前保证集群节点上有足够资源。开启日志采集功能会占用您集群的部分资源。
- 占用 CPU 资源：0.11 - 1.1核，日志量过大时可根据情况自行调大。
- 占用内存资源：24 - 560MB，日志量过大时可根据情况自行调大。
- 日志长度限制：单条512K，如超过会截断。
- 若使用日志采集功能，请确认 Kubernetes 集群内节点能够访问日志服务 CLS。且以下日志采集功能仅支持 Kubernetes 1.10 及以上版本集群。

概念

- 日志采集 Agent：TKE 用于采集日志信息的 Agent，采用 Loglistener，在集群内以 DaemonSet 的方式运行。
- 日志规则：用户可以使用日志规则指定日志的采集源、日志主题、日志解析方式和配置过滤器。
    - 日志采集 Agent 会监测日志采集规则的变化，变化的规则会在最多10s内生效。
    - 多条日志采集规则不会创建多个 DaemonSet，但过多的日志采集规则会使得日志采集 Agent 占用的资源增加。
- 日志源：包含指定容器标准输出、容器内文件以及节点文件。
    - 在采集容器标准输出日志时，用户可选择所有容器、或指定工作负载和指定 Pod Labels 内的容器服务日志作为日志的采集源。
    - 在采集容器文件路径日志时，用户可指定工作负载或 Pod Labels 内容器的文件路径日志作为采集源。
    - 在采集节点文件路径日志时，用户可设定日志的采集源为节点文件路径日志。
- 消费端：用户选择日志服务 CLS 的日志集和日志主题作为消费端。

- 提取模式：日志采集 Agent 支持将采集到的日志以单行文本、JSON、分隔符、多行文本和完全正则的形式发送至用户指定的日志主题。
- 过滤器：开启过滤器后可以根据用户指定的规则采集部分日志，key 支持完全匹配，过滤规则支持正则匹配，如仅采集 ErrorCode = 404 的日志。

操作步骤

开启日志采集

1. 登录 容器服务控制台，选择左侧导航栏中的【集群】。
2. 在页面上方选择地域，单击需要开启日志采集的集群右侧的【设置】。
3. 在“设置功能”页面，单击日志采集【编辑】，开启日志采集后确认。

配置日志规则

1. 登录 容器服务控制台，选择左侧导航栏中的【运维中心】>【日志采集】。
2. 在页面上方选择地域和需要配置日志采集规则的集群，单击【新建】。



3. 在“新建日志采集规则”页面中，选择采集类型，并配置日志源。目前采集类型支持 容器标准输出、容器文件路径 和 节点文件路径。

采集容器标准输出日志

选择【容器标准输出】采集类型，并根据需求配置日志源。该类型日志源支持一次选择多个 Namespace 的工作负载。

采集容器内文件日志

选择【容器文件路径】采集类型，并配置日志源。
路径支持文件路径和通配规则，例如当容器文件路径为 /opt/logs/*.log ，可以指定采集路径为 /opt/logs ，文件名为 *.log 。

> 对于容器的标准输出及容器内文件（非 hostPath 挂载），除了原始的日志内容， 还会带上容器或 kubernetes 相关的元数据（例如：产生日志的容器 ID）一起上报到 CLS，方便用户查看日志时追溯来源或根据容器标识、特征（例如：容器名、labels）进行检索。

容器或 kubernetes 相关的元数据请参考下方表格：

字段名	含义
container_id	日志所属的容器 ID。

字段名	含义
container_name	日志所属的容器名称。
image_name	日志所属容器的镜像名称 IP。
namespace	日志所属 pod 的 namespace。
pod_uid	日志所属 pod 的 UID。
pod_name	日志所属 pod 的名字。
pod_lable_{label name}	日志所属 pod 的 label（例如一个 pod 带有两个 label：app=nginx，env=prod， 则在上传的日志会附带两个 metedata：pod_label_app:nginx，pod_label_env:prod）。

采集节点文件日志

选择【节点文件路径】采集类型，用户可根据实际需求进行添加自定义的“metadata”，将采集到的日志信息附加指定 Key-Value 形式的“metadata”，附加 metadata 将会添加到日志记录中。如下图所示：

> 注意：
> 一个节点日志文件只能被一个日志主题采集。

路径支持文件路径和通配规则，例如当需要采集所有文件路径形式为 /opt/logs/service1/*.log ， /opt/logs/service2/*.log ，可以指定采集路径的文件夹为 /opt/logs/service* ，文件名为 *.log 。

1. 配置日志服务 CLS 为消费端。选择日志集和相应的日志主题，可以选择新建和已有日志主题。

   > 注意：
   > - 日志服务 CLS 目前只能支持同地域的容器集群进行日志采集上报。
   > - 若日志主题下已存在10个日志主题，则不能新建日志主题。

2. 单击【下一步】，选择日志提取模式。

   > 注意：
   > 一个日志主题目前仅支持一个采集配置，请保证选用该日志主题的所有容器的日志都可以接受采用所选的日志解析方式。若在同一日志主题下新建了不同的采集配置，旧的采集配置会被覆盖。

提取模式	说明	相关文档
单行文本	一条日志仅包含一行的内容，以换行符 \n 作为一条日志的结束标记，每条日志将被解析为键值为 CONTENT 的一行完全字符串，开启索引后可通过全文检索搜索日志内容。日志时间以采集时间为准。	单行文本格式
多行文本	指一条完整的日志跨占多行，采用首行正则的方式进行匹配，当某行日志匹配上预先设置的正则表达式，就认为是一条日志的开头，而下一个行首出现作为该条日志的结束标识符，也会设置一个默认的键值 CONTENT，日志时间以采集时间为准。	多行文本格式

提取模式	说明	相关文档
完全正则	指将一条完整日志按正则方式提取多个 key-value 的日志解析模式，您需先输入日志样例，其次自定义正则表达式，系统将根据正则表达式里的捕获组提取对应的 key-value。	完全正则格式
JSON	JSON 格式日志会自动提取首层的 key 作为对应字段名，首层的 value 作为对应的字段值，以该方式将整条日志进行结构化处理，每条完整的日志以换行符\n为结束标识符。	JSON 格式
分隔符	指一条日志数据可以根据指定的分隔符将整条日志进行结构化处理，每条完整的日志以换行符\n为结束标识符。日志服务在进行分隔符格式日志处理时，您需要为每个分开的字段定义唯一的 key，无效字段即无需采集的字段可填空，不支持所有字段均为空。	分隔符格式

3. 根据需求开启过滤器并配置规则，并单击【完成】，完成创建。

更新日志规则

1. 登录 容器服务控制台，选择左侧导航栏中的【集群】。
2. 在页面上方选择地域和需要更新日志采集规则的集群，单击右侧的【编辑收集规则】。
3. 根据需求更新相应配置，单击【完成】，完成更新。

> 注意：
> 日志集和日志主题不可更新。

采集容器日志到对应消费端

本文将介绍如何在容器服务控制台配置日志采集规则并投递到对应消费端。

前提条件

您已购买对应功能。
您已获取租户端控制台登录账号和密码。

操作步骤

创建日志采集规则

1. 登录容器服务控制台，选择左侧导航栏中的【运维中心】>【日志采集】，进入日志页面。
2. 在日志页面上方选择地域和需要配置日志采集规则的集群，单击【新建】。
3. 在新建日志采集规则页面中配置日志服务消费端。
   收集规则名称：您可以自定义日志收集规则名称。
   所在地域：选择日志投递的目标地域。
4. 选择采集类型并配置日志源。
5. 单击【完成】，在日志列表查看。

更新日志规则

1. 登录容器服务控制台，选择左侧导航栏中的【运维中心】>【日志采集】，进入日志页面。
2. 在日志页面上方选择地域和需要配置日志采集规则的集群，单击【编辑收集规则】。
3. 根据需要更新响应配置，单击【完成】。

节点管理
节点概述

简介

节点是容器集群组成的基本元素。节点取决于业务，既可以是虚拟机，也可以是物理机。每个节点都包含运行 Pod 所需要的基本组件，包括 Kubelet、Kube-proxy 等。

节点相关操作

- 新增节点
- 移除节点
- 驱逐或封锁节点
- 设置节点的启动脚本

节点生命周期

节点生命周期状态说明

状态	说明
健康	节点正常运行，并连接上集群。
异常	节点运行异常，未连接上集群。
已封锁	节点已被封锁，不允许新的 Pod 调度到该节点。
驱逐中	节点正在驱逐 Pod 到其他节点。
其他状态	参考云服务器生命周期。

新增节点

操作场景

您可以通过以下方式为集群添加节点。

- 新建节点
- 添加已有节点

前提条件

已登录TKE 控制台。

操作步骤

新建节点

1. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
2. 在需要创建云服务器的集群中，选择【节点管理】>【节点】>【新建节点】。
3. 在“选择机型”页面，根据实际需求，选择可用区、节点网络和实例类型等。
4. 单击【下一步】。
5. 在“云主机配置”页面，配置云服务器。
   主要参数信息如下：
   - 系统盘：提供普通云硬盘、高性能云硬盘和 SSD 云硬盘，请根据实际需求进行选择。
   - 数据盘：请根据实际需求进行选择。
   - 公网宽带：提供 “按宽带计费” 和 “按使用流量” 两种计费模式，请根据实际需求进行选择。
   - 主机名：
     - 自动命名：云服务器将自动命名，其命名格式为 ccs_集群id_node。
     - 手动命名：自定义云服务器名称，云服务器名称不超过60个字符。
   - 登录方式：
     - 设置密码：请根据提示设置对应密码。
     - 立即关联密码：密钥对是通过一种算法生成的一对参数，是比常规密码更安全的登录云服务器的方式，具体详情可参阅 SSH 密钥。
     - 自动生成密码：自动生成的密码，该密码将通过站内信发送给您。
   - 安全组：用于设置云服务器 CVM 的网络访问控制，请根据实际需求进行选择。您还可以单击【新建安全组】，放通其他端口。
6. 单击【下一步】。

7. 在【信息确认】页面，确认已选配置的信息，设置云主机数量，并单击【完成】。

添加已有节点

> 说明：
> 当前仅支持添加同一 VPC 下的云服务器。

1. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
2. 选择【节点管理】>【节点】，进入节点列表页面。
3. 在需要添加已有节点的集群行中，单击【添加已有节点】。
4. 在“选择节点”页面，勾选需要添加的节点，单击【下一步】。
5. 在“云主机配置”页面，配置需要添加到集群的云服务器。主要参数信息如下：
    - 数据盘设置：选择 “将容器和镜像存储在数据盘”。
        - 暂不设置：不设置容器和镜像的存储路径，系统盘容量足够大或无数据盘的云主机，或者数据盘需要后续手动挂载并使用的情况，建议选择不设置容器目录。
        - 将容器与镜像存储在数据盘：系统盘容量较小，或有数据盘的主机，需接受格式化数据盘的情况，可选择设置容器与镜像的存储目录。
    - 操作系统：请根据实际需求进行选择。
    - 登录方式：
        - 设置密码：请根据提示设置对应密码。
        - 立即关联密码：密钥对是通过一种算法生成的一对参数，是比常规密码更安全的登录云服务器的方式，具体详情可参阅 SSH 密钥。
        - 自动生成密码：自动生成的密码，该密码将通过站内信发送给您。
    - 安全组：用于设置云服务器 CVM 的网络访问控制，请根据实际需求进行选择。您还可以单击【新建安全组】，放通其他端口。
6. 单击【完成】。

移除节点

操作场景

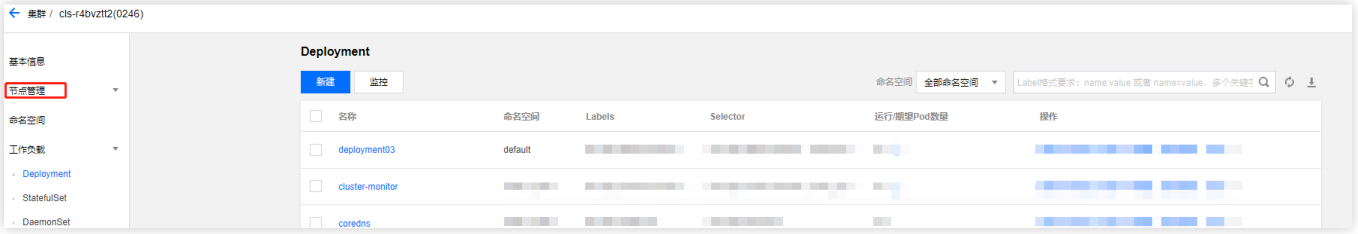
本文档指导您移除集群下的节点。

注意事项

- 按量计费节点移除节点可选择销毁或不销毁，如若不销毁，将继续扣费。
- 节点移出后再添加到集群将会进行重装系统，请谨慎操作。

操作步骤

1. 登录TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要移除节点的集群【ID/名称】，进入该集群的管理页面。



4. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。
5. 在节点列表中，选择需要移除节点的节点行，单击【移出】。
6. 在弹出的“您确定要移出以下节点么？”窗口中，单击【确定】，即可完成操作。

驱逐或封锁节点

操作场景

本文档指导您如何驱逐或封锁节点。

操作步骤

封锁节点

封锁（cordon）节点后，将不接受新的 Pod 调度到该节点，您需要手动取消封锁的节点。封锁节点有以下两种方法：

方法一

新增节点 时，在"云主机配置"页面，单击【高级设置】，勾选【开启封锁】。



方法二

1. 登录TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入"集群管理"页面。
3. 单击需要封锁节点的集群【ID/名称】，进入该集群的管理页面。
4. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。
5. 在节点列表中，选择需要封锁的节点行，单击【封锁】。



6. 在弹出的对话框中，单击【确定】，即可完成封锁。

取消封锁节点

取消封锁（uncordon）节点后，将允许新的 Pod 调度到该节点。取消封锁有以下两种方法：

方法一

通过执行脚本的方式新增节点时，您可以在该脚本中添加取消封锁节点的命令，即可取消封锁。其示例如下：

```
#!/bin/sh
#your initialization script
echo "hello world!"
#If you set unschedulable when you create a node,
#after executing your initialization script,
#use the following command to make the node schedulable.
node=`ifconfig eth0 | grep inet | awk '{print $2}' | tr -d "addr:"`
#echo ${node}
kubectl uncordon ${node} --kubeconfig=/root/.kube/config
```

kubectl uncordon 命令即表示取消封锁节点。

方法二

1. 登录TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要取消封锁节点的集群 ID/名称，进入该集群的管理页面。
4. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。
5. 在节点列表中，选择需要取消封锁的节点行，单击【取消封锁】。



6. 在弹出的对话框中，单击【确定】，即可完成取消封锁。

驱逐节点

概述

在节点上执行维护之前，您可以通过驱逐（drain）节点安全地从节点中逐出 Pod。节点驱逐后，自动将节点内的所有 Pod（不包含 DaemonSet 管理的 Pod）驱逐到集群内其他节点上，并将驱逐的节点设置为封锁状态。

> 说明：
> 本地存储的 Pod 被驱逐后数据将丢失，请谨慎操作。

操作方法

1. 登录TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要取消封锁节点的集群【ID/名称】，进入该集群的管理页面。
4. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。
5. 在需要驱逐节点的节点行中，单击【更多】>【驱逐】。



6. 在弹出的对话框中，单击【确定】，即可完成驱逐。

设置节点的启动脚本

操作场景

设置节点的启动脚本可以帮助您在节点 ready 前对您的节点进行初始化工作，既当节点启动的时候运行配置的脚本。如果一次购买多台云服务器，自定义数据会在所有的云服务器上运行。

使用限制

- 建议您不要通过启动脚本修改 TKE 节点上的 Kubelet、kube-proxy、docker 等配置。
- 启动脚本执行失败不重试，需自行保证脚本的可执行性和重试机制。
- 脚本及其生成的日志文件可在节点的 /data/ccs_userscript/ 路径查看。

操作步骤

您可以在以下三个场景设置节点的启动脚本：

- 创建集群或新增节点时，设置节点的启动脚本
- 添加已有节点时，设置节点的启动脚本
- 创建伸缩组时，设置节点的启动脚本

创建集群或新增节点时，设置节点的启动脚本

- 创建集群时，在“云主机配置”页面，单击【高级设置】，填写自定义数据，启动脚本。



- 新增节点时，在“云主机配置”页面，单击【高级设置】，填写自定义数据，启动脚本。



添加已有节点时，设置节点的启动脚本

添加已有节点时，在“云主机配置”页面，单击【高级设置】，填写自定义数据，启动脚本。



创建伸缩组时，设置节点的启动脚本

创建伸缩组时，在“云主机配置”页面，单击【高级设置】，填写自定义数据，启动脚本。如下图所示：



设置节点Label

操作场景

本文档指导您设置节点 Label。

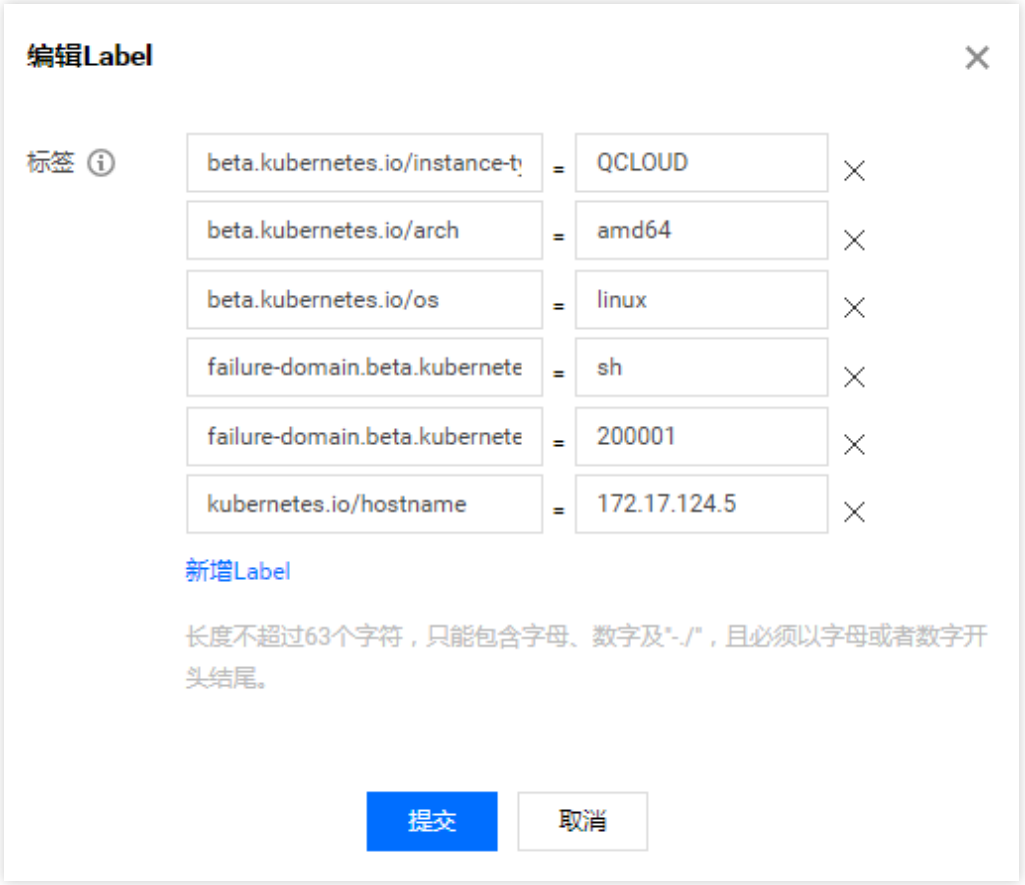
使用限制

- *kubernetes* 和 *qcloud* 相关标签禁用编辑和删除。
- *kubernetes* 和 *qcloud* 标签为保留键，不支持添加。
- 当前仅支持单个节点设置 Label，不支持批量设置。

操作步骤

控制台设置节点 Label

1. 登录TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 选择需要设置节点 Label 的集群【ID/名称】，进入集群详情。
4. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。
5. 选择需要设置 Label 的节点行，单击【更多】>【编辑标签】。
6. 在弹出的“编辑 Label”窗口中，编辑 Label，单击【提交】。



Kubectl 设置节点 Label

1. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。

2. 执行以下命令，设置节点 Label。

   kubectl label nodes <node-name> <label-key>=<label-value>

3. 执行以下命令，查看节点 Label。

   kubectl get nodes --show-labels

返回类似如下信息：

```
NAME          STATUS   ROLES    AGE     VERSION        LABELS
172.17.124.5  Ready    <none>   12d     v1.10.5-tke.3  beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=QCLOUD,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/region=s
```

h,failure-domain.beta.kubernetes.io/zone=200001,kubernetes.io/hostname=172.17.124.5
172.17.124.8 Ready <none> 12d v1.10.5-tke.3 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=QCLOUD,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/region=sh,failure-domain.beta.kubernetes.io/zone=200001,kubernetes.io/hostname=172.17.124.8

创建伸缩组

操作场景

本文指导您创建伸缩组

前提条件

已创建集群

操作步骤

1. 登录TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建伸缩组的集群【ID/名称】，进入该集群的管理页面。
4. 在左侧导航栏中，选择【节点管理】 > 【伸缩组】，进入“伸缩组列表”页面。
5. “新建伸缩组”页面，参考提示进行设置。
   名称：自定义，可根据业务需求等信息进行命名，方便后续资源管理。
   创建方式/机型设置：根据实际需求进行选择。
   登录方式：提供以下三种登录方式，请根据实际情况进行选择。
   - 设置密码：根据提示自定义密码
   - 立即关联密钥：密钥对是通过算法生成的一对参数，是一种比常规密码更安全的登录云服务器的方式。详情请参见 SSH 密钥。
   - SSH密钥：此项仅在登录方式选择“立即关联密钥”后出现。如果已有SSH密钥可以根据需要选择密钥；若需新建，请参考创建 SSH 密钥。
   - 自动生成密码：自动生成的密码将通过站内信发送给您。

   数据盘挂载：根据实际需求进行选择
   安全组：默认为创建集群时所设置的安全组，可根据实际需要进行更换或添加。
   Label：单击【新增Label】，即可进行 Label 自定义设置。可用于后续根据 Label 筛选、管理节点。
   高级设置（可选）：
   - 自定义数据：指定自定义数据来配置Node，即当Node启动后运行配置的脚本，需要自行保证脚本的可重入及重试逻辑, 脚本及其生成的日志文件可在节点的/data/ccs_userscript/路径查看。
   - 封锁（cordon）：勾选开启封锁后，将不接受新的 Pod 调度到该节点，需要手动取消封锁的节点，或在自定义数据中执行 取消封锁命令，请按需设置。

   支持网络/支持子网：根据实际需求进行选择。
   节点数量范围：根据实际需求进行填写。

重试策略：

- 快速重试：立即重试，在较短时间内快速重试，连续失败超过一定次数（5次）后不再重试。
- 间隔递增重试：间隔递增重试，随着连续失败次数的增加，重试间隔逐渐增大，重试间隔从秒级到1天不等。

6. 单击【创建伸缩组】即可创建伸缩组。

查看伸缩组

操作场景

本文介绍如何通过容器服务控制台 查看集群中已创建的伸缩组，并获取伸缩组的详细信息，以便后续对伸缩组进行管理。

前提条件

集群下已创建伸缩组

操作步骤

1. 登录TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建伸缩组的集群【ID/名称】，进入该集群的管理页面。
4. 在左侧导航栏中，选择【节点管理】 > 【伸缩组】，进入“伸缩组列表”页面。
   伸缩组信息及配置如下：
   - **全局配置：**包含该集群下所有伸缩组的共同配置项。可单击该模块右上角【编辑】进行修改，详情请参见调整伸缩组全局配置。
     - 自动缩容：本例此处已关闭。正常开启时，集群中节点空闲资源较多时将触发缩容。详情请参见 集群自动扩缩容说明。
     - 扩容算法：本例此处默认为“随机”，表示伸缩组将随机选择一个伸缩组进行扩容。容器服务还支持以下两种扩容算法，您可根据实际需求进行更改：
       - most-pods：选择能调度更多 Pod 的伸缩组进行扩容。
       - least-waste：选择 Pod 调度后资源剩余更少的伸缩组进行扩容。
     - 集群规模上限：展示当前集群规模信息。对已有伸缩组进行数量调整或再次新建伸缩组时，请注意参考此处规模限制，合理设置节点池的节点数量。
   - 伸缩组列表：全局配置下方即为伸缩组排列区域，主要包含以下信息：
     说明：当伸缩组较多时，可在该区域右上角的搜索框中输入伸缩组 ID 或节名称进行筛选。
     - 伸缩组 ID/名称：单击伸缩组 ID 可进入该伸缩组详情页查看更多相关信息。
     - 状态：本例为“已启用”，表示该伸缩组处于正常状态。
     - 操作：包含启用、停用、更多等，详情请参见 调整伸缩组配置。
     - 启动配置ID/名称：此配置是由创建伸缩组时，所选机型决定，详情请参见启动配置。

调整伸缩组

操作场景

本文介绍如何通过容器服务控制台 调整伸缩组配置。

前提条件

已创建已启用的伸缩组。

操作步骤

调整全局配置

1. 在“伸缩组列表”页面，单击“全局配置”右侧【编辑】，进入配置页面。
2. 在弹出的“设置集群伸缩全局配置”窗口中，参考以下信息进行设置。
   - 自动缩容：默认不勾选。开启自动缩容时，集群中节点空闲资源较多时将触发缩容。详情请参见 集群自动扩缩容说明。
   - 缩容配置：此项在勾选自动缩容后显示。
     - 最大并发缩容数：该数值表示为可以同时进行缩容的节点数，此处默认为“10”，可按需自定义设置。
       注意：此处只缩容完全空闲的空节点。如果节点上存在 Pod，则每次缩容最多一个节点。
     - Pod 占用资源/可分配资源小于：可设置 Pod 占用资源/可分配资源在占比小于设定值时开始判断缩容条件。占比值范围需确保在0 - 80之间。
     - 节点连续空闲：可自定义设置节点连续空闲时间超过几分钟之后会被缩容。
     - 集群扩容：可自定义设置集群首次判断扩容条件的时间点
     - 不缩容节点：请根据实际需求勾选配置项。
   - 扩容算法
     - 随机：有多个伸缩组时，随机选择一个进行扩容。
     - most-pods：有多个伸缩组时，选择能调度更多 Pod 的伸缩组进行扩容。
     - least-waste：有多个伸缩组时，选择 Pod 调度后资源剩余更少的伸缩组进行扩容。
3. 单击【确定】，即可设置成功。

调整伸缩组配置

1. 在“伸缩组列表”页面，单击所需配置的伸缩组ID，进入配置页面。
2. 在伸缩组详情页面可以通过点击【编辑】来调整伸缩组配置，详情请参见修改伸缩组。

说明：调整最大、最小伸缩的实例数有两种方法。

- 通过伸缩组详情页面调整“最大伸缩数”和“最小伸缩数”
- 通过伸缩组列表页面“操作”列的【更多】>【调整配置】中的“实例范围”

删除伸缩组

操作场景

本文介绍如何通过容器服务控制台 删除集群下已创建的伸缩组。您可参考本文删除不再使用的伸缩组，减少不必要的资源浪费。

前提条件

已创建已启用的伸缩组。

1. 单击伸缩组列表页面“操作”列的【更多】>【删除】。
2. 在弹出的“删除伸缩组”窗口中，按需设置是否保留节点。
   注意：
   - 默认勾选销毁按量计费的节点，可根据实际需求取消勾选。
   - 按量计费节点销毁后不可恢复，请谨慎操作，提前备份好数据。
3. 单击【提交】，等待删除成功即可。

查看伸缩组伸缩记录

操作场景

本文介绍如何查看伸缩组的伸缩记录，适用于以下场景：

- 您可以通过伸缩活动了解自己业务的流量变化，更有效的按需配置伸缩组。
- 您可以通过伸缩组内节点扩缩容活动来了解自己的花费来源，进行更高效的成本管理。
- 您可以了解扩缩容活动失败的原因（例如扩容时由于地域资源售罄导致扩容失败），进行风险管理。
- 您可以查看两个层级的伸缩记录：全局伸缩记录 和 特定伸缩组伸缩记录
  说明：
- 在存在多个伸缩组的情况下，CA（Cluster Autoscaler）负责选择合适的伸缩组进行扩缩容，全局伸缩记录可以从 CA 的 Event 得到。
- 如果您只关心特定伸缩组的伸缩记录，不关心 CA 的行为，可进入伸缩组详情页查看该伸缩组的扩缩容活动记录。

前提条件

已创建可用伸缩组。详情请参见 创建伸缩组。
已进入“伸缩组列表”页面。详情请参见 查看伸缩组。

操作步骤

查看全局伸缩记录

1. 登录TKE 控制台。
2. 选择左侧导航栏中的【运维中心】>【运维功能管理】，单击目标集群操作列的【设置】。
3. 在弹出的”设置功能”窗口中，选择“事件存储”功能右侧的编辑，勾选“开启事件存储”，并创建或者选择已有的日志主题。
4. 单击确定即可开启事件持久化功能。

查看事件持久化

1. 登录 日志服务控制台。
2. 选择左侧导航栏中的【检索分析】，进入“检索分析”管理页面。
3. 在“检索分析”页面上方选择地域，选择希望查看事件持久化的日志集和日志主题。
4. 勾选event.source.component:cluster-autoscaler，单击检索分析。
5. 在右侧的版面设置可配置数据列，对关注的列进行可视化。

查看特定节点池伸缩记录

1. 登录TKE 控制台，选择左侧导航栏中的【集群】。
2. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “Deployment” 页面。
3. 选择左侧菜单栏中的节点管理 > 伸缩组，进入“伸缩组列表”页面。
4. 在“伸缩组列表”中，单击目标伸缩组 ID。
5. 进入该伸缩组详情页，选择“伸缩活动”页签，即可查看伸缩记录。

Kubernetes对象管理
概述

对象管理说明

您可以通过控制台直接操作原生 Kubernetes 对象，例如 Deployment、DaemonSet 等。
Kubernetes 对象是集群中持久实体，用来承载集群内运行的业务。不同的 Kubernetes 对象可以表达不同的含义：

- 正在运行的应用程序
- 应用程序可用的资源
- 应用程序关联的策略等
  您可以直接通过 TKE 控制台或者 Kubernetes API 使用 Kubernetes 的对象，例如 Kubectl。

对象分类

Kubernetes 常用对象主要分为以下类型：

- 工作负载
  - Deployment：用于管理指定调度规则的 Pod。
  - StatefulSet：管理应用程序的工作负载 API 对象，且该应用程序为有状态的应用程序。
  - DaemonSet：确保所有或部分节点上运行 Pod，例如日志采集。
  - Job：一个 Job 创建一个或多个 Pod，直至运行结束。
  - CronJob：定时运行的 Job 任务。
- 服务
  - Service：提供 Pod 访问的 Kubernetes 对象，可以根据业务需求定义不同类型。
  - Ingress：管理集群中 Services 的外部访问的 Kubernetes 对象。
- 配置
  - ConfigMap：用于保存配置信息。
  - Secret：用于保存敏感信息，例如密码、令牌等。
- 存储
  - Volume：可以存储容器访问相关的数据。
  - Persistent Volumes（PV）：Kubernetes 集群中配置的一块存储。
  - Persistent Volumes Claim（PVC）：请求存储的声明。如果把 PV 比作 Pod，那么 PVC 相当于工作负载。
  - StorageClass：用于描述存储的类型。 创建 PVC 时，通过 StorageClass 创建指定类型的存储，即存储的模板。
    Kubernetes 对象还包括 Namespaces、HPA、Resource Quotas等数十种，您可以根据业务需要使用

不同的 Kubernetes 对象。不同版本的 Kubernetes 可使用的对象也不相同，更多说明可登录 Kubernetes 官方网站 查询。

对象管理操作

- 工作负载
  - Deployment 管理
  - StatefulSet 管理
  - DaemonSet 管理
  - Job 管理
  - CronJob 管理
- 服务
  - Service 管理
  - Ingress 管理
- 配置
  - ConfigMap 管理
  - Secret 管理
- 存储
  - Volumes 管理
  - Persisten Volumes 管理
  - Persisten Volumes Claim 管理
  - Storage Classes 管理

Namespaces

Namespaces 是 Kubernetes 在同一个集群中进行逻辑环境划分的对象，您可以通过 Namespaces 进行管理多个团队多个项目的划分。在 Namespaces 下，Kubernetes 对象的名称必须唯一。您可以通过资源配额进行可用资源的分配，还可以进行不同 Namespaces 网络的访问控制。

使用方法

通过 TKE 控制台使用：

TKE 控制台提供 Namespaces 的增删改查功能。
创建命名空间

1. 登录TKE 控制台。
2. 在左侧导航栏选择【集群】，进入集群管理页面。
3. 选择所需集群，单击该集群的【ID/名称】，进入集群详情页。
4. 在集群详情页左侧，选择【命名空间】，进入 Namespace页面。
5. 单击【新建】，输入名称、描述后，单击【创建Namespace】完成创建。

设置资源配额和限制

1. 在命名空间页面，根据需要单击对应的namespace操作列的【配额管理】，进入配置页面。
2. 在资源配额与限制页面，可以通过单击【编辑配额】快速配置计算、存储、其他资源限制。
3. 在编辑LimitRange页面可以快速设置资源限制。

通过 Kubectl 使用

更多详情可查看 Kubernetes 官网文档。

通过 ResourceQuota 设置 Namespaces 资源的使用配额

一个命名空间下可以拥有多个 ResourceQuota 资源，每个 ResourceQuota 可以设置每个 Namespace 资源的使用约束。可以设置 Namespaces 资源的使用约束如下：

- 计算资源的配额，例如 CPU、内存。
- 存储资源的配额，例如请求存储的总存储。
- Kubernetes 对象的计数，例如 Deployment 个数配额。
  不同的 Kubernetes 版本，ResourceQuota 支持的配额设置略有差异，更多详情可查看 Kubernetes ResourceQuota 官方文档。
  ResourceQuota 的示例如下所示：

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: object-counts
  namespace: default
spec:
  hard:
    configmaps: "10"  ## 最多10个 ConfigMap
    replicationcontrollers: "20" ## 最多20个 replicationcontroller
    secrets: "10" ## 最多10个 secret
    services: "10" ## 最多10个 service
    services.loadbalancers: "2"  ## 最多2个 Loadbanlacer 模式的 service
    cpu: "1000" ## 该 Namespaces 下最多使用1000个 CPU 的资源
    memory: 200Gi ## 该 Namespaces 下最多使用200Gi的内存
```

通过 NetWorkPolicy 设置 Namespaces 网络的访问控制

Network Policy 是 k8s 提供的一种资源，用于定义基于 Pod 的网络隔离策略。不仅可以限制 Namespaces，还可以控制 Pod 与 Pod 之间的网络访问控制，即控制一组 Pod 是否可以与其它组 Pod，以及其它 network endpoints 进行通信。

在集群内部署 NetworkPolicy Controller，并通过 NetworkPolicy 实现 Namespaces 之间的网络控制的操作详情可查看 使用 Network Policy 进行网络访问控制。

工作负载
Deployment管理

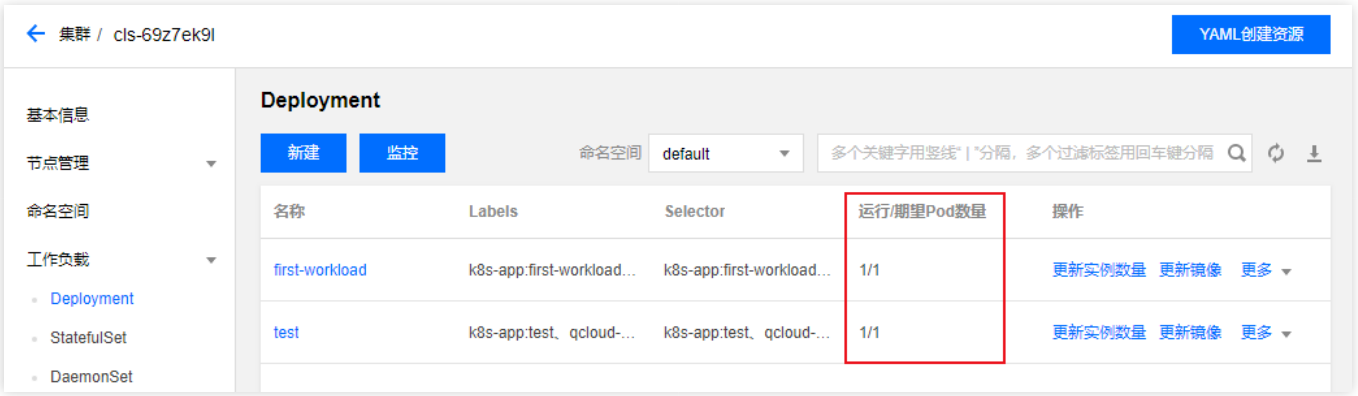
简介

Deployment 声明了 Pod 的模板和控制 Pod 的运行策略，适用于部署无状态的应用程序。您可以根据业务需求，对 Deployment 中运行的 Pod 的副本数、调度策略、更新策略等进行声明。

Deployment 控制台操作指引

创建 Deployment

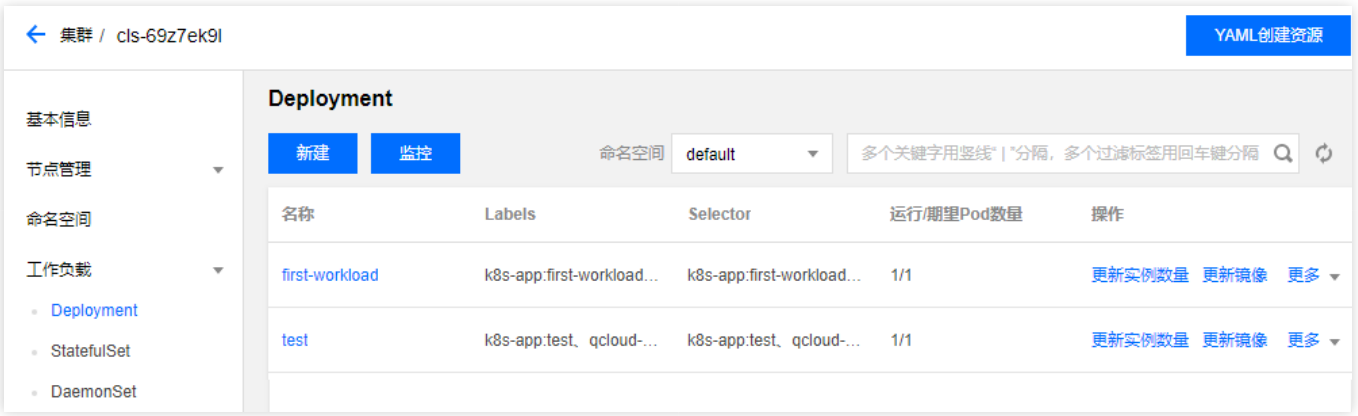
1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建 Deployment 的集群【ID/名称】，进入待创建 Deployment 的集群管理页面。
4. 单击【新建】，进入“新建Workload”页面。
5. 根据实际需求，设置 Deployment 参数。关键参数信息如下：
    - 工作负载名：自定义。
    - 命名空间：根据实际需求进行选择。
    - 类型：选择 “Deployment（可扩展的部署Pod）”。
    - 实例内容器：根据实际需求，为 Deployment 的一个 Pod 设置一个或多个不同的容器。
        - 名称：自定义。
        - 镜像：根据实际需求进行选择。
        - 镜像版本：根据实际需求进行填写。
        - 环境变量：设置容器中的变量。
        - CPU/内存限制：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
        - 高级设置：可设置 “工作目录”，“运行命令”，“运行参数”，“容器健康检查”，“特权级”等参数。
    - 实例数量：根据实际需求选择调节方式，设置实例数量。
6. 单击【创建Workload】，完成创建。
   当运行数量=期望数量时，即表示 Deployment 下的所有 Pod 已创建完成。



更新 Deployment

更新 YAML

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入"集群管理"页面。
3. 单击需要更新 Deployment 的集群【ID/名称】，进入待更新 Deployment 的集群管理页面。



4. 在需要更新 YAML 的 Deployment 行中，单击【更多】>【编辑YAML】，进入更新 Deployment 页面。
5. 在"更新Deployment"页面，编辑 YAML，单击【完成】，即可更新 YAML。

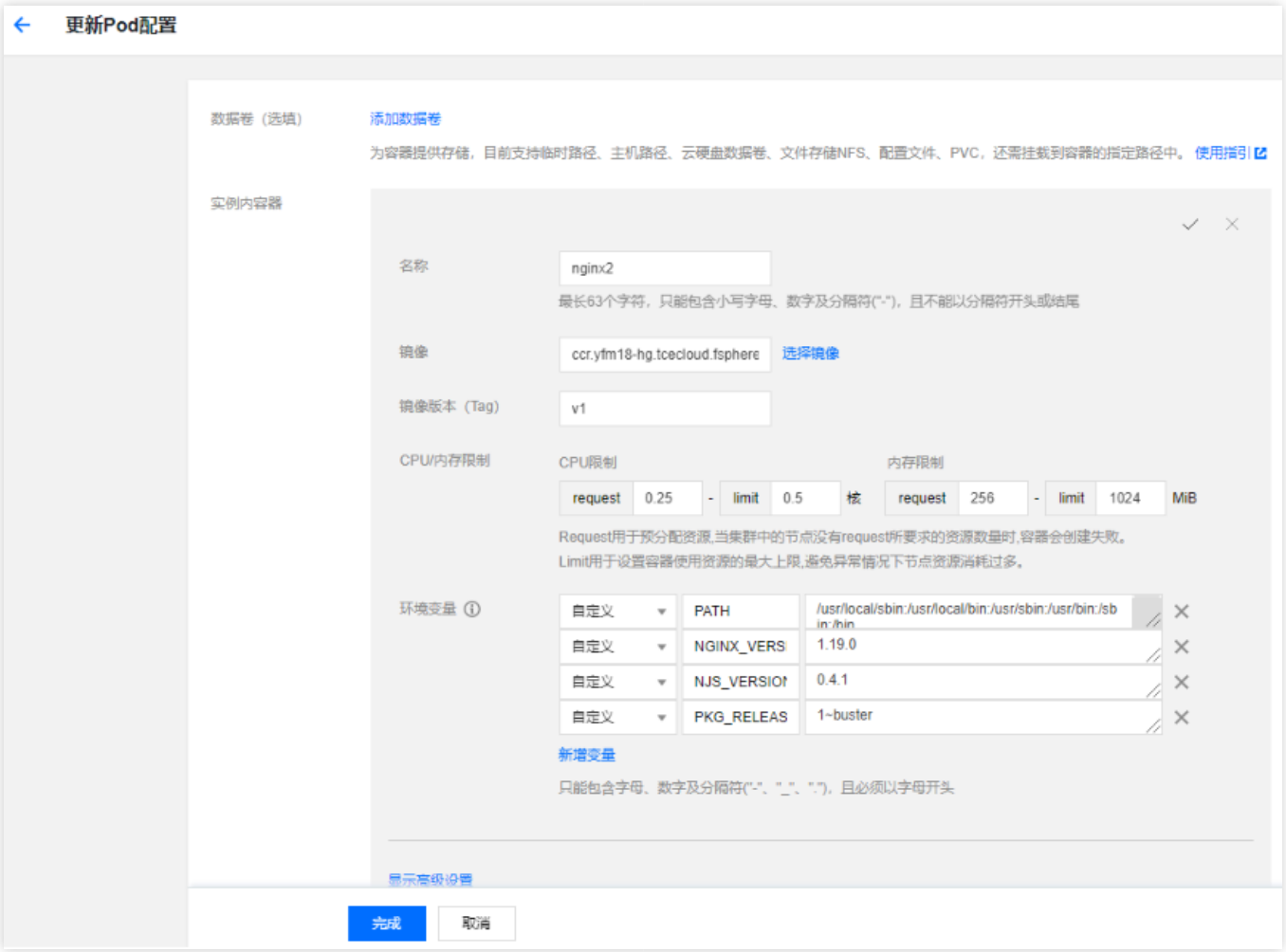


更新Pod配置

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 在集群管理页面，单击需要更新镜像的 Deployment 的集群 ID，进入待更新Pod配置的 Deployment 的集群管理页面。
4. 在需要更新镜像的 Deployment 行中，单击【更新Pod配置】。



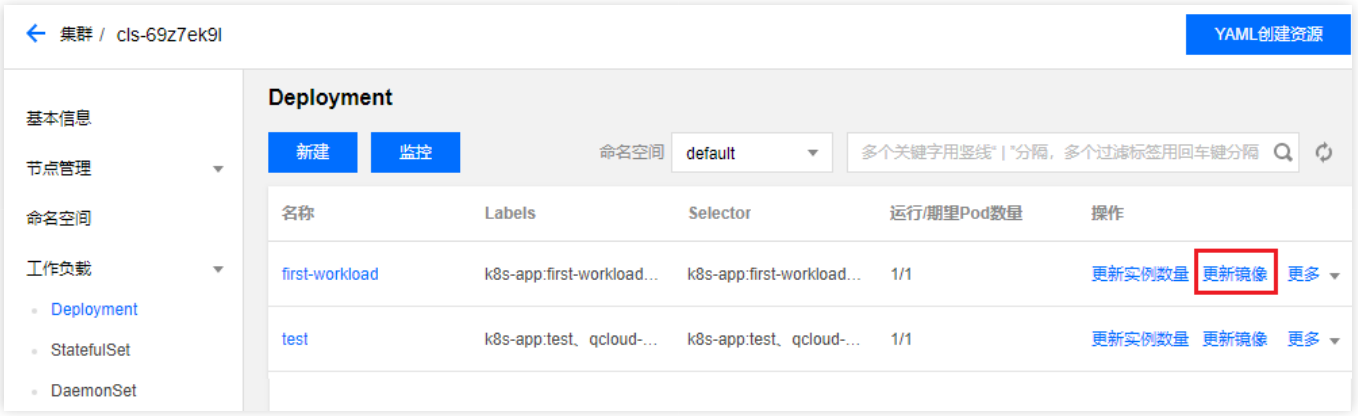
5. 在“更新Pod配置”页面，根据实际需求修改更新方式，设置参数。



6. 单击【完成】，即可更新 Pod 配置。

更新镜像

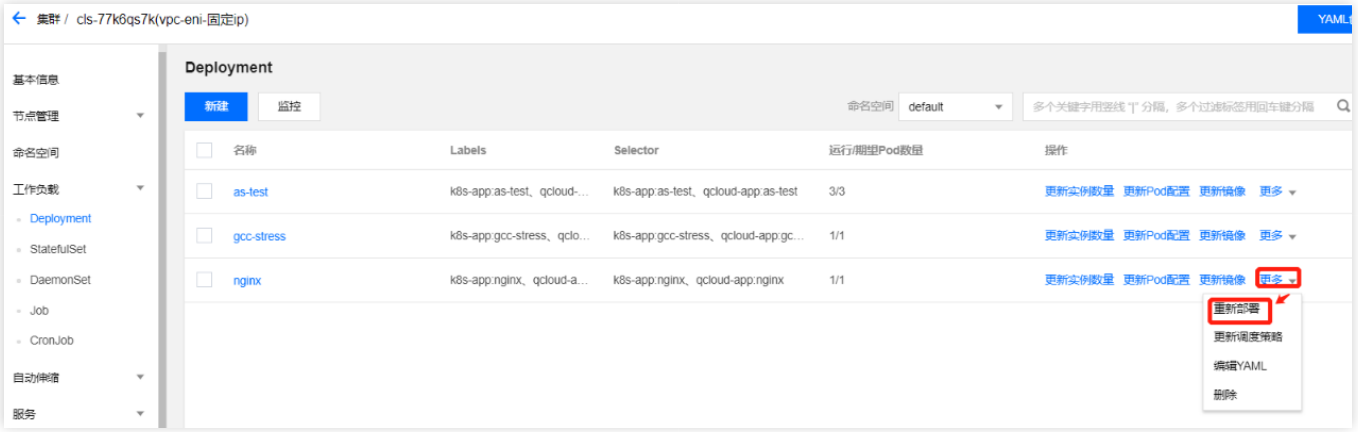
1. 在集群管理页面，单击需要更新镜像的 Deployment 的集群 ID，进入待更新镜像的 Deployment 的集群管理页面。
2. 在需要更新镜像的 Deployment 行中，单击【更新镜像】。



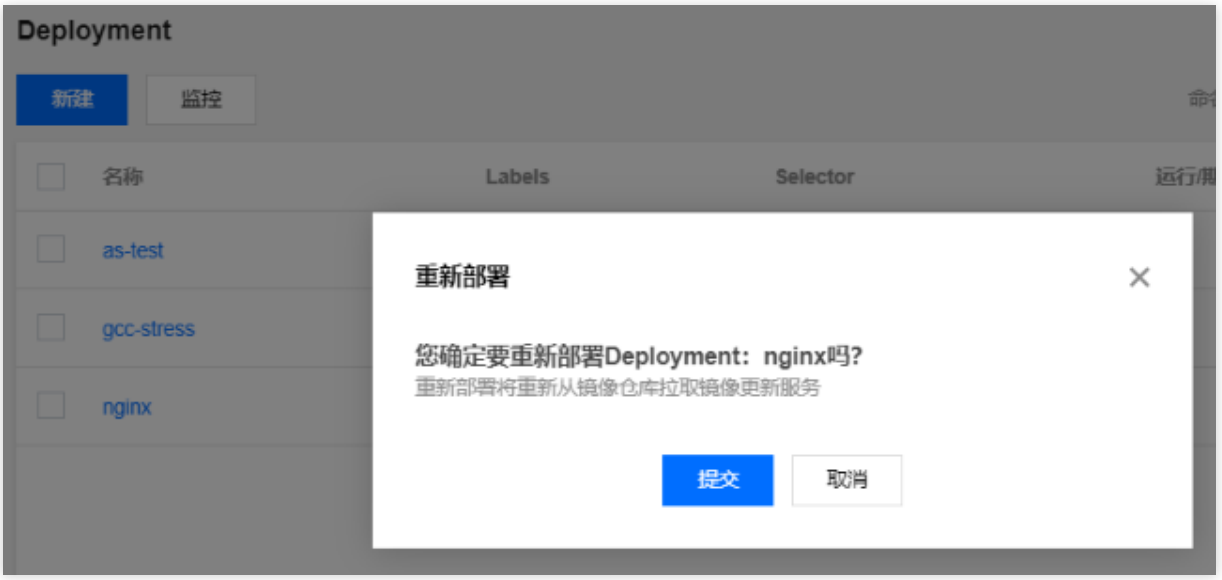
3. 在“滚动更新镜像”页面，根据实际需求修改更新方式，设置参数。
4. 单击【完成】，即可更新镜像。

重新部署Deployment

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 Deployment 的集群【ID/名称】，进入待重新部署 Deployment 的集群管理页面。

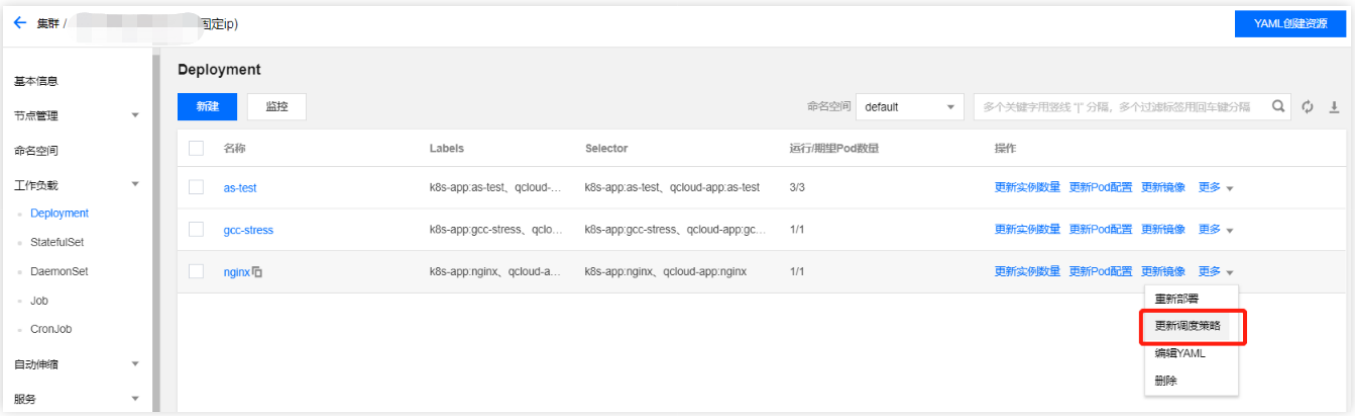


4. 在需要重新部署 的 Deployment 行中，单击【更多】>【重新部署】，弹出“重新部署”窗口。
5. 在“重新部署”窗口，单击【完成】，即可重新部署Deployment。如下图所示：

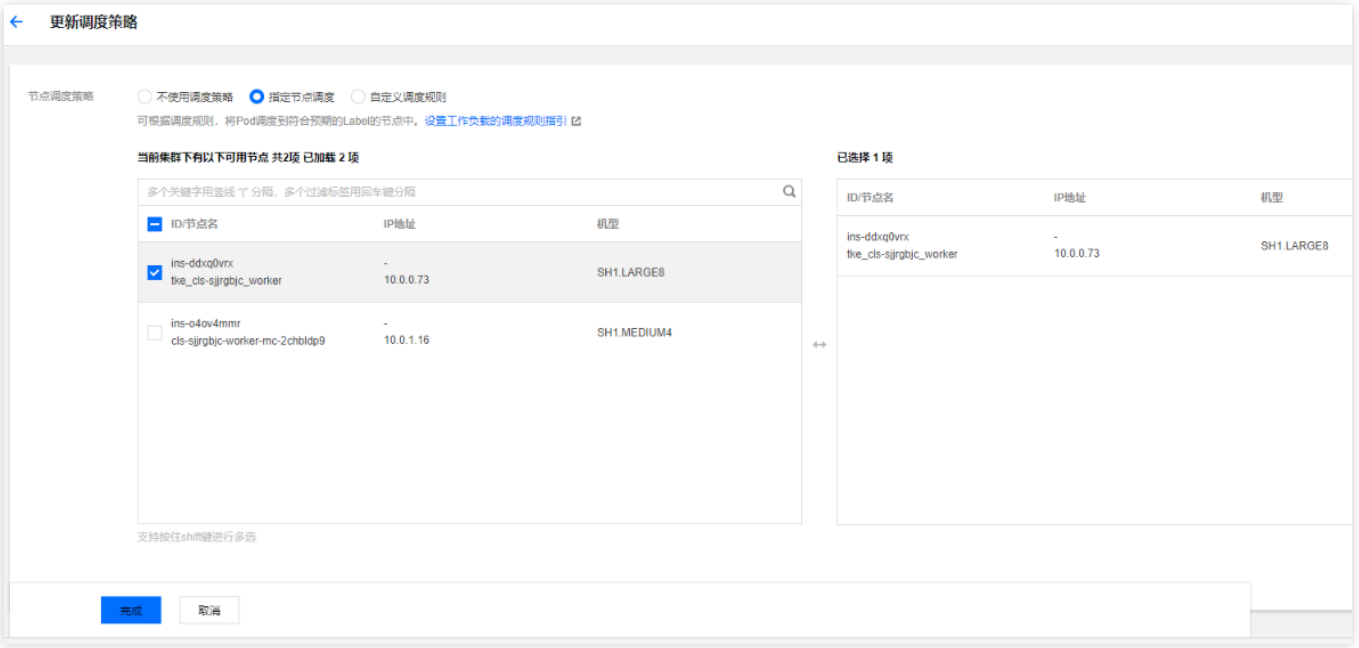


更新调度策略

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 Deployment 的集群【ID/名称】，进入待更新调度策略 Deployment 的集群管理页面。如下图所示：

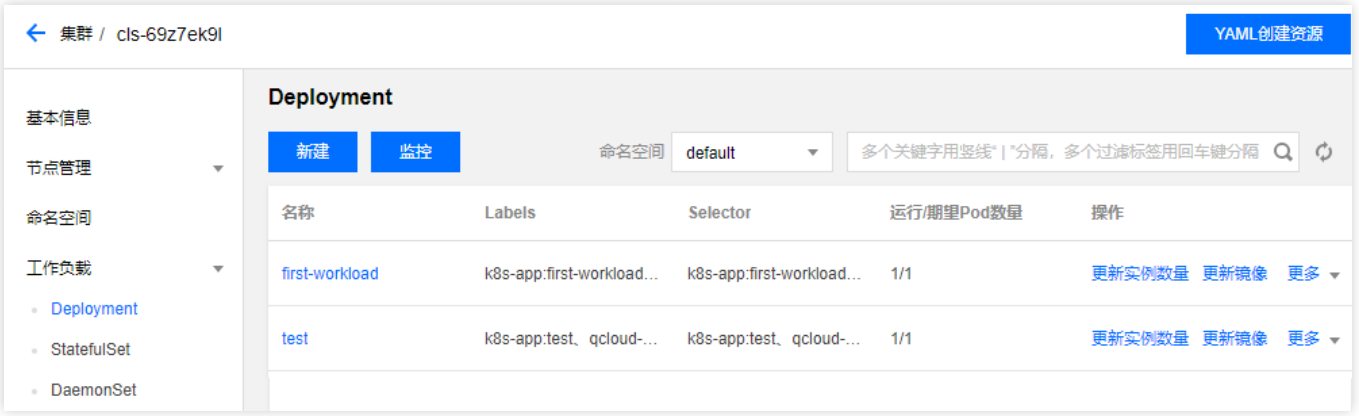


4. 在需要更新调度策略 的 Deployment 行中，单击【更多】>【更新调度策略】，弹出“更新调度策略”窗口。
5. 在“更新调度策略”窗口，更新节点调度策略，单击【完成】。如下图所示：



回滚 Deployment

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要回滚 Deployment 的集群【ID/名称】，进入待回滚 Deployment 的集群管理页面。如下图所示：



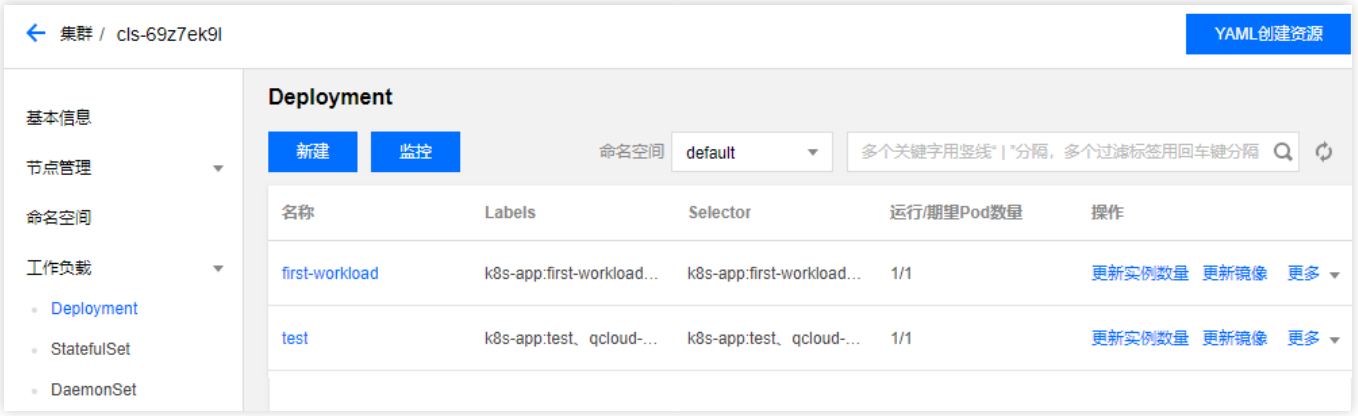
4. 单击需要回滚的 Deployment 名称，进入 Deployment 信息页面。
5. 选择【修订历史】页签，在需要回滚的版本行中，单击【回滚】。如下图所示：



6. 在弹出的【回滚资源】提示框中，单击【提交】，完成回滚。

调整 Pod 数量

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要调整 Pod 数量的 Deployment 的集群【ID/名称】，进入待调整 Pod 数量的 Deployment 的集群管理页面。



4. 在需要调整 Pod 数量的 Deployment 行中，单击【更新实例数量】，进入“更新实例数量”页面。如下图所示：



5. 根据实际需求调整 Pod 数量，单击【更新实例数量】，完成调整。

Kubectl 操作 Deployment 指引

YAML 示例

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: default
  labels:
    app: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-deployment
  template:
    metadata:
      labels:
        app: nginx-deployment
    spec:
      containers:
```

```
- name: nginx
  image: nginx:latest
  ports:
  - containerPort: 80
```

- kind：标识 Deployment 资源类型。
- metadata：Deployment 的名称、Namespace、Label等基本信息。
- metadata.annotations：对 Deployment 的额外说明，可通过该参数设置 TKE 的额外增强能力。
- spec.replicas：Deployment 管理的 Pod 数量。
- spec.selector：Deployment 管理 Seletor 选中的 Pod 的 Label。
- spec.template：Deployment 管理的 Pod 的详细模板配置。

更多参数详情可查看 Kubernetes Deployment 官方文档。

Kubectl 创建 Deployment

1. 参考 YAML 示例，准备 Deployment YAML 文件。

2. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。

3. 执行以下命令，创建 Deployment YAML 文件。

   ```
   kubectl create -f Deployment YAML 文件名称
   ```

   例如，创建一个文件名为 nginx.Yaml 的 Deployment YAML 文件，则执行以下命令：

   ```
   kubectl create -f nginx.yaml
   ```

4. 执行以下命令，验证创建是否成功。

   ```
   kubectl get deployments
   ```

   返回类似以下信息，即表示创建成功。

   ```
   NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
   first-workload   1         1         1            0           6h
   ng             1         1         1            1           42m
   ```

Kubectl 更新 Deployment

通过 Kubectl 更新 Deployment 有以下三种方法。其中，方法一 和 方法二 均支持 Recreate 和 RollingUpdate 两种更新策略。

- Recreate 更新策略为先销毁全部 Pod，再重新创建 Deployment。
- RollingUpdate 更新策略为滚动更新策略，逐个更新 Deployment 的 Pod。RollingUpdate 还支持暂停、设置更新时间间隔等。

方法一

执行以下命令，更新 Deployment。

```
kubectl edit  deployment/[name]
```

此方法适用于简单的调试验证，不建议在生产环境中直接使用。您可以通过此方法更新任意的 Deployment 参数。

方法二

执行以下命令，更新指定容器的镜像。

```
kubectl set image deployment/[name] [containerName]=[image:tag]
```

建议保持 Deployment 的其他参数不变，业务更新时，仅更新容器镜像。

方法三

执行以下命令，滚动更新指定资源。

```
kubectl rolling-update [NAME] -f FILE
```

更多滚动更新可参见 滚动更新说明。

Kubectl 回滚 Deployment

1. 执行以下命令，查看 Deployment 的更新历史。

   ```
   kubectl rollout history deployment/[name]
   ```

2. 执行以下命令，查看指定版本详情。

   ```
   kubectl rollout history deployment/[name] --revision=[REVISION]
   ```

3. 执行以下命令，回滚到前一个版本。

   ```
   kubectl rollout undo deployment/[name]
   ```

如需指定回滚版本号，可执行以下命令。

```
kubectl rollout undo deployment/[name] --to-revision=[REVISION]
```

Kubectl 调整 Pod 数量

手动更新 Pod 数量

执行以下命令，手动更新 Pod 数量。

```
kubectl scale deployment [NAME] --replicas=[NUMBER]
```

自动更新 Pod 数量

前提条件

开启集群中的 HPA 功能。TKE 创建的集群默认开启 HPA 功能。

操作步骤

执行以下命令，设置 Deployment 的自动扩缩容。

```
kubectl autoscale deployment [NAME] --min=10 --max=15 --cpu-percent=80
```

Kubectl 删除 Deployment

执行以下命令，删除 Deployment。

```
kubectl delete deployment [NAME]
```

StatefulSet管理

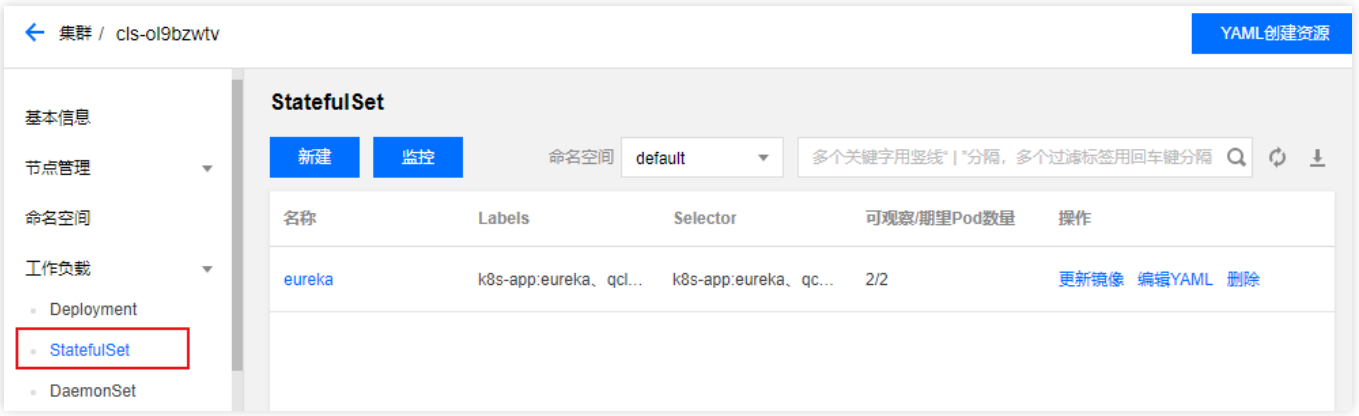
简介

StatefulSet 主要用于管理有状态的应用，创建的 Pod 拥有根据规范创建的持久型标识符。Pod 迁移或销毁重启后，标识符仍会保留。 在需要持久化存储时，您可以通过标识符对存储卷进行一一对应。如果应用程序不需要持久的标识符，建议您使用 Deployment 部署应用程序。

StatefulSet 控制台操作指引

创建 StatefulSet

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建 StatefulSet 的集群【ID/名称】，进入待创建 StatefulSet 的集群管理页面。
4. 选择【工作负载】 > 【StatefulSet】，进入 StatefulSet 信息页面。如下图所示：

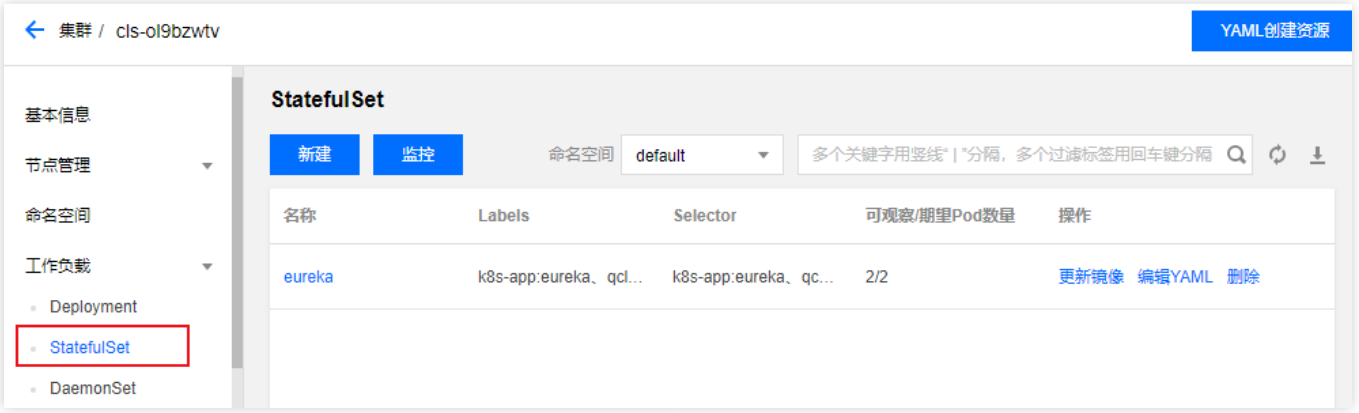


5. 单击【新建】，进入“新建Workload”页面。
6. 根据实际需求，设置 Deployment 参数。关键参数信息如下：
   - 工作负载名：自定义。
   - 命名空间：根据实际需求进行选择。
   - 类型：选择 “StatefulSet（有状态集的运行Pod）”。
   - 实例内容器：根据实际需求，为 StatefulSet 的一个 Pod 设置一个或多个不同的容器。
   - 名称：自定义。
   - 镜像：根据实际需求进行选择。
   - 镜像版本：根据实际需求进行填写。
   - CPU/内存限制：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
   - 高级设置：可设置 “工作目录”，“运行命令”，“运行参数”，“容器健康检查”，“特权级”等参数。
   - 实例数量：根据实际需求选择调节方式，设置实例数量。
7. 单击【创建Workload】，完成创建。

更新 StatefulSet

更新 YAML

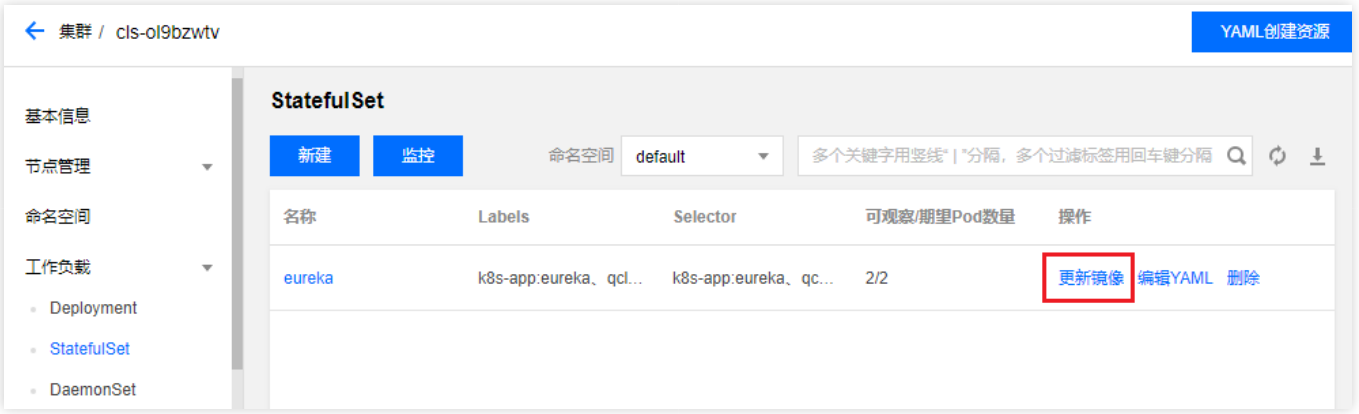
1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 YAML 的集群【ID/名称】，进入待更新 YAML 的集群管理页面。
4. 选择【工作负载】 > 【StatefulSet】，进入 StatefulSet 信息页面。



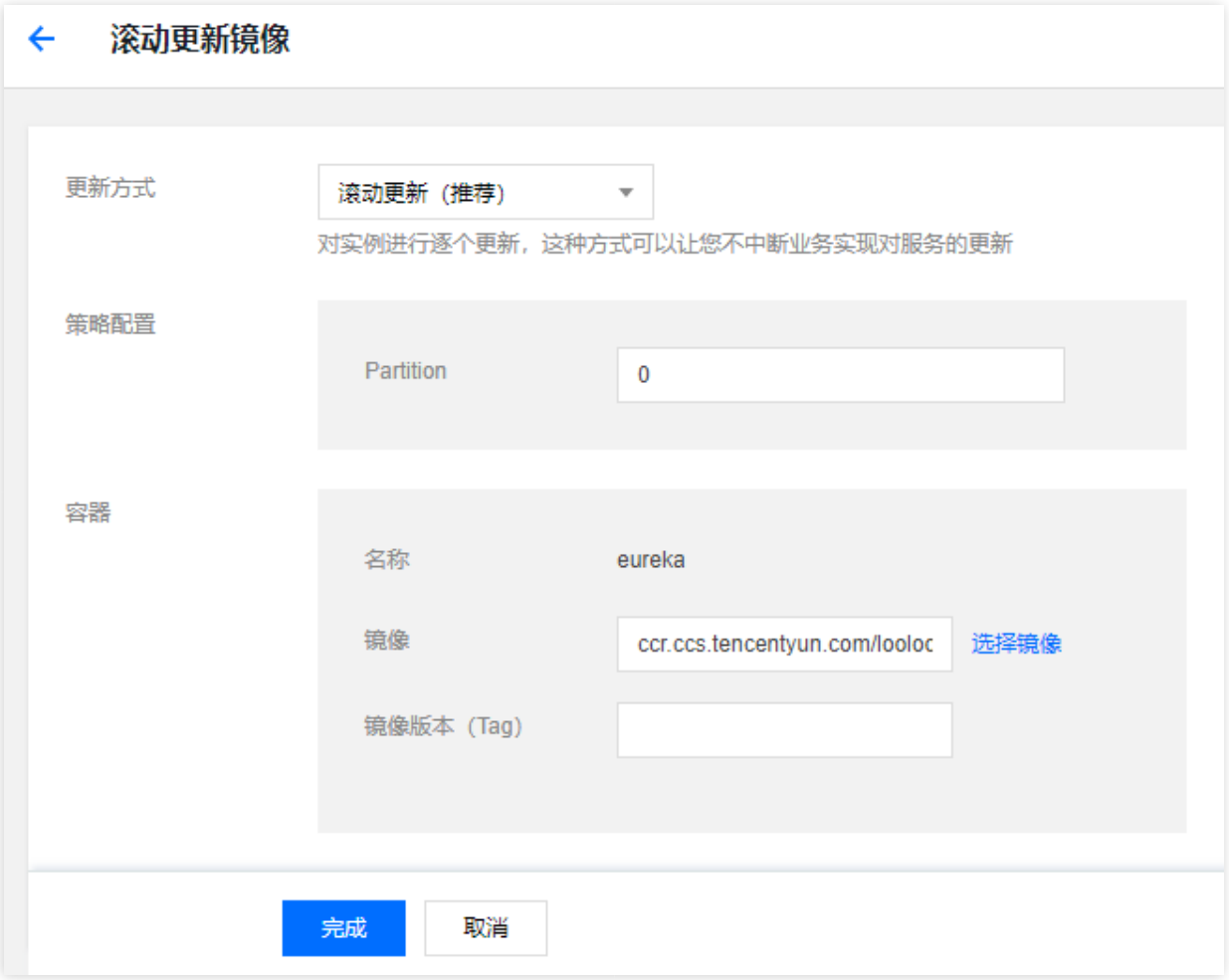
5. 在需要更新 YAML 的 StatefulSet 行中，单击【编辑YAML】，进入更新 StatefulSet 页面。
6. 在【更新StatefulSet】页面，编辑 YAML，单击【完成】，即可更新 YAML。

更新镜像

1. 在“集群管理”页面，单击需要更新镜像的 StatefulSet 的集群【ID/名称】，进入待更新镜像的 StatefulSet 的集群管理页面。
2. 在需要更新镜像的 StatefulSet 行中，单击【更新镜像】。



3. 在 “滚动更新镜像” 页面，根据实际需求修改更新方式，设置参数。



4. 单击【完成】，即可更新镜像。

更新Pod配置

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 在集群管理页面，单击需要更新镜像的 StatefulSet 的集群 ID，进入待更新Pod配置的 StatefulSet 的集群管理页面。
4. 在需要更新镜像的 StatefulSet 行中，单击【更新Pod配置】。
5. 在“更新Pod配置”页面，根据实际需求修改更新方式，设置参数。
6. 单击【完成】，即可更新 Pod 配置。

更新调度策略

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 StatefulSet 的集群【ID/名称】，进入待更新调度策略 StatefulSet 的集群管理页面。
4. 在需要更新调度策略 的 StatefulSet 行中，单击【更多】>【更新调度策略】，弹出“更新调度策略”窗口。
5. 在“更新调度策略”窗口，更新节点调度策略，单击【完成】。

Kubectl 操作 StatefulSet 指引

YAML 示例

```
apiVersion: v1
kind: Service  ## 创建一个 Headless Service，用于控制网络域
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    name: web
  clusterIP: None
  selector:
    app: nginx
---
apiVersion: apps/v1
kind: StatefulSet ### 创建一个 Nginx的StatefulSet
metadata:
  name: web
  namespace: default
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 3 # by default is 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
```

```
accessModes: [ "ReadWriteOnce" ]
storageClassName: "cbs"
resources:
  requests:
    storage: 10Gi
```

- kind：标识 StatefulSet 资源类型。
- metadata：StatefulSet 的名称、Label等基本信息。
- metadata.annotations：对 StatefulSet 的额外说明，可通过该参数设置TKE 的额外增强能力。
- spec.template：StatefulSet 管理的 Pod 的详细模板配置。
- spec.volumeClaimTemplates：提供创建 PVC&PV 的模板。

更多参数详情可查看 Kubernetes StatefulSet 官方文档。

创建 StatefulSet

1. 参考 YAML 示例，准备 StatefulSet YAML 文件。
2. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
3. 执行以下命令，创建 StatefulSet YAML 文件。

   ```
   kubectl create -f StatefulSet YAML 文件名称
   ```

   例如，创建一个文件名为 web.yaml 的 StatefulSet YAML 文件，则执行以下命令：

   ```
   kubectl create -f web.yaml
   ```

4. 执行以下命令，验证创建是否成功。

   ```
   kubectl get StatefulSet
   ```

   返回类似以下信息，即表示创建成功。

   ```
   NAME      DESIRED   CURRENT   AGE
   test      1         1         10s
   ```

更新 StatefulSet

执行以下命令，查看 StatefulSet 的更新策略类型。

```
kubectl get ds/<daemonset-name> -o go-template='{{.spec.updateStrategy.type}}{{"\n"}}'
```

StatefulSet 有以下两种更新策略类型：

- OnDelete：默认更新策略。该更新策略在更新 StatefulSet 后，需手动删除旧的 StatefulSet Pod 才会创建新的 StatefulSet Pod。
- RollingUpdate：支持 Kubernetes 1.7或更高版本。该更新策略在更新 StatefulSet 模板后，旧的 StatefulSet Pod 将被终止，并且以滚动更新方式创建新的 StatefulSet Pod（Kubernetes 1.7或更高版本）。

方法一

执行以下命令，更新 StatefulSet。

```
kubectl edit StatefulSet/[name]
```

此方法适用于简单的调试验证，不建议在生产环境中直接使用。您可以通过此方法更新任意的 StatefulSet 参数。

方法二

执行以下命令，更新指定容器的镜像。

```
kubectl patch statefulset <NAME> --type='json' -p='[{"op": "replace", "path": "/spec/template/spec/containers/130618223492100096/image", "value":"<newImage>"}]'
```

建议保持 StatefulSet 的其他参数不变，业务更新时，仅更新容器镜像。
如果更新的 StatefulSet 是滚动更新方式的策略，可执行以下命令查看更新状态：

```
kubectl rollout status sts/<StatefulSet-name>
```

删除 StatefulSet

执行以下命令，删除 StatefulSet。

```
kubectl delete  StatefulSet [NAME] --cascade=false
```

--cascade=false 参数表示 Kubernetes 仅删除 StatefulSet，且不删除任何 Pod。如需删除 Pod，则执行以下命令：

```
kubectl delete  StatefulSet [NAME]
```

更多 StatefulSet 相关操作可查看 Kubernetes官方指引。

DaemonSet管理

简介

DaemonSet 主要用于部署常驻集群内的后台程序，例如节点的日志采集。DaemonSet 保证在所有或部分节点上均运行指定的 Pod。 新节点添加到集群内时，也会有自动部署 Pod；节点被移除集群后，Pod 将自动回收。

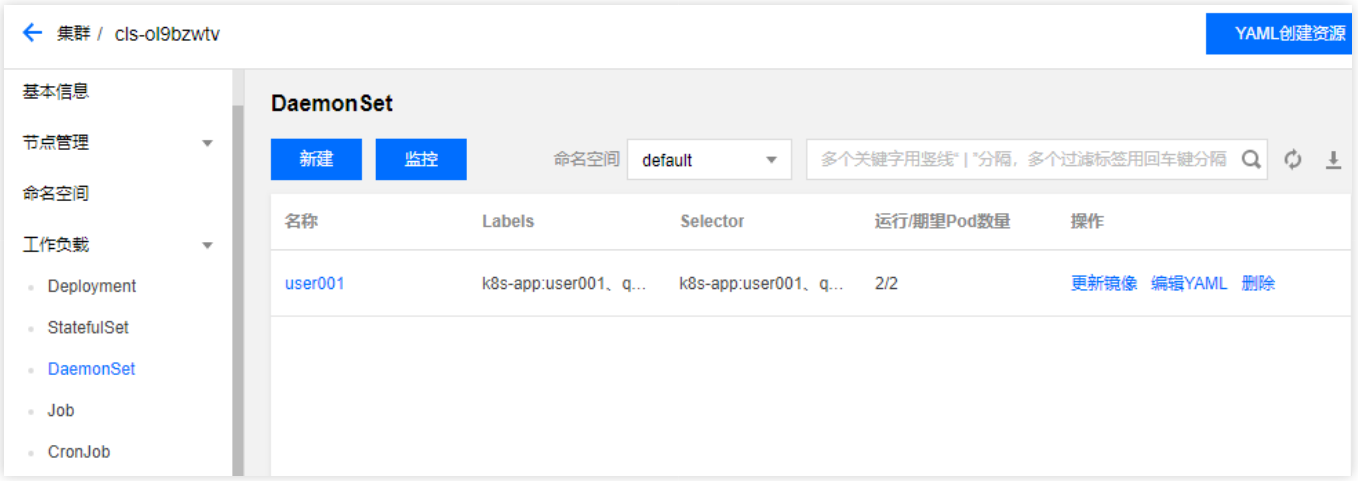
调度说明

若配置了 Pod 的 nodeSelector 或 affinity 参数，DaemonSet 管理的 Pod 将按照指定的调度规则调度。若未配置 Pod 的 nodeSelector 或 affinity 参数，则将在所有的节点上部署 Pod。

DaemonSet 控制台操作指引

创建 DaemonSet

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建 DaemonSet 的集群【ID/名称】，进入待创建 DaemonSet 的集群管理页面。
4. 选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。



5. 单击【新建】，进入“新建Workload”页面。
6. 根据实际需求，设置 DaemonSet 参数。关键参数信息如下：
   - 工作负载名：自定义。
   - 命名空间：根据实际需求进行选择。
   - 类型：选择 “DaemonSet（在每个主机上运行Pod）”。
   - 实例内容器：根据实际需求，为 DaemonSet 的一个 Pod 设置一个或多个不同的容器。
   - 名称：自定义。

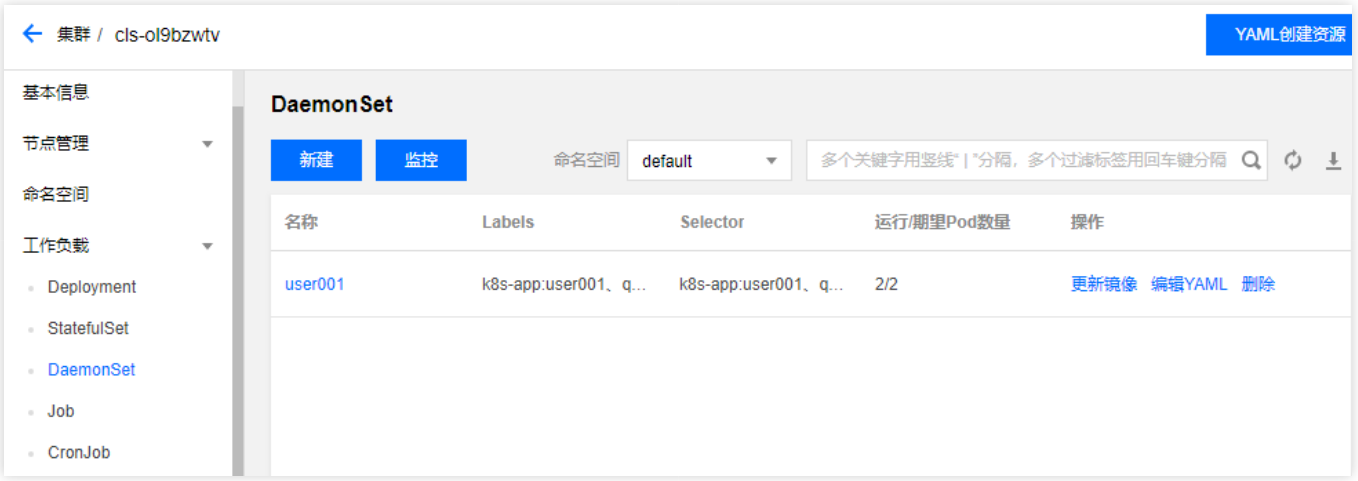
- 镜像：根据实际需求进行选择。
- 镜像版本：根据实际需求进行填写。
- CPU/内存限制：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
- 高级设置：可设置“工作目录”，“运行命令”，“运行参数”，“容器健康检查”，“特权级”等参数。

7. 单击【创建Workload】，完成创建。

更新 DaemonSet

更新 YAML

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 YAML 的集群【ID/名称】，进入待更新 YAML 的集群管理页面。
4. 选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。

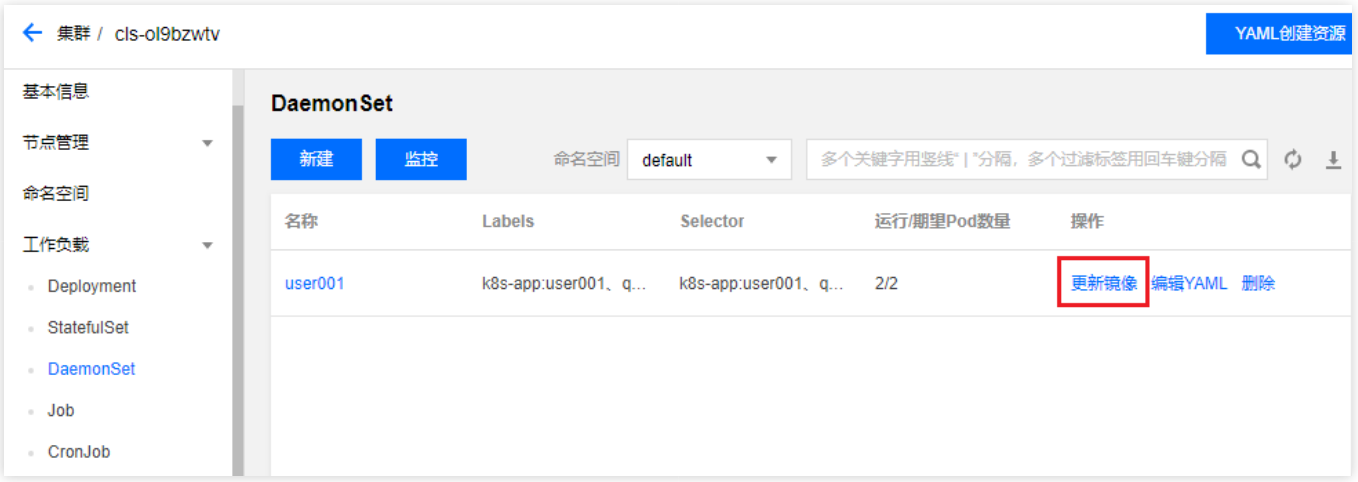


5. 在需要更新 YAML 的 DaemonSet 行中，单击【编辑YAML】，进入“更新DaemonSet”页面。
6. 在“更新DaemonSet”页面，编辑 YAML，单击【完成】，即可更新 YAML。

更新镜像

> 仅在 Kubernetes 1.6或更高版本中支持 DaemonSet 滚动更新功能。

1. 在集群管理页面，单击需要更新镜像的 DaemonSet 的集群【ID/名称】，进入待更新镜像的 DaemonSet 的集群管理页面。
2. 在需要更新镜像的 DaemonSet 行中，单击【更新镜像】。



3. 在“滚动更新镜像”页面，根据实际需求修改更新方式，设置参数。



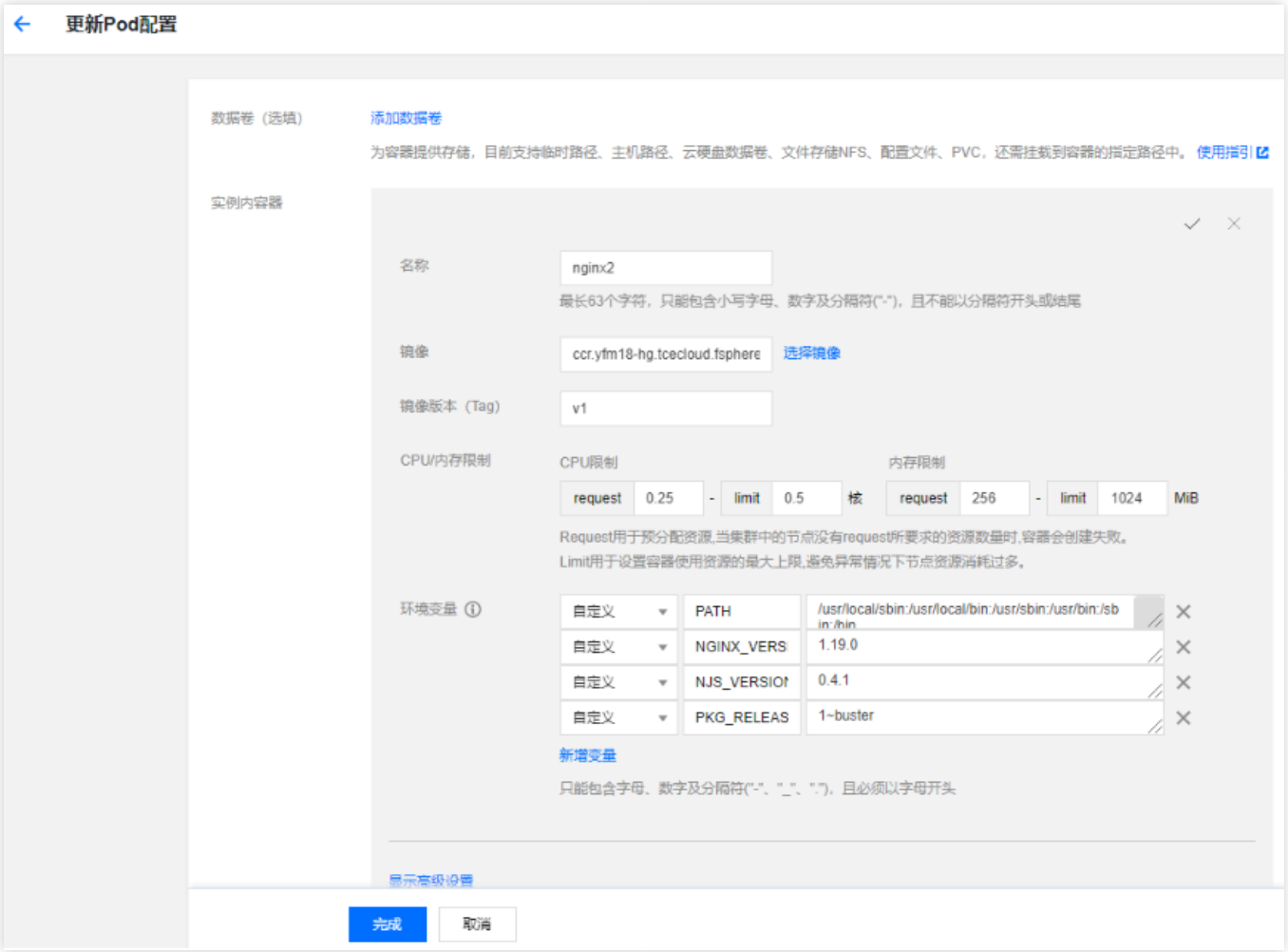
4. 单击【完成】，即可更新镜像。

更新Pod配置

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 在集群管理页面，单击需要更新镜像的 DaemonSet 的集群 ID，进入待更新Pod配置的 DaemonSet 的集群管理页面。
4. 在需要更新镜像的 DaemonSet 行中，单击【更新Pod配置】。



5. 在“更新Pod配置”页面，根据实际需求修改更新方式，设置参数。



6. 单击【完成】，即可更新 Pod 配置。

更新调度策略

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 DaemonSet 的集群【ID/名称】，进入待更新调度策略 DaemonSet 的集群管理页面。
4. 在需要更新调度策略 的 DaemonSet 行中，单击【更多】>【更新调度策略】，弹出“更新调度策略”窗口。
5. 在“更新调度策略”窗口，更新节点调度策略，单击【完成】。

Kubectl 操作 DaemonSet 指引

YAML 示例

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: fluentd-elasticsearch
        image: k8s.gcr.io/fluentd-elasticsearch:1.20
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
```

```
      mountPath: /var/lib/docker/containers
      readOnly: true
  terminationGracePeriodSeconds: 30
  volumes:
  - name: varlog
    hostPath:
      path: /var/log
  - name: varlibdockercontainers
    hostPath:
      path: /var/lib/docker/containers
```

> 注：以上YAML示例引用于 https://kubernetes.io/docs/concepts/workloads/controllers/daemonset， 创建时可能存在容器镜像拉取不成功的情况，仅供于本文介绍DaemonSet的组成。

- kind：标识 DaemonSet 资源类型。
- metadata：DaemonSet 的名称、Label等基本信息。
- metadata.annotations：DaemonSet 的额外说明，可通过该参数设置TCloudFinanceZone TKE 的额外增强能力。
- spec.template：DaemonSet 管理的 Pod 的详细模板配置。
  更多可查看 Kubernetes DaemonSet 官方文档

Kubectl 创建 DaemonSet

1. 参考 YAML 示例，准备 StatefulSet YAML 文件。

2. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。

3. 执行以下命令，创建 DaemonSet YAML 文件。

   ```
   kubectl create -f DaemonSet YAML 文件名称
   ```

   例如，创建一个文件名为 fluentd-elasticsearch.yaml 的 StatefulSet YAML 文件，则执行以下命令：

   ```
   kubectl create -f fluentd-elasticsearch.yaml
   ```

4. 执行以下命令，验证创建是否成功。

   ```
   kubectl get DaemonSet
   ```

   返回类似以下信息，即表示创建成功。

   ```
   NAME      DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE SELECTOR      AGE
   frontend  0         0         0         0            0           app=frontend-node  16d
   ```

Kubectl 更新 DaemonSet

执行以下命令，查看 DaemonSet 的更新策略类型。

```
kubectl get ds/<daemonset-name> -o go-template='{{.spec.updateStrategy.type}}{{"\n"}}'
```

DaemonSet 有以下两种更新策略类型：

- OnDelete：默认更新策略。该更新策略在更新 DaemonSet 后，需手动删除旧的 DaemonSet Pod 才会创建新的 DaemonSet Pod。
- RollingUpdate：支持 Kubernetes 1.6或更高版本。该更新策略在更新 DaemonSet 模板后，旧的 DaemonSet Pod 将被终止，并且以滚动更新方式创建新的 DaemonSet Pod。

方法一

执行以下命令，更新 DaemonSet。

```
kubectl edit DaemonSet/[name]
```

此方法适用于简单的调试验证，不建议在生产环境中直接使用。您可以通过此方法更新任意的 DaemonSet 参数。

方法二

执行以下命令，更新指定容器的镜像。

```
kubectl set image ds/[daemonset-name][container-name]=[container-new-image]
```

建议保持 DaemonSet 的其他参数不变，业务更新时，仅更新容器镜像。

Kubectl 回滚 DaemonSet

1. 执行以下命令，查看 DaemonSet 的更新历史。

   ```
   kubectl rollout history daemonset /[name]
   ```

2. 执行以下命令，查看指定版本详情。

   ```
   kubectl rollout history daemonset /[name] --revision=[REVISION]
   ```

3. 执行以下命令，回滚到前一个版本。

   ```
   kubectl rollout undo daemonset /[name]
   ```

如需指定回滚版本号，可执行以下命令。

```
kubectl rollout undo daemonset /[name] --to-revision=[REVISION]
```

Kubectl 删除 DaemonSet

执行以下命令，删除 DaemonSet。

```
kubectl delete  DaemonSet [NAME]
```

Job管理

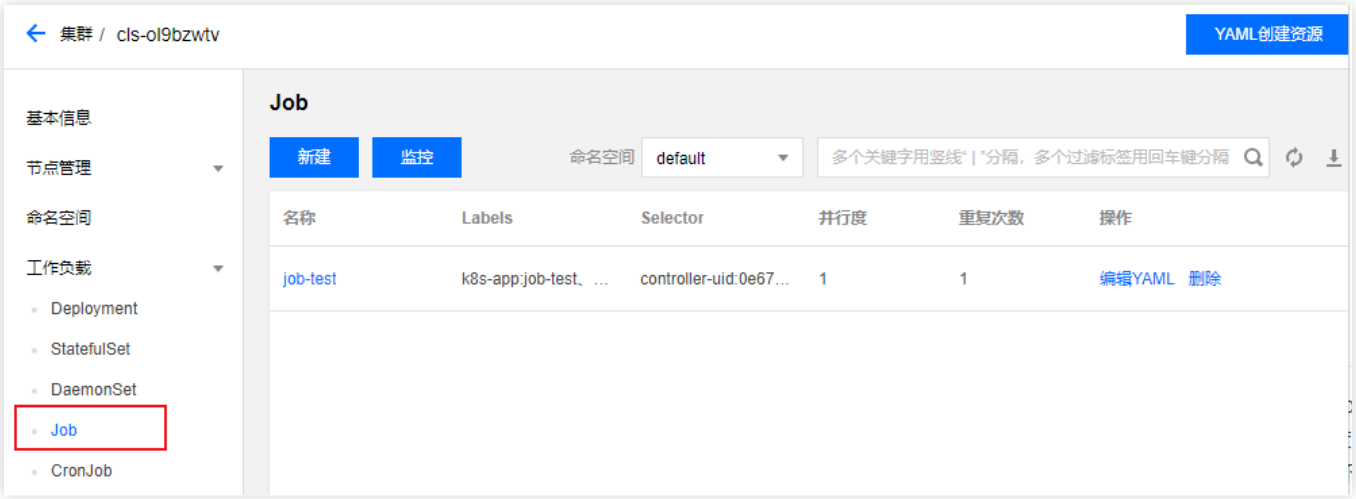
简介

Job 控制器会创建 1-N 个 Pod，这些 Pod 按照运行规则运行，直至运行结束。Job 可用于批量计算、数据分析等场景。通过设置重复执行次数、并行度、重启策略等满足业务述求。
Job 执行完成后，不再创建新的 Pod，也不会删除 Pod，您可在 “日志” 中查看已完成的 Pod 的日志。如果您删除了 Job，Job 创建的 Pod 也会同时被删除，将查看不到该 Job 创建的 Pod 的日志。

Job 控制台操作指引

创建 Job

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建 Job 的集群【ID/名称】，进入待创建 Job 的集群管理页面。
4. 选择【工作负载】 > 【Job】，进入“Job”信息页面。



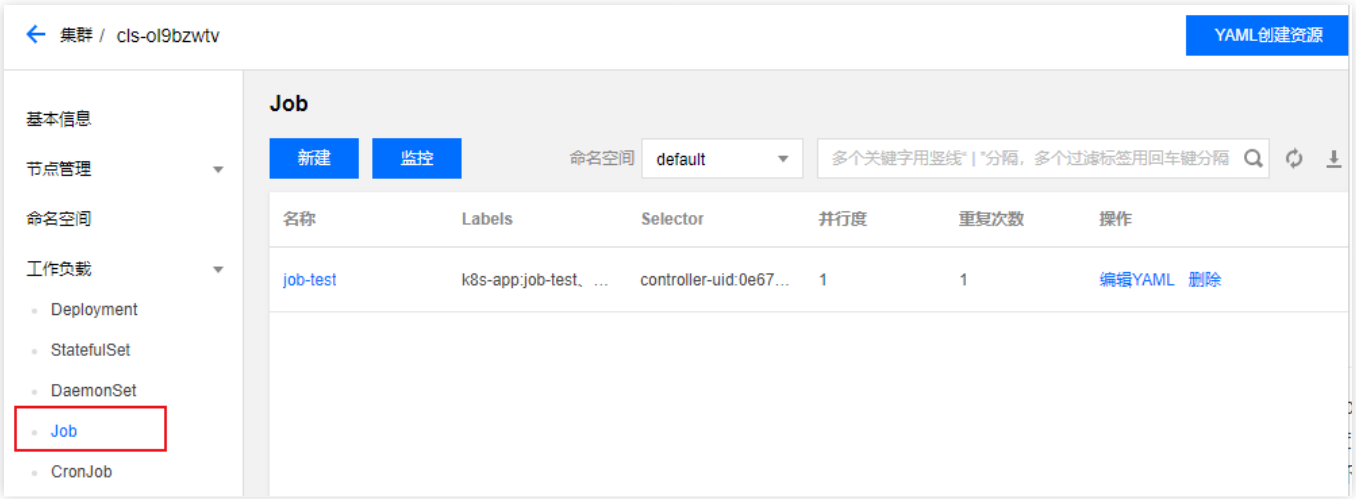
5. 单击【新建】，进入“新建Workload”页面。


新建Workload
工作负载名
请输入Workload名称
最长63个字符，只能包含小写字母、数字及分隔符("-")，且必须以小写字母开头，数字或小写字母结尾
描述
请输入描述信息，不超过1000个字符
标签
k8s-app
=
Value
新增变量
只能包含小写字母、数字及分隔符("-")，且必须以小写字母开头，数字或小写字母结尾
命名空间
default
类型
Deployment（可扩展的部署Pod）
DaemonSet（在每个主机上运行Pod）
StatefulSet（有状态集的运行Pod）
CronJob（按照Cron的计划定时运行）
Job（单次任务）
Job设置
重复次数
1
并行度
1
失败重启策略
OnFailure
数据卷（选填）
添加数据卷
为容器提供存储，目前支持主机路径、云硬盘数据卷、文件存储NFS、配置项、PVC、Secret，还需挂载到容器的指定路径中。
使用指引
实例内容器
名称
请输入容器名称
最长63个字符，只能包含小写字母、数字及分隔符("-")，且不能以分隔符开头或结尾
镜像
选择镜像
镜像版本（Tag）
CPU/内存限制
CPU限制
内存限制
request 0.25 - limit 0.5 核
request 256 - limit 1024 MiB
Request用于预分配资源,当集群中的节点没有request所要求的资源数量时,容器会创建失败。
Limit用于设置容器使用资源的最大上限,避免异常情况下节点资源消耗过多。
环境变量
新增变量 引用ConfigMap/Secret
变量名只能包含大小写字母、数字及下划线，并且不能以数字开头
显示高级设置
注意：Workload创建完成后，容器的配置信息可以通过更新YAML的方式进行修改
添加容器
显示高级设置
创建Workload
取消

6. 根据实际需求，设置 Job 参数。关键参数信息如下：
    - 工作负载名：自定义。
    - 命名空间：根据实际需求进行选择。
    - 类型：选择 “Job（单次任务）”。
    - Job设置：根据实际需求，为 Job 的一个 Pod 设置一个或多个不同的容器。
        - 重复次数：设置 Job 管理的 Pod 需要重复执行的次数。
        - 并行度：设置 Job 并行执行的 Pod 数量。
        - 失败重启策略：设置 Pod 下容器异常退出后的重启策略。
            - 选择 Never：不重启容器，直至 Pod 下所有容器退出。
            - 选择 OnFailure：Pod 继续运行，容器将重新启动。
    - 实例内容器：根据实际需求，为 Job 的一个 Pod 设置一个或多个不同的容器。
        - 名称：自定义。
        - 镜像：根据实际需求进行选择。
        - 镜像版本：根据实际需求进行填写。
        - CPU/内存限制：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
        - 高级设置：可设置 “工作目录”，“运行命令”，“运行参数”，“容器健康检查”，“特权级”等参数。
7. 单击【创建Workload】，完成创建。

查看 Job 状态

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要查看 Job 状态的集群【ID/名称】，进入待查看 Job 状态的集群管理页面。
4. 选择【工作负载】 > 【Job】，进入“Job”信息页面。



5. 单击需要查看状态的 Job 名称，即可查看 Job 详情。

删除 Job

Job 执行完成后，不再创建新的 Pod，也不会删除 Pod，您可在【日志】中查看已完成的 Pod 的日志。如果您删除了

Job，Job 创建的 Pod 也会同时被删除，将查看不到该 Job 创建的 Pod 的日志。

Kubectl 操作 Job 指引

YAML 示例

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  completions: 2
  parallelism: 2
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
  backoffLimit: 4
```

- kind：标识 Job 资源类型。
- metadata：Job 的名称、Label等基本信息。
- metadata.annotations：Job 的额外说明，可通过该参数设置TCloudFinanceZone TKE 的额外增强能力。
- spec.completions：Job 管理的 Pod 重复执行次数。
- spec.parallelism：Job 并行执行的 Pod 数。
- spec.template：Job 管理的 Pod 的详细模板配置。

创建 Job

1. 参考 YAML 示例，准备 Job YAML 文件。

2. 安装 Kubectl，并连接集群。

3. 创建 Job YAML 文件。

   ```
   kubectl create -f Job YAML 文件名称
   ```

   例如，创建一个文件名为 pi.yaml 的 Job YAML 文件，则执行以下命令：

```
kubectl create -f pi.yaml
```

4. 执行以下命令，验证创建是否成功。

```
kubectl get job
```

返回类似以下信息，即表示创建成功。

```
NAME      DESIRED   SUCCESSFUL   AGE
job       1         0            1m
```

删除 Job

执行以下命令，删除 Job。

```
kubectl delete job [NAME]
```

CronJob管理

简介

一个 CronJob 对象类似于 crontab（cron table）文件中的一行。它根据指定的预定计划周期性地运行一个 Job。
Cron 格式说明如下：

```
# 文件格式说明
#  ——分钟（0 - 59）
# |  ——小时（0 - 23）
# ||  ——日（1 - 31）
# |||  ——月（1 - 12）
# ||||  ——星期（0 - 7，星期日=0或7）
# |||||
# * * * * *
```

CronJob 控制台操作指引

创建 CronJob

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建 CronJob 的集群【ID/名称】，进入待创建 CronJob 的集群管理页面。
4. 选择【工作负载】 > 【CronJob】，进入“CronJob”信息页面。



5. 单击【新建】，进入“新建Workload”页面。


新建Workload
工作负载名
请输入Workload名称
最长63个字符，只能包含小写字母、数字及分隔符("-")，且必须以小写字母开头，数字或小写字母结尾
描述
请输入描述信息，不超过1000个字符
标签
k8s-app
=
Value
新增变量
只能包含小写字母、数字及分隔符("-")，且必须以小写字母开头，数字或小写字母结尾
命名空间
default
类型
Deployment（可扩展的部署Pod）
DaemonSet（在每个主机上运行Pod）
StatefulSet（有状态集的运行Pod）
CronJob（按照Cron的计划定时运行）
Job（单次任务）
执行策略
请输入执行策略，如: * * * * * command to ex
Job设置
重复次数
1
并行度
1
失败重启策略
OnFailure
数据卷（选填）
添加数据卷
为容器提供存储，目前支持主机路径、云硬盘数据卷、文件存储NFS、配置项、PVC、Secret，还需挂载到容器的指定路径中。
使用指引
实例内容器
名称
请输入容器名称
最长63个字符，只能包含小写字母、数字及分隔符("-")，且不能以分隔符开头或结尾
镜像
选择镜像
镜像版本（Tag）
CPU/内存限制
CPU限制
内存限制
request 0.25 - limit 0.5 核
request 256 - limit 1024 MiB
Request用于预分配资源,当集群中的节点没有request所要求的资源数量时,容器会创建失败。
Limit用于设置容器使用资源的最大上限,避免异常情况下节点资源消耗过多。
环境变量
新增变量 引用ConfigMap/Secret
变量名只能包含大小写字母、数字及下划线，并且不能以数字开头
显示高级设置
注意：Workload创建完成后，容器的配置信息可以通过更新YAML的方式进行修改
添加容器
显示高级设置
创建Workload
取消

6. 根据实际需求，设置 CronJob 参数。关键参数信息如下：
    - 工作负载名：自定义。
    - 命名空间：根据实际需求进行选择。
    - 类型：选择 “CronJob（按照Cron的计划定时运行）”。
    - 执行策略：根据 Cron 格式设置任务的定期执行策略。
    - Job设置
        - 重复次数：Job 管理的 Pod 需要重复执行的次数。
        - 并行度：Job 并行执行的 Pod 数量。
        - 失败重启策略：Pod下容器异常退出后的重启策略。
            - Never：不重启容器，直至 Pod 下所有容器退出。
            - OnFailure：Pod 继续运行，容器将重新启动。
    - 实例内容器：根据实际需求，为 CronJob 的一个 Pod 设置一个或多个不同的容器。
        - 名称：自定义。
        - 镜像：根据实际需求进行选择。
        - 镜像版本：根据实际需求进行填写。
        - CPU/内存限制：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
        - 高级设置：可设置 “工作目录”，“运行命令”，“运行参数”，“容器健康检查”，“特权级”等参数。
7. 单击【创建Workload】，完成创建。

更新 CronJob

更新Pod配置

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 在集群管理页面，单击需要更新镜像的 CronJob 的集群 ID，进入待更新Pod配置的 CronJob 的集群管理页面。
4. 在需要更新镜像的 CronJob 行中，单击【更新Pod配置】。
5. 在“更新Pod配置”页面，根据实际需求修改更新方式，设置参数。
6. 单击【完成】，即可更新 Pod 配置。

更新调度策略

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 CronJob 的集群【ID/名称】，进入待更新调度策略 CronJob 的集群管理页面。
4. 在需要更新调度策略 的 CronJob 行中，单击【更多】>【更新调度策略】，弹出“更新调度策略”窗口。
5. 在“更新调度策略”窗口，更新节点调度策略，单击【完成】。

查看 CronJob 状态

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要查看 CronJob 状态的集群【ID/名称】，进入待查看 CronJob 状态的集群管理页面。
4. 选择【工作负载】 > 【CronJob】，进入“CronJob”信息页面。如下图所示：



5. 单击需要查看状态的 CronJob 名称，即可查看 CronJob 详情。

Kubectl 操作 CronJob 指引

YAML 示例

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
```

```
    restartPolicy: OnFailure
```

- kind: 标识 CronJob 资源类型.
- metadata：CronJob 的名称、Label 等基本信息。
- metadata.annotations：对 CronJob 的额外说明，可通过该参数设置TCloudFinanceZone TKE 的额外增强能力。
- spec.schedule：CronJob 执行的 Cron 的策略。
- spec.jobTemplate：Cron 执行的 Job 模板。

创建 CronJob

方法一

1. 参考 YAML 示例，准备 CronJob YAML 文件。
2. 安装 Kubectl，并连接集群。
3. 执行以下命令，创建 CronJob YAML 文件。

   ```
   kubectl create -f CronJob YAML 文件名称
   ```

   例如，创建一个文件名为 cronjob.yaml 的 CronJob YAML 文件，则执行以下命令：

   ```
   kubectl create -f cronjob.yaml
   ```

方法二

1. 通过执行 kubectl run 命令，快速创建一个 CronJob。
   例如，快速创建一个不需要写完整配置信息的 CronJob，则执行以下命令：

   ```
   kubectl run hello --schedule="*/1 * * * *" --restart=OnFailure --image=busybox -- /bin/sh -c "date; ec
   ho Hello"
   ```

2. 执行以下命令，验证创建是否成功。

   ```
   kubectl get cronjob [NAME]
   ```

   返回类似以下信息，即表示创建成功。

```
NAME     SCHEDULE    SUSPEND   ACTIVE    LAST SCHEDULE   AGE
cronjob   * * * * *   False     0         <none>          15s
```

删除 CronJob

> - 执行此删除命令前，请确认是否存在正在创建的 Job，否则执行该命令将终止正在创建的 Job。
> - 执行此删除命令时，已创建的 Job 和已完成的 Job 均不会被终止或删除。
> - 如需删除 CronJob 创建的 Job，请手动删除。

执行以下命令，删除 CronJob。

```
kubectl delete cronjob [NAME]
```

设置工作负载的资源限制

请求（Request）与 限制（Limit）

Request：容器使用的最小资源需求，作为容器调度时资源分配的判断依赖。只有当节点上可分配资源量 >= 容器资源请求数时才允许将容器调度到该节点。但 Request 参数不限制容器的最大可使用资源值。
Limit： 容器能使用的资源最大值。

> 说明：
> 更多 Limit 和 Request 参数介绍，单击 查看详情。

CPU 限制说明

CPU 资源允许设置 CPU 请求和 CPU 限制的资源量，以核（U）为单位，允许为小数。

> 说明：
>
> - CPU Request 作为调度时的依据，在创建时为该容器在节点上分配 CPU 使用资源，称为“已分配 CPU”资源。
> - CPU Limit 限制容器 CPU 资源的上限，不设置表示不做限制（CPU Limit >= CPU Request）。

内存限制说明

内存资源只允许限制容器最大可使用内存量。以 MiB 为单位，允许为小数。

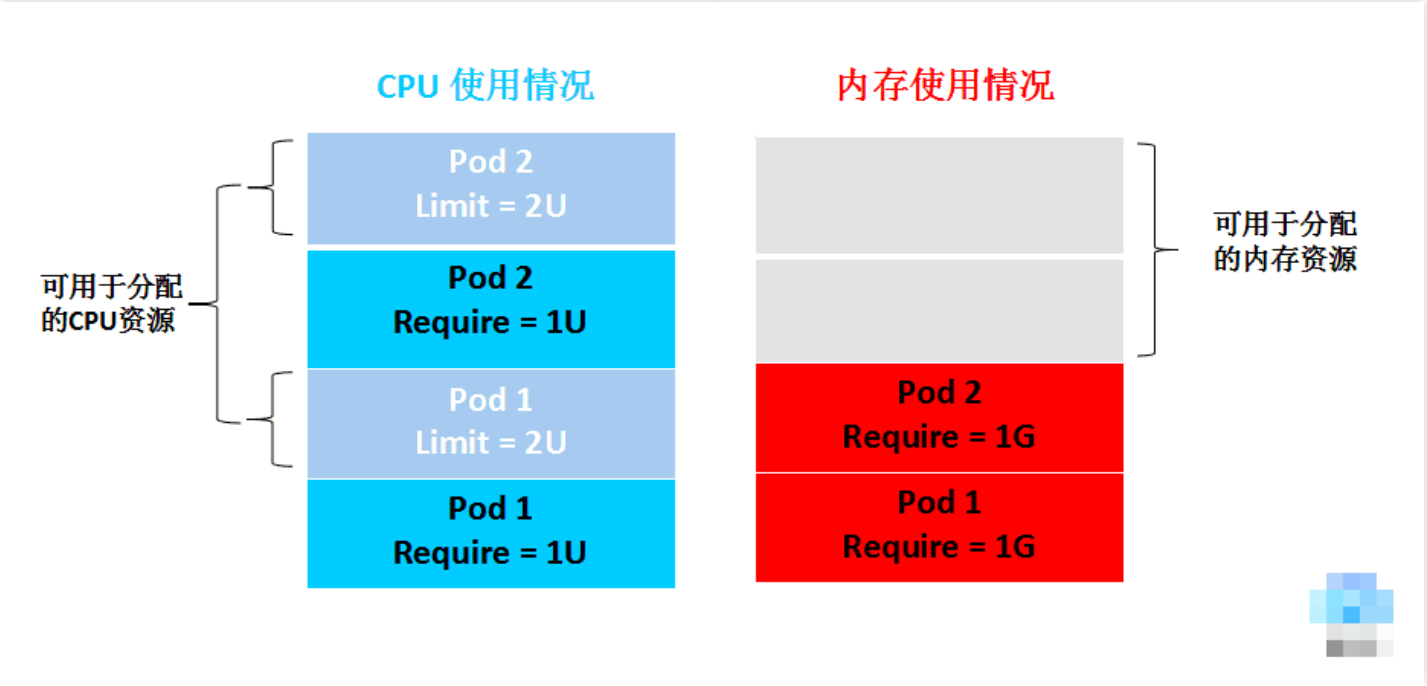
> 说明：
>
> - 内存 Request 作为调度时的依据，在创建时为该容器在节点上分配内存，称为“已分配内存”资源。
> - 内存资源为不可伸缩资源。当节点上所有容器使用内存均超量时，存在 OOM 的风险。因此不设置 Limit 时，默认 Limit = Request，保证容器的正常运作。

CPU 使用量和 CPU 使用率

- CPU 使用量为绝对值，表示实际使用的 CPU 的物理核数，CPU 资源请求和 CPU 资源限制的判断依据都是 CPU 使用量。
- CPU 使用率为相对值，表示 CPU 的使用量与 CPU 单核的比值（或者与节点上总 CPU 核数的比值）。

使用示例

一个简单的示例说明 Request 和 Limit 的作用，测试集群包括1个 4U4G 的节点、已经部署的两个 Pod（Pod1，Pod2），每个 Pod 的资源设置为（CPU Request，CPU Limit，Memory Request，Memory Limit）=（1U，2U，1G，1G）。（1.0G = 1000MiB）
节点上 CPU 和内存的资源使用情况如下图所示：



已经分配的 CPU 资源为：1U（分配 Pod1） + 1U（分配 Pod2） = 2U，剩余可以分配的 CPU 资源为2U。
已经分配的内存资源为：1G（分配 Pod1） + 1G（分配 Pod2） = 2G，剩余可以分配的内存资源为2G。
所以该节点可以再部署一个（CPU Request， Memory Request）=(2U，2G)的 Pod 部署，或者部署2个（CPU Request，Memory Request） = （1U，1G） 的 Pod 部署。

在资源限制方面，每个 Pod1 和 Pod2 使用资源的上限为（2U，1G），即在资源空闲的情况下，Pod 使用 CPU 的量最大能达到2U。

服务资源限制推荐

TKE 会根据您当前容器镜像的历史负载来推荐 Request 与 Limit 值，使用推荐值会保证您的容器更加平稳的运行，大大减小出现异常的概率。

推荐算法：
我们首先会取出过去7天当前容器镜像分钟级别负载，并辅以百分位统计第95%的值来最终确定推荐的 Request，Limit 为 Request 的2倍。

```
Request = Percentile(实际负载[7d],0.95)
Limit = Request * 2
```

如果当前的样本数量（实际负载）不满足推荐计算的数量要求，我们会相应的扩大样本取值范围，尝试重新计算。例如，去掉镜像 tag，namespace，serviceName 等筛选条件。若经过多次计算后同样未能得到有效值，则推荐值为空。

推荐值为空：
在使用过程中，您会发现有部分值暂无推荐的情况，可能由于以下几点造成：

1. 当前数据并不满足计算的需求，我们需要待计算的样本数量（实际负载）大于1440个，即有一天的数据。
2. 推荐值小于您当前容器已经配置的 Request 或者 Limit。

说明：

1. 由于推荐值是根据历史负载来计算的，原则上，容器镜像运行真实业务的时间越长，推荐的值越准确。
2. 使用推荐值创建服务，可能会因为集群资源不足造成容器无法调度成功。在保存时，须确认当前集群的剩余资源。
3. 推荐值是建议值，您可以根据自己业务的实际情况做相应的调整。

设置工作负载的调度规则

简介

通过设置工作负载中高级设置的调度规则，指定该工作负载下的 Pod 在集群内进行调度。 存在以下应用场景：

- 将 Pod 运行在指定的节点上。
- 将 Pod 运行在某一作用域（作用域可以是可用区、机型等属性）的节点上。
- 将 Pod 强制打散到节点上（每个节点一个，不符合条件的 Pod 停止调度）。

使用方法

前置条件

- 设置工作负载高级设置中的调度规则，且集群的 Kubernetes 版本必须是1.7以上的版本。
- 为确保您的 Pod 能够调度成功，请确保您设置的调度规则完成后，节点有空余的资源用于容器的调度。
- 使用自定义调度功能时，需要为节点设置对应 Label。详情请参见 设置节点 Label。

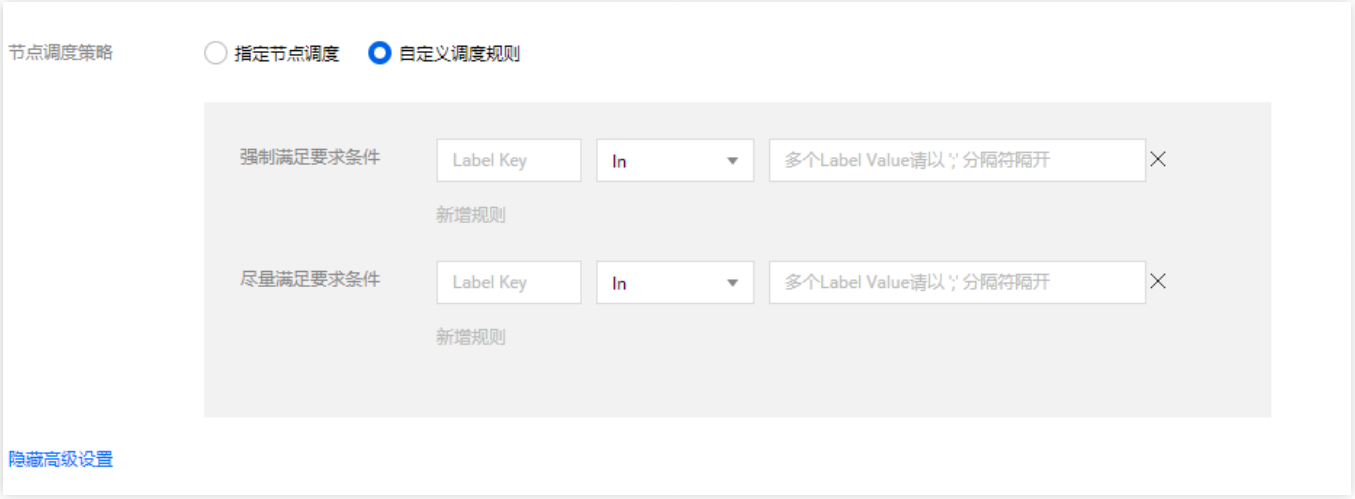
设置调度规则

如果您的集群是1.7或更高的版本，则可以在创建工作负载中设置调度规则。
您可以根据实际需求，选择以下两种调度类型：

- 指定节点调度：可设置实例（Pod）调度到指定规则的节点上，匹配节点标签。



- 自定义调度规则：可自定义实例（Pod）调度规则，匹配实例标签。



自定义调度规则包含以下两种模式：

- 强制满足要求条件：调度期间如果满足亲和性条件，则调度到对应 node。如果没有节点满足条件，则调度失败。
- 尽量满足要求条件：调度期间如果满足亲和性条件，则调度到对应 node。如果没有节点满足条件，则随机调度到任意节点。

自定义调度规则均可以添加多条调度规则， 各规则操作符的含义如下：

- In：Label 的 value 在列表中。
- NotIn：Label 的 value 不在列表中。
- Exists：Label 的 key 存在。
- DoesNotExits：Label 的 key 不存在。
- Gt：Label 的值大于列表值（字符串匹配）。
- Lt：Label 的值小于列表值（字符串匹配）。

原理介绍

服务的调度规则主要通过下发 Yaml 到 Kubernetes 集群， Kubernetes 的 Affinity and anti-affinity 机制会使得 Pod 按一定规则进行调度。更多 Kubernetes 的 Affinity and anti-affinity 机制介绍可 查看详情。

设置工作负载的健康检查

容器集群内核基于 Kubernetes。Kubernetes 支持对容器进行周期性探测，并根据探测结果判断容器的健康状态，执行额外的操作。

健康检查类别

健康检查分为以下类别：

- 容器存活检查：用于检测容器是否存活，类似于执行 ps 命令检查进程是否存在。如果容器的存活检查失败，集群会对该容器执行重启操作。如果容器的存活检查成功，则不执行任何操作。
- 容器就绪检查：用于检测容器是否准备好开始处理用户请求。例如，程序的启动时间较长时，需要加载磁盘数据或者要依赖外部的某个模块启动完成才能提供服务。此时，可通过容器就绪检查方式检查程序进程，确认程序是否启动完成。如果容器的就绪检查失败，集群会屏蔽请求访问该容器。如果容器的就绪检查成功，则会开放对该容器的访问。

健康检查方式

TCP 端口探测

TCP 端口探测的原理如下：
对于提供 TCP 通信服务的容器，集群周期性地对该容器建立 TCP 连接。如果连接成功，证明探测成功，否则探测失败。选择 TCP 端口探测方式，必须指定容器监听的端口。
例如，一个 redis 容器，它的服务端口是 6379。我们对该容器配置了 TCP 端口探测，并指定探测端口为 6379，那么集群会周期性地对该容器的 6379 端口发起 TCP 连接。如果连接成功，证明检查成功，否则检查失败。

HTTP 请求探测

HTTP 请求探测是针对于提供 HTTP/HTTPS 服务的容器，并集群周期性地对该容器发起 HTTP/HTTPS GET 请求。如果 HTTP/HTTPS response 返回码属于200 - 399范围，证明探测成功，否则探测失败。使用 HTTP 请求探测必须指定容器监听的端口和 HTTP/HTTPS 的请求路径。
例如，提供 HTTP 服务的容器，服务端口为 80，HTTP 检查路径为 /health-check ，那么集群会周期性地对容器发起 GET http://containerIP:80/health-check 请求。

执行命令检查

执行命令检查是一种强大的检查方式，该方式要求用户指定一个容器内的可执行命令，集群会周期性地在容器内执行

该命令。如果命令的返回结果是0，检查成功，否则检查失败。
对于 TCP 端口探测 和 HTTP 请求探测，都可以通过执行命令检查的方式来替代：

- 对于 TCP 端口探测，可以写一个程序对容器的端口进行 connect。如果 connect 成功，脚本返回0，否则返回-1。
- 对于 HTTP 请求探测，可以写一个脚本来对容器进行 wget 并检查 response 的返回码。例如， wget http://127.0.0.1:80/health-check 。如果返回码在200 - 399的范围，脚本返回 0，否则返回 -1。

> 注意：
>
> - 必须将需要执行的程序放在容器的镜像中，否则会因找不到程序而执行失败。
> - 如果执行的命令是一个 shell 脚本，不能直接指定脚本作为执行命令，需要加上脚本的解释器。例如，脚本是 /data/scripts/health_check.sh ，那么使用执行命令检查时，指定的程序为 sh /data/scripts/health_check.sh 。

其它公共参数

- 启动延时：单位秒。指定容器启动后，多久开始探测。例如，启动延时设置为5，那么健康检查将在容器启动5秒后开始。
- 间隔时间：单位秒。指定健康检查的频率。例如，间隔时间设置成10，那么集群会每隔10s检查一次。
- 响应超时：单位秒。指定健康探测的超时时间。对应到 TCP 端口探测、HTTP 请求探测、执行命令检查三种方式，分别表示 TCP 连接超时时间、HTTP 请求响应超时时间以及执行命令的超时时间。
- 健康阈值：单位次。指定健康检查连续成功多少次后，才判定容器是健康的。例如，健康阈值设置成3，则说明只有满足连续3次探测都成功，才认为容器是健康的。

  > 注意： 如果健康检查的类型为存活检查，那么健康阈值只能是1，用户设置成其它值将被视为无效。

- 不健康阈值：单位次。指定健康检查连续失败多少次后，才判定容器是不健康的。例如，不健康阈值设置成3，则说明只有满足连续3次都探测失败，才认为容器是不健康的。

设置工作负载的运行命令和参数

概述

创建工作负载时，通常通过镜像来指定实例中容器所运行的进程。在默认的情况下，镜像会运行默认的命令，如果您需要运行一个特定的命令或重写镜像的默认值，您需要使用到以下三个设置：

- 工作目录（workingDir）：指定当前的工作目录。
- 运行命令（command）：控制镜像运行的实际命令。
- 命令参数（args）：传递给运行命令的参数。

工作目录说明

workingDir，即指定当前的工作目录。如果不存在，则自动创建。如果没有指定，则使用容器运行时的默认值。如果镜像中如果没指定 WORKDIR，且在控制台未指定，则 workingDir 默认为 “/”。

命令和参数的使用

单击【详情】，了解如何将 docker run 命令适配到TCloudFinanceZone容器服务。

Docker 的镜像拥有存储镜像信息的相关元数据，如果不提供运行命令和参数，容器将会运行镜像制作时提供的默认的命令和参数。Docker 原生定义的字段为 “Entrypoint” 和 “CMD”。详情可查看 Docker 的 Entrypoint 说明 和 CMD 说明。
如果您在创建服务时，填写了容器的运行命令和参数，容器服务将会覆盖镜像构建时的默认命令（即 “Entrypoint” 和 “CMD”）。其规则如下：

镜像 Entrypoint	镜像 CMD	容器的运行命令	容器的运行参数	最终执行
[ls]	[/home]	未设置	未设置	[ls / home]
[ls]	[/home]	[cd]	未设置	[cd]
[ls]	[/home]	未设置	[/data]	[ls / data]
[ls]	[/home]	[cd]	[/data]	[cd / data]

> 说明：
> Docker entrypoint 对应容器服务控制台上的运行命令，Docker run 的 CMD 参数对应容器服务控制台上的运行参数。当有多个运行参数时，需在容器服务的运行参数中输入参数，且每个参数单独一行。

使用 TCR 企业版实例内容器镜像创建工作负载

操作场景

容器镜像服务企业版面向具有严格数据安全及合规性要求、业务分布在多个地域、集群规模庞大的企业级容器客户，提供企业级的独享镜像安全托管服务。相较于个人版服务，企业版支持容器镜像安全扫描、跨地域自动同步、Helm Chart 托管、网络访问控制等特性，详情请参见 容器镜像服务。
本文介绍如何在容器服务 TKE 中，使用容器镜像服务 TCR 内托管的私有镜像进行应用部署。

前提条件

在使用 TCR 内托管的私有镜像进行应用部署前，您需要完成以下准备工作：

- 已在容器镜像服务创建企业版实例。如尚未创建，请参考 创建企业版实例 完成创建。
- 如果使用子账号进行操作，请参考 企业版授权方案示例 提前为子账号授予对应实例的操作权限。

操作步骤

准备容器镜像

创建命名空间

新建的 TCR 企业版实例内无默认命名空间，且无法通过推送镜像自动创建。请参考 创建命名空间 按需完成创建。建议命名空间名使用项目或团队名，本文以 docker 为例。

创建镜像仓库（可选）

容器镜像托管在具体的镜像仓库内，请参考 创建镜像仓库 按需完成创建。镜像仓库名称请设置为期望部署的容器镜像名称，本文以 getting-started 为例。

> 说明
>
> 通过 docker cli 或其他镜像工具，例如 jenkins 推送镜像至企业版实例内时，若镜像仓库不存在，将会自动创建，无需提前手动创建。

推送容器镜像

您可通过 docker cli 或其他镜像构建工具，例如 jenkins 推送镜像至指定镜像仓库内，本文以 docker cli 为例。此步骤需要您使用一台安装有 Docker 的云服务器或物理机，并确保访问的客户端已在 配置网络访问策略 定义的公网或

内网允许访问范围内。

1. 参考 获取实例访问凭证 获取登录指令，并进行 Docker Login。

2. 登录成功后，您可在本地构建新的容器镜像或从 DockerHub 上获取一个公开镜像用于测试。
   本文以 DockerHub 官方的 Nginx 最新镜像为例，在命令行工具中依次执行以下指令，即可推送该镜像。请将 demo-tcr、docker 及 getting-started 依次替换为您实际创建的实例名称、命名空间名称及镜像仓库名。

   ```
   docker tag getting-started:latest demo-tcr.cloudcr.com/docker/getting-started:latest
   ```

   ```
   docker push demo-tcr.cloudcr.com/docker/getting-started:latest
   ```

   推送成功后，即可前往控制台的 镜像仓库 页面，选择仓库名进入详情页面查看。

配置 TKE 集群访问 TCR 实例

TCR 企业版实例支持网络访问控制，默认拒绝全部来源的外部访问。您可根据 TKE 集群的网络配置，选择通过公网或内网访问指定实例，拉取容器镜像。若 TKE 集群与 TCR 实例部署在同一地域，建议通过内网访问方式拉取容器镜像，可提升拉取速度，并节约公网流量成本。

使用 TCR 扩展组件进行快速配置（推荐）

1. 登录容器服务控制台，选择左侧导航栏中的集群。

2. 在集群管理页面，选择集群 ID，进入集群详情页。

3. 选择左侧菜单栏中的组件管理，在组件管理页面单击新建。

4. 在新建组件管理页面中勾选 TCR 组件。

   > 说明
   >
   > 当前 TCR 组件暂只支持 K8S 版本 ≥ 1.12 的集群，如集群版本暂不支持，请采用手动配置方式，或升级集群版本。

   - 单击查看详情了解组件功能及配置说明。
   - 单击参数配置开始配置组件。
     在 TCR 组件参数设置页面，参考查看详情中介绍的组件配置方式，配置相关参数。
   - 关联实例：选择与集群同地域的 TCR 实例。
   - 免密拉取配置：可采用默认配置。
   - 内网访问配置：可选功能，在 TCR 实例接入集群所在 VPC 并开启自动解析后，集群内节点可内网访问

TCR 实例，无需使用本功能。由于 TCR 侧自动解析功能依赖于 PrivateDNS，若当前集群所在地暂未支持 PrivateDNS 产品，可使用本配置实现内网访问。如内网访问链路中未展示为“链路正常”，请参考 内网访问控制，配置 TCR 实例与 TKE 集群所在私有网络 VPC 的内网链路。

5. 单击确定返回组件选择界面。

6. 在组件选择界面单击完成，开始为集群安装 TCR 扩展组件。
   组件安装完成后，集群将具备内网免密拉取该关联实例内镜像的能力，无需额外配置。

手动配置内网访问及访问凭证

1. 配置内网访问

   1. 参考 内网访问控制，配置 TCR 实例与 TKE 集群所在私有网络 VPC 的内网链路，并开启自动解析。
   2. 如当前 TCR 实例所在地域暂不支持开启自动解析，可在 TKE 集群中直接配置 TCR 实例的域名解析。请根据您的实际情况，选择以下方案：

      - 创建集群时配置节点 Host
        在创建 TKE 集群的“云服务器配置”步骤中，选择高级设置并在“节点启动配置”中输入如下内容：

        ```
        echo '172.21.17.69 demo.cloudcr.com' >> /etc/hosts
        ```

      - 为已有集群配置节点 Host

登录集群各个节点，并执行以下命令：

```bash
echo '172.21.17.69 demo.cloudcr.com' >> /etc/hosts
```

`172.21.17.69` 及 `demo.cloudcr.com` 请替换为您实际使用的内网解析 IP 及 TCR 实例域名。

2. 配置访问凭证

   1. 参考TCR - 服务级账号管理，获取服务级账号的用户名及密码。
   2. 在集群详情页，选择左侧的 配置管理 > Secret，进入 Secret 页面。
      在 Secret 页面单击新建进入新建 Secret 页面，参考以下信息下发访问凭证。
      主要参数信息如下：
      - Secret 类型：选择 Dockercfg (镜像仓库访问凭证)。
      - 生效范围：勾选需下发凭证的命名空间。
      - 镜像仓库域名：填写 TCR 实例的访问域名。
      - 用户名、密码：填写 步骤1 已获取的用户名及密码。

3. 单击创建 Secret即可完成下发。

使用 TCR 实例内容器镜像创建工作负载

1. 在集群详情页面，选择左侧工作负载> Deployment。
2. 在 Deployment 页面单击新建。
3. 在新建****Deployment 页面，参考以下信息创建工作负载。
   主要参数信息如下，其他参数请按需设置：
   - 命名空间：选择已下发访问凭证的命名空间。
   - 实例内容器：
     - 镜像：单击选择镜像，并在选择镜像页面，选择容器镜像服务企业版，再根据需要选择地域、实例和镜像仓库。
     - 镜像版本：选择好镜像后，单击选择镜像版本，在选择镜像版本页面，根据需要选择该镜像仓库的某个版本。若不选择则默认为 latest。
   - 镜像访问凭证：
     - 集群已安装 TCR 扩展组件：无需配置。
       集群未安装 TCR 扩展组件：选择添加镜像访问凭证，并选择 配置访问凭证 步骤中已下发的访问凭证。
4. 完成其他参数设置后，单击创建 Deployment后查看该工作负载的部署进度。
   部署成功后，可在 Deployment 页面查看该工作负载的“运行/期望Pod数量”为“1/1”。

自动伸缩
自动伸缩基本操作

操作场景

实例（Pod）自动扩缩容功能（Horizontal Pod Autoscaler，HPA）可以根据目标实例 CPU 利用率的平均值等指标自动扩展、缩减服务的 Pod 数量。本文介绍如何通过TCloudFinanceZone容器服务控制台实现 Pod 自动扩缩容。

工作原理

HPA 后台组件会每隔30秒向云监控拉取容器和 Pod 的监控指标，然后根据当前指标数据、当前副本数和该指标目标值进行计算，计算所得结果作为服务的期望副本数。当期望副本数与当前副本数不一致时，HPA 会触发 Deployment 进行 Pod 副本数量调整，从而达到自动伸缩的目的。
以 CPU 利用率为例，假设当前有2个实例， 平均 CPU 利用率（当前指标数据）为90%，自动伸缩设置的目标 CPU 为60%， 则自动调整实例数量为：90% × 2 / 60% = 3个。

> 如果用户设置了多个弹性伸缩指标，HPA 会依据各个指标，分别计算出目标副本数，取最大值进行扩缩容操作。

注意事项

- 当指标类型选择为 **CPU 利用率（占 Request）**时，必须为容器设置 CPU Request。
- 策略指标目标设置合理，例如设置70%给容器和应用，预留30%的余量。
- 保持 Pod 和 Node 健康（避免 Pod 频繁重建）。
- 保证用户请求的负载均衡稳定运行。
- HPA 在计算目标副本数时会有一个10%的波动因子。如果在波动范围内，HPA 并不会调整副本数目。
- 如果服务对应的 Deployment.spec.replicas 值为0，HPA 将不起作用。
- 如果对单个 Deployment 同时绑定多个 HPA ，则创建的 HPA 会同时生效，会造成的集群反复扩缩容。

前提条件

已创建集群。关于创建集群，详情请参见 创建集群。

操作步骤

开启自动扩缩容

可以通过以下三种方式开启自动扩缩容。

通过设置实例数量调节

1. 单击左侧导航栏中【集群】，进入“集群管理”页面。
2. 单击需要创建伸缩组的集群 ID，进入工作负载 Deployment 详情页，选择【新建】。



3. 在“新建Workload”页面，设置实例数量为自动调节。



   - 触发策略：自动伸缩功能依赖的策略指标。详情请参见 指标类型。
   - 实例范围：请根据实际需求进行选择，实例数量会在设定的范围内自动调节，不会超出该设定范围。

通过新建自动伸缩组

1. 单击左侧导航栏中【集群】，进入“集群管理”页面。
2. 单击需要创建伸缩组的集群 ID，进入工作负载 Deployment 详情页，选择【自动伸缩】。
3. 在“HorizontalPodAutoscaler”页面，单击【新建】。



4. 在“新建HPA”页面，根据以下提示，进行 HPA 配置。

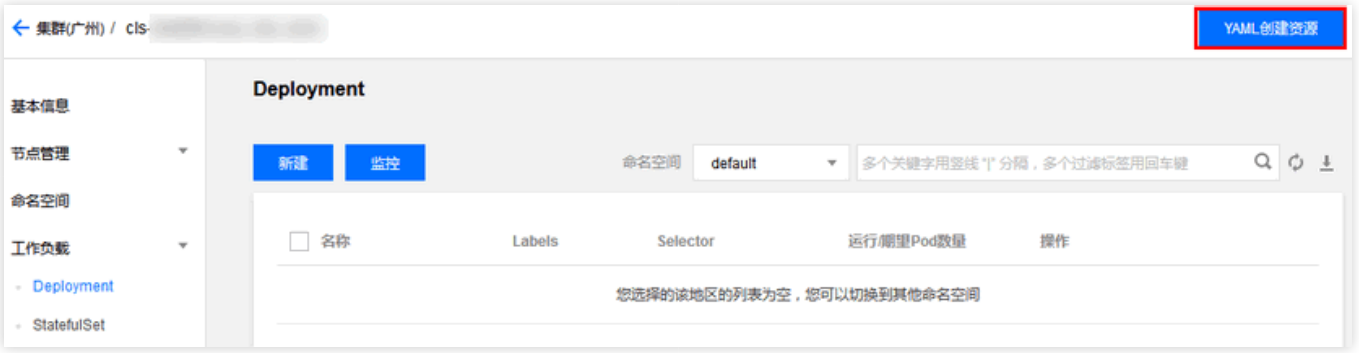


- 名称：输入要创建的自动伸缩组的名称。
- 命名空间：请根据实际需求进行选择。
- 工作负载类型：请根据实际需求进行选择。
- 关联工作负载：不能为空，请根据实际需求进行选择。
- 触发策略：自动伸缩功能依赖的策略指标，详情请参见 指标类型。
- 实例范围：请根据实际需求进行选择，实例数量会在设定的范围内自动调节，不会超出该设定范围。

5. 单击【创建HPA】，完成 HPA 的创建。

通过 YAML 创建

1. 单击左侧导航栏中【集群】，进入“集群管理”页面。
2. 单击需要创建伸缩组的集群 ID，进入工作负载 Deployment 详情页。
3. 单击该页面右上角【YAML创建资源】。



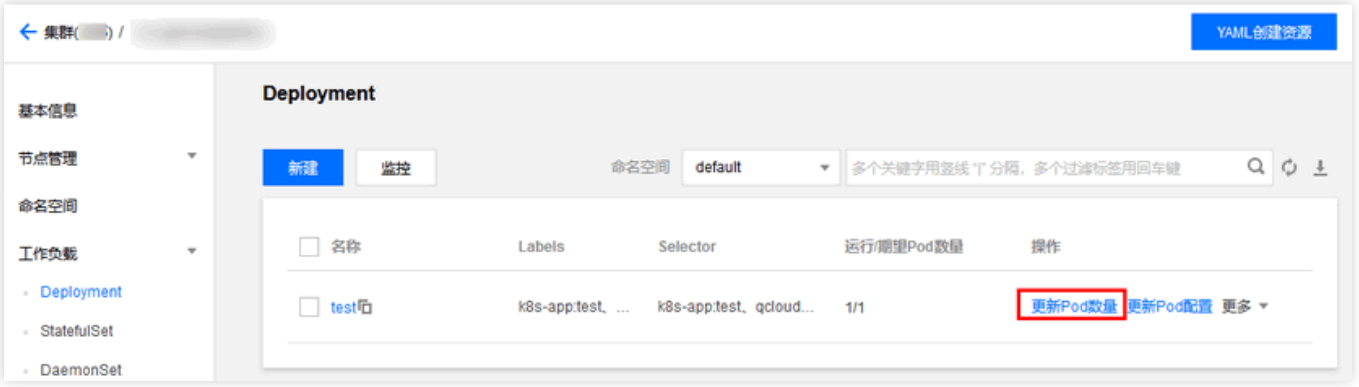
4. 在“YAML创建资源”页面，根据实际需求编辑内容，单击【完成】，即可新建 HPA 。

更新自动扩缩容规则

可以通过以下三种方式更新服务自动扩缩容规则。

通过更新 Pod 数量

1. 单击左侧导航栏中【集群】，进入“集群管理”页面。
2. 选择需要创建伸缩组的集群 ID，进入工作负载 Deployment 详情页，单击【更新Pod数量】。



3. 在“更新Pod数量”页面，根据实际需求进行设置，并单击【更新实例数目】。

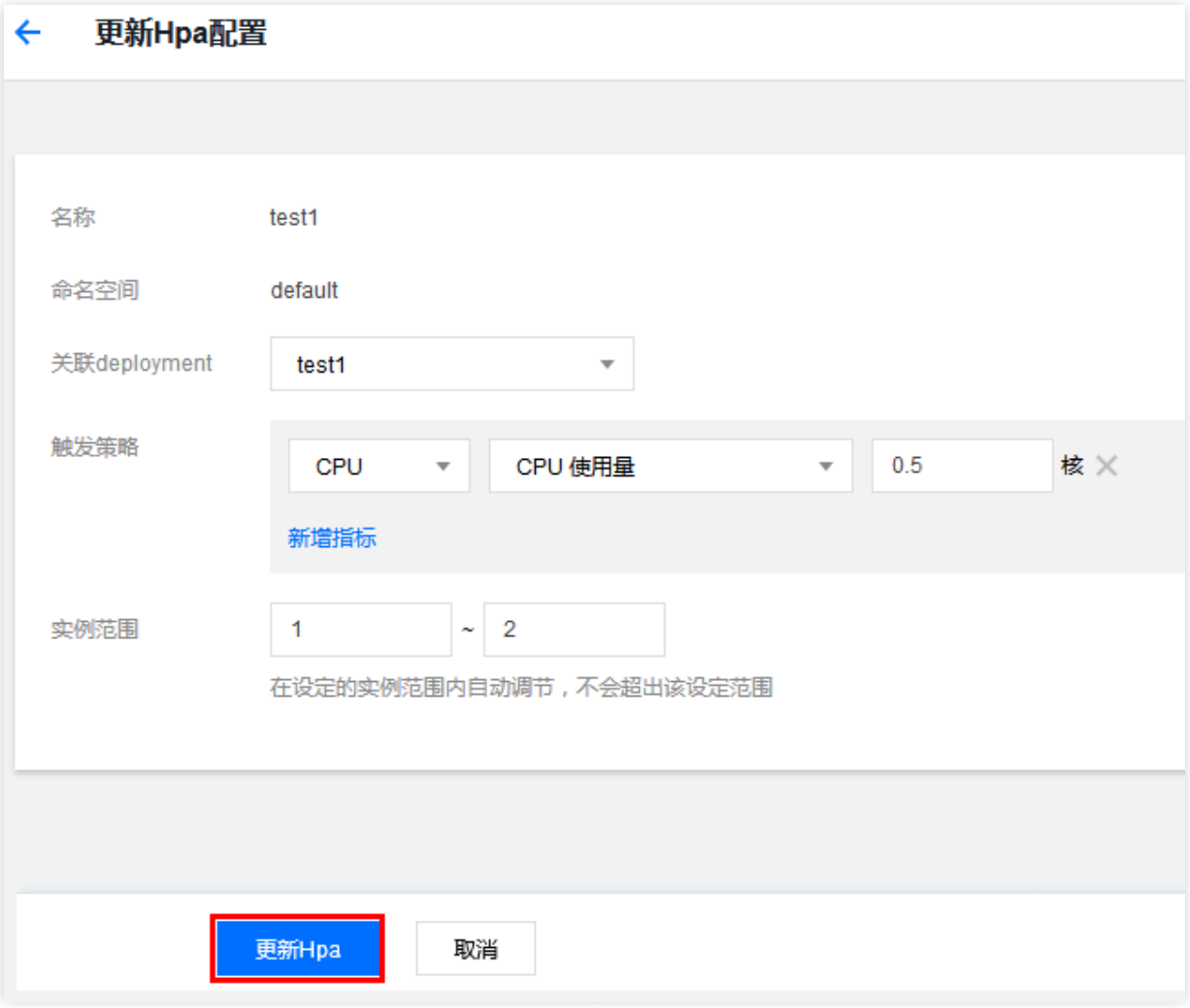


通过修改 Hpa 配置

1. 单击左侧导航栏中【集群】，进入“集群管理”页面。
2. 选择需要创建伸缩组的集群 ID，进入工作负载 Deployment 详情页，单击【自动伸缩】。
3. 在“HorizontalPodAutoscaler”页面，单击需要更新配置的 HPA 所在行右侧的【修改配置】。



4. 在“更新Hpa配置”页面，根据实际需求进行设置，并单击【更新Hpa】。



通过编辑 YAML 更新

1. 单击左侧导航栏中【集群】，进入“集群管理”页面。
2. 单击需要创建伸缩组的集群 ID，选择【自动伸缩】。
3. 在“HorizontalPodAutoscaler”页面，选择需要更新配置的 HPA 所在行右侧的【编辑YAML】。



4. 在“更新HorizontalPodAutoscaler”页面，根据实际需求进行编辑，单击【完成】即可。

指标类型

相关指标和类型请参见 自动伸缩指标说明。

自动伸缩指标说明

实例（Pod）自动扩缩容功能（Horizontal Pod Autoscaler，HPA）可以根据目标实例 CPU 利用率的平均值等指标自动扩展、缩减服务的 Pod 数量。自动扩缩容时，可供在控制台进行设置的触发指标类型包括 CPU 指标、内存、硬盘、网络和 GPU 相关指标。此外，这些指标还可以在您通过 YAML 文件创建和编辑 HPA 时使用，本文将给出配置 YAML 文件示例。

自动伸缩指标

自动伸缩指标详情如下表所示：

> 其中 metricName 中的变量本身有单位，即表中所示默认单位，该单位在编写 YAML 文件时可忽略。

CPU 指标

指标名称（控制台）	单位（控制台）	备注	type	metricName	默认单位
CPU 使用量	核	Pod 的 CPU 使用量	Pods	k8s_pod_cpu_core_used	核
CPU 利用率（占节点）	%	Pod 的 CPU 使用量占节点总量之比	Pods	k8s_pod_rate_cpu_core_used_node	%
CPU 利用率（占 Request）	%	Pod 的 CPU 使用量和设置的 Request 值之比	Pods	k8s_pod_rate_cpu_core_used_request	%
CPU 利用率（占 Limit）	%	Pod 的 CPU 使用量和设置的 Limit 值之比	Pods	k8s_pod_rate_cpu_core_used_limit	%

硬盘

指标名称（控制台）	单位（控制台）	备注	type	metricName	默认单位

指标名称（控制台）	单位（控制台）	备注	type	metricName	默认单位
硬盘写流量	KB/s	Pod 的硬盘写速率	Pods	k8s_pod_fs_write_bytes	B/s
硬盘读流量	KB/s	Pod 的硬盘读速率	Pods	k8s_pod_fs_read_bytes	B/s
硬盘读 IOPS	次/s	Pod 从硬盘读取数据的 IO 次数	Pods	k8s_pod_fs_read_times	次/s
硬盘写 IOPS	次/s	Pod 把数据写入硬盘的 IO 次数	Pods	k8s_pod_fs_write_times	次/s

网络

指标名称（控制台）	单位（控制台）	备注	type	metricName	默认单位
网络入带宽	Mbps	Pod 的入方向带宽之和	Pods	k8s_pod_network_receive_bytes_bw	Bps
网络出带宽	Mbps	Pod 的出方向带宽之和	Pods	k8s_pod_network_transmit_bytes_bw	Bps
网络入流量	KB/s	Pod 的入方向流量之和	Pods	k8s_pod_network_receive_bytes	B/s
网络出流量	KB/s	Pod 的出方向流量之和	Pods	k8s_pod_network_transmit_bytes	B/s
网络入包量	个/s	Pod 的入方向包数之和	Pods	k8s_pod_network_receive_packets	个/s
网络出包量	个/s	Pod 的出方向包数之和	Pods	k8s_pod_network_transmit_packets	个/s

内存

指标名称（控制台）	单位（控制台）	备注	type	metricName	默认单位

指标名称（控制台）	单位（控制台）	备注	type	metricName	默认单位
内存使用量	Mib	Pod 内存使用量	Pods	k8s_pod_mem_usage_bytes	B
内存使用量（不包含 Cache）	Mib	Pod 内存使用，不包含 Cache	Pods	k8s_pod_mem_no_cache_bytes	B
内存利用率（占节点）	%	Pod 内存使用占 node 的比例	Pods	k8s_pod_rate_mem_usage_node	%
内存利用率（占节点，不包含 Cache）	%	Pod 内存使用占 node 的比例，不含 Cache	Pods	k8s_pod_rate_mem_no_cache_node	%
内存利用率（占 Request）	%	Pod 内存使用占 Request 的比例	Pods	k8s_pod_rate_mem_usage_request	%
内存利用率（占 Request，不包含Cache	%	Pod 内存使用占 Request 的比例，不含 Cache	Pods	k8s_pod_rate_mem_no_cache_request	%
内存利用率（占 Limit）	%	Pod 内存使用占 limit 的比例	Pods	k8s_pod_rate_mem_usage_limit	%
内存利用率（占 Limit，不包含 Cache）	%	Pod 内存使用占 limit 的比例，不含 Cache	Pods	k8s_pod_rate_mem_no_cache_limit	%

GPU

指标名称	单位	备注	type	metricName	默认单位

（控制台）	（控制台）				
GPU 使用量	CUDA Core	Pod GPU 使用量	Pods	k8s_pod_gpu_used	CUDA Core
GPU 申请量	CUDA Core	Pod GPU 申请量	Pods	k8s_pod_gpu_request	CUDA Core
GPU 利用率（占 Request）	%	GPU 使用占 Request 的比例	Pods	k8s_pod_rate_gpu_used_request	%
GPU 利用率（占节点）	%	GPU 使用占 node 的比例	Pods	k8s_pod_rate_gpu_used_node	%
GPU memory 使用量	Mib	Pod GPU memory 使用量	Pods	k8s_pod_gpu_memory_used_bytes	B
GPU memory 申请量	Mib	Pod GPU memory 申请量	Pods	k8s_pod_gpu_memory_request_bytes	B
GPU memory 利用率（占 Request）	%	GPU memory 使用占 Request 的比例	Pods	k8s_pod_rate_gpu_memory_used_request	%
GPU memory 利用率（占节点）	%	GPU memory 使用占 node 的比例	Pods	k8s_pod_rate_gpu_memory_used_node	%

通过 YAML 创建和编辑 HPA

您可以通过 YAML 文件创建和编辑 HPA 。以下为配置文件的示例，该文件定义了一条名称为 example 的 HPA ，CPU 使用量为1时触发 HPA ，实例范围为1 - 2。

> TKE 同样兼容原生的 Resource 类型。

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
    name: example
    namespace: default
    labels:
      qcloud-app: example
spec:
    minReplicas: 1
    maxReplicas: 2
    metrics:
    - type: Pods    # 支持使用 Resource
      pods:
        metricName: k8s_pod_cpu_core_used
        targetAverageValue: "1"
    scaleTargetRef:
      apiVersion: apps/v1beta2
      kind: Deployment
      name: nginx
```

服务
Service管理

简介

Service 定义访问后端 Pod 的访问策略，提供固定的虚拟访问 IP。您可以通过 Service 负载均衡地访问到后端的 Pod。
Service 支持以下类型：

- 公网访问： 使用 Service 的 Loadbalance 模式，自动创建公网 CLB。 公网 IP 可直接访问到后端的 Pod。
- VPC内网访问：使用 Service 的 Loadbalance 模式，自动创建内网 CLB。指定 annotations:service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx ，VPC 内网即可通过内网 IP 直接访问到后端的 Pod。
- 集群内访问：使用 Service 的 ClusterIP 模式，自动分配 Service 网段中的 IP，用于集群内访问。

Service 控制台操作指引

注意事项

- 建议您的容器业务不要和 CVM 业务共用一个 CLB。
- 建议您不要在 CLB 控制台直接操作 TKE 自动管理的 CLB。
- 使用已有的 CLB 时，TKE 会自动覆盖 CLB 已有的后端 RS。
- TKE 会自动覆盖和更新名称为 TKE_Dedicated_Listener 的监听器，其他监听器不覆盖。

更多操作详见

Service 基本功能

Ingress管理

简介

Ingress 是允许访问到集群内 Service 的规则的集合，您可以通过配置转发规则，实现不同 URL 可以访问到集群内不同的 Service。
为了使 Ingress 资源正常工作，集群必须运行 Ingress-controller。TKE 服务在集群内默认启用了基于负载均衡器实现的 l7-lb-controller ，支持 HTTP、HTTPS，同时也支持 nginx-ingress 类型，您可以根据您的业务需要选择不同的 Ingress 类型。

Ingress 控制台操作指引

创建 Ingress

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建 Ingress 的集群【ID/名称】，进入待创建 Ingress 的集群管理页面。
4. 选择【服务】 > 【Ingress】，进入“Ingress”信息页面。
5. 单击【新建】，进入“新建Ingress”页面。
6. 根据实际需求，设置 Ingress 参数。关键参数信息如下：
    - Ingress名称：自定义。
    - 网络类型：默认为 “公网”，请根据实际需求进行选择。
    - 命名空间：根据实际需求进行选择。
    - 监听端口：默认为 “Http:80”，请根据实际需求进行选择。
    - 转发配置：根据实际需求进行设置。
7. 单击【创建Ingress】，完成创建。

更新 Ingress

更新 YAML

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 YAML 的集群【ID/名称】，进入待更新 YAML 的集群管理页面。
4. 选择【服务】 > 【Ingress】，进入“Ingress”信息页面。
5. 在需要更新 YAML 的 Ingress 行中，单击【编辑YAML】，进入更新 Ingress 页面。
6. 在“更新Ingress”页面，编辑 YAML，单击【完成】，即可更新 YAML。

更新转发规则

1. 集群管理页面，单击需要更新 YAML 的集群【ID/名称】，进入待更新 YAML 的集群管理页面。
2. 选择【服务】 > 【Ingress】，进入“Ingress”信息页面。
3. 在需要更新转发规则的 Ingress 行中，单击【更新转发配置】，进入“更新转发配置”页面。
4. 根据实际需求，修改转发配置，单击【更新转发配置】，即可完成更新。

kubectl 操作 Ingress 指引

YAML 示例

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: qcloud ## 可选值：qcloud（CLB类型ingress），nginx（nginx-ingress）
    ## kubernetes.io/ingress.subnetId: subnet-xxxxxxxx  ##若是创建CLB类型内网ingress需指定该条annotation
  name: my-ingress
  namespace: default
spec:
  rules:
  - host: localhost
    http:
      paths:
      - backend:
          serviceName: non-service
          servicePort: 65535
        path: /
```

- kind：标识Ingress资源类型。
- metadata：Ingress 的名称、Label等基本信息。
- metadata.annotations: Ingress 的额外说明，可通过该参数设置TKE 的额外增强能力。
- spec.rules：Ingress 的转发规则，配置该规则可实现简单路由服务、基于域名的简单扇出路由、简单路由默认域名、配置安全的路由服务等。

  > 如果您使用的是带宽上移账号，在创建公网访问方式的服务时需要指定以下两个 annotations 项：
  >
  > - kubernetes.io/ingress.internetChargeType 公网带宽计费方式，TRAFFIC_POSTPAID_BY_HOUR（按使用流量计费），BANDWIDTH_POSTPAID_BY_HOUR（按

带宽计费）。
- kubernetes.io/ingress.internetMaxBandwidthOut 带宽上限，范围：[1,2000] Mbps。

例如：

```
metadata:
  annotations:
    kubernetes.io/ingress.internetChargeType: TRAFFIC_POSTPAID_BY_HOUR
    kubernetes.io/ingress.internetMaxBandwidthOut: "10"
```

创建 Ingress

1. 参考 YAML 示例，准备 Ingress YAML 文件。
2. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
3. 执行以下命令，创建 Ingress YAML 文件。

   ```
   kubectl create -f Ingress YAML 文件名称
   ```

   例如，创建一个文件名为 my-ingress.yaml 的 Ingress YAML 文件，则执行以下命令：

   ```
   kubectl create -f my-ingress.yaml
   ```

4. 执行以下命令，验证创建是否成功。

   ```
   kubectl get ingress
   ```

   返回类似以下信息，即表示创建成功。

   ```
   NAME          HOSTS       ADDRESS   PORTS     AGE
   clb-ingress   localhost             80        21s
   ```

更新 Ingress

方法一

执行以下命令，更新 Ingress。

```
kubectl edit  ingress/[name]
```

方法二

1. 手动删除旧的 Ingress。
2. 执行以下命令，重新创建 Ingress。

```
kubectl create/apply
```

Service 管理
概述

Service 基本概念

用户在 Kubernetes 中可以部署各种容器，其中一部分是通过 HTTP、HTTPS 协议对外提供七层网络服务，另一部分是通过 TCP、UDP 协议提供四层网络服务。而 Kubernetes 定义的 Service 资源就是用来管理集群中四层网络的服务访问。
Kubernetes 的 ServiceTypes 允许指定 Service 类型，默认为 ClusterIP 类型。ServiceTypes 的可取值以及行为描述如下：

可取值	说明
ClusterIP	通过集群的内部 IP 暴露服务。当您的服务只需要在集群内部被访问时，请使用该类型。该类型为默认的 ServiceType。
NodePort	通过每个集群节点上的 IP 和静态端口（NodePort）暴露服务。NodePort 服务会路由到 ClusterIP 服务，该 ClusterIP 服务会自动创建。通过请求 <NodeIP>:<NodePort> ，可从集群的外部访问该 NodePort 服务。除了测试以及非生产环境以外，不推荐在生产环境中直接通过集群节点对外甚至公网提供服务。从安全上考虑，使用该类型会直接暴露集群节点，容易受到攻击。通常认为集群节点是动态的、可伸缩的，使用该类型使得对外提供服务的地址和集群节点产生了耦合。
LoadBalancer	使用负载均衡器，可以向公网或者内网暴露服务。负载均衡器可以路由到 NodePort 服务，或直接转发到处于 VPC-CNI 网络条件下的容器中。

ClusterIP 和 NodePort 类型的 Service，在不同云服务商或是自建集群中的行为表现通常情况下相同。而 LoadBalancer 类型的 Service，由于使用了云服务商的负载均衡进行服务暴露，云服务商会围绕其负载均衡的能力提供不同的额外功能。

服务访问方式

根据上述 ServiceTypes 定义，您可以使用容器服务 TKE 提供的以下四种服务访问方式：

访问方式	Service 类型	说明
公网	LoadBalancer	使用 Service 的 LoadBalancer 模式，公网 IP 可直接访问到后端的 Pod，适用于 Web 前台类的服务。 创建完成后的服务在集群外可通过负载均衡域名或 IP + 服务端口访问服务，集群内可通过服务名 + 服务端口访问服务。

访问方式	Service 类型	说明
VPC 内网	LoadBalancer	使用 Service 的 LoadBalancer 模式，指定注解 service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx ，即可通过内网 IP 直接访问到后端的 Pod。 创建完成后的服务在集群外可通过负载均衡域名或 IP + 服务端口访问服务，集群内可通过服务名 + 服务端口访问服务。
主机端口访问	NodePort	提供一个主机端口映射到容器的访问方式，支持 TCP、UDP、Ingress。可用于业务定制上层 LB 转发到 Node。 创建完成后的服务可以通过云服务器 IP + 主机端口访问服务。
仅集群内访问	ClusterIP	使用 Service 的 ClusterIP 模式，自动分配 Service 网段中的 IP，用于集群内访问。数据库类等服务如 MySQL 可以选择集群内访问，以保证服务网络隔离。 创建完成后的服务可以通过服务名 + 服务端口访问服务。

负载均衡相关概念

Service 工作原理

容器集群中的 Service Controller 组件负责用户 Service 资源的同步。当用户创建、修改或删除 Service 资源时、集群节点或 Service Endpoints 出现变化时、组件容器发生飘移重启时，组件都会对用户的 Service 资源进行同步。Service Controller 会依照用户 Service 资源的描述创建对应的负载均衡资源，并对监听器及其后端进行配置。当用户删除集群 Service 资源时，也会回收对应负载均衡资源。

Service 生命周期管理

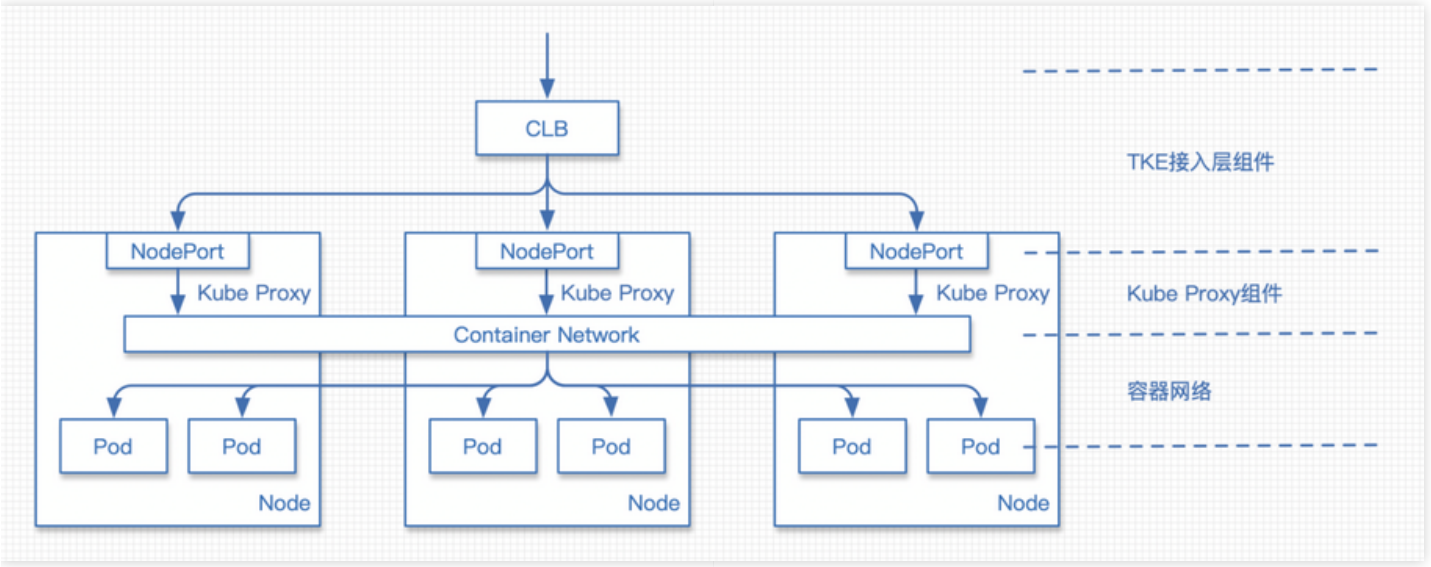
Service 对外服务的能力依赖于负载均衡所提供的资源，服务资源管理也是 Service 的重要工作之一。Service 在资源的生命周期管理中会使用以下标签：

- tke-createdBy-flag = yes ：标识该资源是由容器服务创建。
- tke-clusterId = <ClusterId> ：标识该资源被哪一个 Cluster 所使用的。
- tke-lifecycle-owner = tke|user : 标识 CLB 的控制权是 TKE 还是用户，如果值是 user，删除 Service 时不会删除负载均衡。

  > 说明：
  >
  > - tke-createdBy-flag 和 tke-clusterId 这两个标签对用户只读，请不要修改或删除这些标签，否则可能导致资源泄漏。
  > - 若用户使用了已有负载均衡，则 Service 仅会使用该负载均衡，而不会删除该负载均衡。
  > - 若用户将 tke-lifecycle-owner 置为 user，或者使用 私有连接，则删除 Service 时，不会删除该负载均衡。

当 LoadBalancer 类型的 Service 集群资源被创建时，对应负载均衡的生命周期就开始了。直到 Service 资源被删除

或是负载均衡被重建时，负载均衡的生命周期就结束了。在此期间负载均衡会持续根据 Service 资源的描述进行同步。**当用户切换 Service 的网络访问时，例如公网 > Cluster IP、VPC 内网 > Cluster IP、Cluster IP 切换成使用的已有负载均衡，此类操作都会涉及到负载均衡的删除或销毁。**LoadBalancer 类型 Service 工作原理如下所示：



Service 注意事项

- 请勿在负载均衡控制台手动删除 Service 所关联的 CLB 实例，也不能手动修改 CLB 的配置，一切 CLB 相关操作都应在 TKE 侧进行。
- Service 有一个字段： .spec.externalTrafficPolicy 。kube-proxy 基于 spec.internalTrafficPolicy 的设置来过滤路由的目标服务端点。当它的值设为 Local 时，只会选择节点本地的服务端点。当它的值设为 Cluster 或缺省时，Kubernetes 会选择所有的服务端点。

Service 基本功能

控制台操作指引

创建 Service

1. 登录容器服务控制台，选择左侧导航栏中的集群。
2. 在集群管理页面，单击需要创建 Service 的集群 ID，进入集群基本信息页。
3. 选择服务 > Service，在 Service 页面单击新建。
4. 在新建 Service 页面，根据实际需求，设置 Service 参数。关键参数信息如下：
   - 服务名称：自定义。
   - 命名空间：根据实际需求进行选择。
   - 访问设置：请参考 服务访问方式说明 进行设置。
   - Workload 绑定：引用一个存量的 Workload，或自定义标签，该 Service 会根据自定义的标签选择拥有这些标签的 Workload。
5. 单击创建 Service，完成创建。

更新 Service

> 说明：
>
> 为防止 Service 在切换不同的 CLB 时发生资源异常问题，例如：CLB 可能会脱离 TKE 的管控导致资源泄漏，或无法创建出对应的 CLB 导致服务中断，TKE 对 Service 的生命周期的变更做了如下限制：
>
> 1. 禁止服务访问方式中的公网 LB 访问与内网 LB 访问之间互相切换。
> 2. 选择公网 LB 访问时，禁止切换当前 VPC 的可用区，以及其他 VPC；禁止在负载均衡器中的自动创建与使用已有之间互相切换。
> 3. 选择内网 LB 访问时，禁止在负载均衡器中的自动创建与使用已有之间互相切换，禁止切换 LB 所在子网。
> 4. 选择使用已有时，禁止切换成其他已有 CLB。
> 5. 禁止传统型和应用型 CLB 之间的切换。
>   若您尝试切换以上动作，前台会做限制，后台会拦截。若您有以上切换诉求，可以尝试：
> 6. 先将服务访问方式切换至仅在集群内访问或主机端口访问。
> 7. 再配置成目标的参数配置。

更新配置

1. 登录容器服务控制台，选择左侧导航栏中的集群。
2. 在集群管理页面，单击集群 ID，进入集群基本信息页。
3. 选择服务 > Service，在 Service 页面单击 Service 所在行右侧的更新访问方式。
4. 在更新访问方式页面，根据实际需求进行访问设置。
5. 设置完成后，单击更新访问方式即可。

编辑 YAML

1. 选择服务 > Service，在 Service 页面单击 Service 所在行右侧的编辑YAML。
2. 在编辑Yaml 页面，根据实际需求编辑 YAML 后单击完成即可。

删除 Service

1. 登录容器服务控制台，选择左侧导航栏中的集群。
2. 在集群管理页面，单击集群 ID，进入集群基本信息页。
3. 选择服务> Service，在 Service 页面单击 Service 所在行右侧的删除。

Kubectl 操作 Service 指引

YAML 示例

```
kind: Service
apiVersion: v1
metadata:
  ## annotations:
  ##   service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx ##若是创建内网访问的 Service 需指定该条 annotation
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

说明：

- kind：标识 Service 资源类型。
- metadata：Service 的名称、Label 等基本信息。
- metadata.annotations：Service 的额外说明，可通过该参数设置容器服务的额外增强能力。

- spec.selector：该 Service 会根据这里标签选择器里的标签，选择拥有这些标签的 Workload。
- spec.type：标识 Service 的被访问形式。
  - ClusterIP：在集群内部公开服务，可用于集群内部访问。
  - NodePort：使用节点的端口映射到后端 Service，集群外可以通过节点 IP:NodePort 访问。
  - LoadBalancer：使用云平台提供的负载均衡器公开服务，默认创建公网负载均衡，指定 annotations 可创建内网负载均衡。
  - ExternalName：将服务映射到 DNS，仅适用于 kube-dns1.7及更高版本。

创建 Service

1. 参考YAML 示例，准备 Service YAML 文件。
2. 安装 Kubectl，并连接集群。操作详情请参见 通过 Kubectl 连接集群。
3. 执行以下命令，创建 Service YAML 文件。

   ```
   kubectl create -f Service YAML 文件名称
   ```

   例如，创建一个文件名为 my-service.yaml 的 Service YAML 文件，则执行以下命令：

   ```
   kubectl create -f my-service.yaml
   ```

4. 执行以下命令，验证创建是否成功。

   ```
   kubectl get services
   ```

   返回类似以下信息，即表示创建成功。

   ```
   NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
   kubernetes   ClusterIP   172.16.255.1   <none>        443/TCP   38d
   ```

更新 Service

方法1

执行以下命令，更新 Service。

```
kubectl edit service/[name]
```

方法2

1. 手动删除旧的 Service。
2. 执行以下命令，重新创建 Service。

```
kubectl create/apply
```

删除 Service

执行以下命令，删除 Service。

```
kubectl delete service [NAME]
```

Service 负载均衡配置

TkeServiceConfig

TkeServiceConfig 是容器服务提供的自定义资源 CRD， 通过 TkeServiceConfig 能够帮助您更灵活的配置 LoadBalancer 类型的 Service ，及管理其中负载均衡的各种配置。

使用场景

Service YAML 的语义无法定义的负载均衡的参数和功能，可以通过 TkeServiceConfig 进行配置。

配置说明

使用 TkeServiceConfig 能够帮您快速进行负载均衡器的配置。通过 Service 注解 service.finance.cloud.tencent.com/tke-service-config:<config-name> ，您可以指定目标配置并应用到 Service 中。

> 注意：
>
> TkeServiceConfig 资源需要与 Service 处于同一命名空间。

TkeServiceConfig 并不会帮您直接配置并修改协议和端口，您需要在配置中描述协议和端口以便指定配置下发的监听器。在一个 TkeServiceConfig 中可以声明多组监听器配置，目前主要针对负载均衡的健康检查以及对后端访问提供配置。
通过指定协议和端口，配置能够被准确的下发到对应监听器：

- spec.loadBalancer.l4Listeners.protocol ：四层协议
- spec.loadBalancer.l4Listeners.port ：监听端口

Service 与 TkeServiceConfig 关联行为

1. 创建 Loadbalancer 模式 Service 时，设置注解 service.finance.cloud.tencent.com/tke-service-config-auto: "true" ，将自动创建 -auto-service-config。您也可以通过 service.finance.cloud.tencent.com/tke-service-config: 直接指定您自行创建的 TkeServiceConfig。两个注解不可同时使用，且手动指定的 <config-name> 不能以 -auto-service-config 与 -auto-ingress-config 为后缀。
2. 其中自动创建的 TkeServiceConfig 存在以下同步行为：
    - 更新 Service 资源时，新增若干四层监听器时，如果该监听器或转发规则没有对应的 TkeServiceConfig 配置片段。Service-Controller 将主动添加 TkeServiceConfig 对应片段。
    - 删除若干四层监听器时，Service-controller 组件将主动删除 TkeServiceConfig 对应片段。
    - 删除 Service 资源时，联级删除该 TkeServiceConfig。
    - 用户修改 Service 默认的 TkeServiceConfig，TkeServiceConfig 内容同样会被应用到负载均衡。

3. 您也可以参考下列 TkeServiceConfig 完整配置参考自行创建需要的 CLB 配置，Service 通过注解：service.finance.cloud.tencent.com/tke-service-config: 引用该配置。
4. 其中您手动创建的 TkeServiceConfig 存在以下同步行为：
   - 当用户在 Service 中添加配置注解时，负载均衡将会立即进行设置同步。
   - 当用户在 Service 中删除配置注解时，负载均衡将会保持不变。
   - 修改 TkeServiceConfig 配置时，引用该配置 Service 的负载均衡将会根据新的 TkeServiceConfig 进行设置同步。
   - Service 的监听器未找到对应配置时，该监听器将不会进行修改。
   - Service 的监听器找到对应配置时，若配置中没有声明的属性，该监听器将不会进行修改。

完整配置参考

```
apiVersion: finance.cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
  name: sample # 配置的名称
  namespace: default # 配置的命名空间
spec:
  loadBalancer:
    l4Listeners: # 四层规则配置，适用于Service的监听器配置。
    - protocol: TCP # 协议端口锚定Service的四层规则。必填，枚举值：TCP|UDP|TCP_SSL|QUIC。
      port: 80 # 必填，可选值：1~65535。
      deregisterTargetRst: true # 选填，布尔值。双向 RST 开关，建议非直连类型 Service 启用，非直连 Service 的连接会经过 NodePort 中转，不启用双向 RST 可能导致下线集群节点后造成业务中断。
      session: # 会话保持相关配置。选填
        enable: true # 是否开启会话保持。必填，布尔值
        sessionExpireTime: 100 # 会话保持的时间。选填，默认值：30，可选值：30~3600，单位：秒。
      healthCheck: # 健康检查相关配置。选填
        enable: true # 是否开启健康检查。必填，布尔值
        checkType: "TCP" # 健康检查类型。选填，枚举值：TCP|HTTP|CUSTOM（仅适用于TCP/UDP监听器，其中UDP监听器只支持CUSTOM；如果使用自定义健康检查功能，则必传）。
        checkPort: 80 # 健康检查端口，选填。默认为后端服务的端口，除非您希望指定特定端口，否则建议留空。（仅适用于TCP/UDP监听器）。
        intervalTime: 10 # 健康检查探测间隔时间。选填，默认值：5，可选值：5~300，单位：秒。
        healthNum: 2 # 健康阈值，表示当连续探测几次健康则表示该转发正常。选填，默认值：3，可选值：2~10，单位：次。
        unHealthNum: 3 # 不健康阈值，表示当连续探测几次健康则表示该转发异常。选填，默认值：3，可选值：2~10，单位：次。
        timeout: 10 # 健康检查的响应超时时间，响应超时时间要小于检查间隔时间。选填，默认值：2，可选值：2~60，单位：秒。
        httpCode: 31 # 健康检查状态码，选填，默认值：31，可选值：1~31。仅适用于HTTP/HTTPS转发规则、TCP监听器的HTTP健康检查方式。 1 表示探测后返回值 1xx 代表健康，2 表示返回 2xx 代表健康，4 表示返回 3xx 代表健康，8 表示返回 4xx 代表健康，16 表示返回 5xx 代表健康。若希望多种返回码都可代表健康，
```

则将相应的值相加。
httpCheckPath: "/" # 健康检查路径，选填。仅适用于HTTP/HTTPS转发规则、TCP监听器的HTTP健康检查方式。
httpCheckDomain: "" # 健康检查域名，选填。默认为七层规则域名（仅适用于HTTP/HTTPS转发规则、TCP监听器的HTTP健康检查方式）。
httpCheckMethod: "HEAD" # 健康检查方法（仅适用于HTTP/HTTPS转发规则、TCP监听器的HTTP健康检查方式），默认值：HEAD，可选值HEAD或GET。
httpVersion: "HTTP/1.1" # 自定义探测相关参数。健康检查协议CheckType的值取HTTP时，必传此字段，代表后端服务的HTTP版本：HTTP/1.0、HTTP/1.1；（仅适用于TCP监听器）。
sourceIpType: 0 # 健康检查探测来源。0(VIP为源IP) 1(100.64为源IP)。对于域名化clb默认值为1且只能为1，对于非域名化的clb默认值不一定，可在clb控制台配置页面看能否看到VIP探测方式，如能看到默认值为0，否则为1。
scheduler: WRR # 请求转发方式配置。WRR、LEAST_CONN 分别表示按权重轮询、最小连接数。选填，枚举值：WRR|LEAST_CONN。
proxyProtocol:
enable: false # 启用 ProxyProtocol，仅对 TCP_SSL 和 QUIC 协议有效，CLB 申请开通白名单后可用，默认关闭。

示例

Deployment 示例：jetty-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: jetty
  name: jetty-deployment
  namespace: default
spec:
  progressDeadlineSeconds: 600
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: jetty
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
```

```
    labels:
      app: jetty
  spec:
    containers:
    - image: jetty:9.4.27-jre11
      imagePullPolicy: IfNotPresent
      name: jetty
      ports:
      - containerPort: 80
        protocol: TCP
      - containerPort: 443
        protocol: TCP
      resources: {}
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    terminationGracePeriodSeconds: 30
```

Service 示例：jetty-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.finance.cloud.tencent.com/tke-service-config: jetty-service-config
    # 指定已有的 tke-service-config
    # service.finance.cloud.tencent.com/tke-service-config-auto: "true"
    # 自动创建 tke-service-config
  name: jetty-service
  namespace: default
spec:
  ports:
  - name: tcp-80-80
    port: 80
    protocol: TCP
    targetPort: 80
  - name: tcp-443-443
    port: 443
    protocol: TCP
    targetPort: 443
  selector:
    app: jetty
  type: LoadBalancer
```

该示例中包含以下配置：

- Service 为公网 LoadBalancer 类型。声明了两个 TCP 服务，一个在80端口，一个在443端口。
- 使用了 jetty-service-config 负载均衡配置。

TkeServiceConfig 示例：jetty-service-config.yaml

```
apiVersion: finance.cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
  name: jetty-service-config
  namespace: default
spec:
  loadBalancer:
    l4Listeners:
    - protocol: TCP
      port: 80
      deregisterTargetRst: true
      healthCheck:
        enable: false
    - protocol: TCP
      port: 443
```

```
    session:
      enable: true
      sessionExpireTime: 3600
    healthCheck:
      enable: true
      intervalTime: 10
      healthNum: 2
      unHealthNum: 2
      timeout: 5
    scheduler: WRR
```

该示例中包含以下配置：

名称为 jetty-service-config 。且在四层监听器配置中，声明了以下两段配置：

1. 80端口的 TCP 监听器将会被配置。关闭健康检查。
2. 443端口的 TCP 监听器将会被配置。
    - 打开健康检查，健康检查间隔调整为10s，健康阈值2次，不健康阈值2次，超时5s。
    - 打开会话保持功能，会话保持的超时时间设置为3600s。
    - 转发策略配置为：按权重轮询。

kubectl 配置命令

```
$ kubectl apply -f jetty-deployment.yaml
$ kubectl apply -f jetty-service.yaml
$ kubectl apply -f jetty-service-config.yaml

$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
jetty-deployment-8694c44b4c-cxscn   1/1     Running   0          8m8s
jetty-deployment-8694c44b4c-mk285   1/1     Running   0          8m8s
jetty-deployment-8694c44b4c-rjrtm   1/1     Running   0          8m8s

$ kubectl get service jetty-service
NAME    TYPE           CLUSTER-IP       EXTERNAL-IP       PORT(S)                    AGE
jetty   LoadBalancer   10.127.255.209   150.158.220.237   80:31338/TCP,443:32373/TCP   2m47s

# 获取TkeServiceConfig配置列表
$ kubectl get tkeserviceconfigs.finance.cloud.tencent.com
NAME                 AGE
jetty-service-config   52s

# 更新修改TkeServiceConfig配置
$ kubectl edit tkeserviceconfigs.finance.cloud.tencent.com jetty-service-config
tkeserviceconfig.finance.cloud.tencent.com/jetty-service-config edited
```

Service 使用已有 CLB

容器服务 TKE 具备通过 service.kubernetes.io/tke-existed-lbid: <LoadBalanceId> 注解实现使用已有负载均衡的功能，您可使用该注解指定集群 Service 资源关联的负载均衡实例。还提供了 Service 负载均衡复用功能，即指定多个 Service 使用同一个已有负载均衡，您可参考本文进行设置。

使用已有负载均衡的同步行为

- 使用已有负载均衡时，指定 Service 的网络类型的注解不生效。
- 当 Service 不再使用已有负载均衡时，该 Service 描述的对应监听器会删除，该负载均衡将保留。
  删除监听器时，会校验监听器名称是否被修改。如果用户修改监听器名称，则认为该监听器可能由用户创建，不进行主动删除。
- 如果 Service 目前正在使用自动创建的负载均衡，那么给它添加使用已有负载均衡的注解，会使得当前负载均衡的生命周期结束并释放，Service 的配置将会与该负载均衡进行同步。反之，如果删除 Service 正在使用的已有负载均衡的注解，Service Controller 组件将会为该 Service 创建负载均衡并进行同步。

使用已有负载均衡同步标签行为

- 默认情况下，Service 创建的 CLB 均会配置 tke-createdBy-flag = yes 标签，Service 会在销毁时删除对应资源。若使用已有 CLB，则不会配置该标签，Service 销毁时也不会删除对应资源。
- 所有 Service 均会配置 tke-clusterId = 标签，若 ClusterId 正确，则 Service 会在销毁时删除对应标签。
- 于2020年8月17日起创建的集群，将默认关闭多个 Service 复用相同 CLB 的功能。该日期前后集群内 Service 创建的 CLB 标签配置规则变更情况及详细信息，请参见 多 Service 复用 CLB。

注意事项

- 指定使用的负载均衡需和集群处于同一 VPC。
- 请确保您的容器业务不和云服务器 CVM 业务共用一个负载均衡。
- 不支持您在负载均衡控制台操作 TKE 所管理负载均衡的监听器和后端绑定的服务器，您的更改会被 TKE 的自动同步所覆盖。
- 使用已有的负载均衡时：
  - Service Controller 将不负责该已有负载均衡的释放与回收。
  - 仅支持使用通过负载均衡控制台创建的负载均衡器，不支持复用由 TKE 自动创建的负载均衡，会破坏其他 Service 负载均衡的生命周期管理。
- 复用负载均衡时：
  - 不支持跨集群复用负载均衡。

- 您需要使用复用功能时，建议有明确的监听器端口管理，否则负载均衡在多个 Service 的使用下，会出现管理混乱。
- 复用负载均衡的端口冲突时，将会被拒绝。如果在修改中出现冲突，那么出现冲突的监听器后端同步无法确保正确。
- 复用负载均衡的 Service 不支持开启 Local 访问（传统型负载均衡限制）。
- 删除 Service，则复用负载均衡绑定的后端云服务器需要自行解绑，同时会保留一个 tag tke-clusterId: cls-xxxx ，需自行清理。

Service 示例

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
  name: nginx-service
spec:
  ports:
    - name: 80-80-no
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

> 说明
>
> - service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx 注解表示该 Service 将使用已有负载均衡进行配置。
> - 请注意 Service 的类型，需设置为 LoadBalancer 类型。

使用场景 示例

在同一端口暴露 TCP 和 UDP 服务

Kubernetes 官方在 Service 的设计中具有限制：一个 Service 下暴露的多个端口协议必须相同。有许多游戏场景下的用户，有在同一个端口同时暴露 TCP 和 UDP 服务的需求，负载均衡服务支持在同一个端口上同时监听 UDP 和 TCP 协议，此需求可以通过 Service 负载均衡复用来解决。
例如以下 Service 配置， game-service 被描述为两个 Service 资源，描述的内容除了监听的协议以外基本相同。两个 Service 都通过注解指定使用已有负载均衡 lb-6swtxxxx 。通过 kubectl 将以上资源应用到集群中，就可以实现

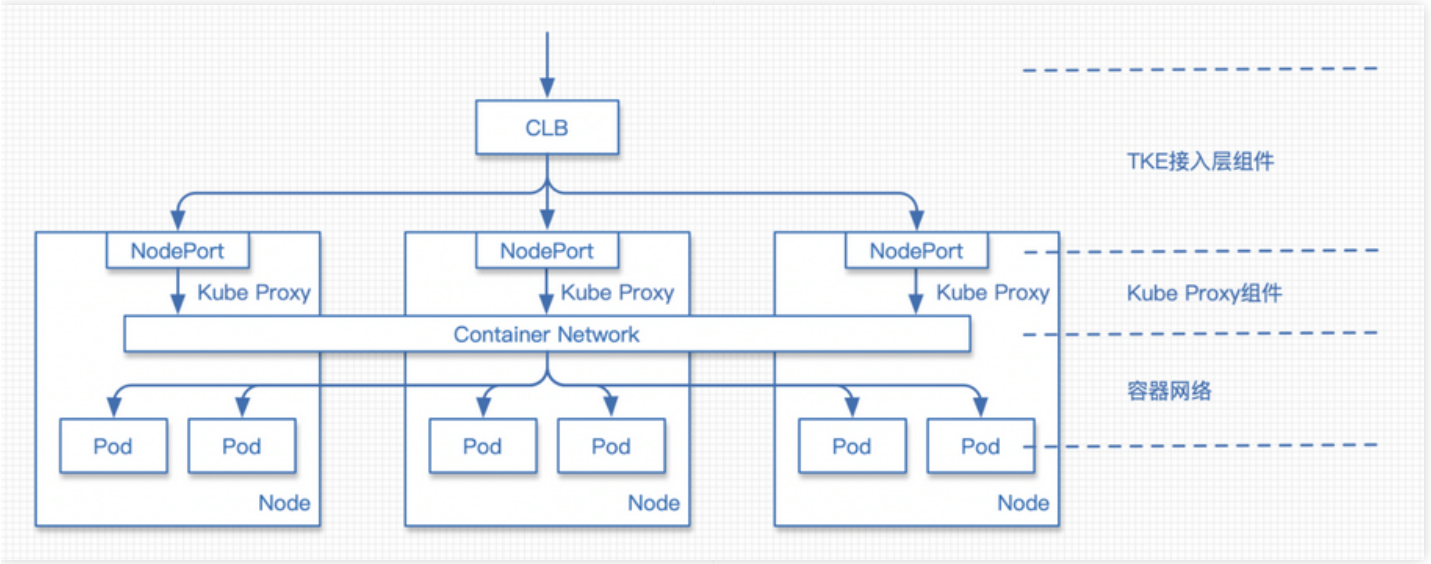
在同一个负载均衡的端口上暴露多种协议的目的。

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
  name: game-service-a
spec:
  ports:
    - name: 80-80-tcp
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: game
  type: LoadBalancer
---
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
  name: game-service-b
spec:
  ports:
    - name: 80-80-udp
      port: 80
      protocol: UDP
      targetPort: 80
  selector:
    app: game
  type: LoadBalancer
```

Service 后端选择

默认后端选择

默认情况下，Service 会配置负载均衡的后端到集群节点的 NodePort，如下图 TKE 接入层组件部分。此方案具有非常高的容错性，流量从负载均衡到任何一个 NodePort 之后，NodePort 会再一次随机选择一个 Pod 将流量转发过去。同时这也是 Kubernetes 官方提出的最基础的网络接入层方案。如下图所示：



TKE Service Controller 默认不会将以下节点作为负载均衡后端：

- Master 节点（不允许 Master 节点参与网络接入层的负载）。
- 节点状态为 NotReady （节点不健康）。

> 注意
>
> TKE Service Controller 可以绑定状态为 Unschedulable 的节点。Unschedulable 的节点也可以作为流量的入口，因为流量进入到节点之后，会再做一层容器网络里的流量转发，流量在 Unschedulable 的节点里面不会被丢弃，如上图所示。

指定接入层后端

对于一些规模很大的集群，Service 管理的负载均衡会挂载几乎所有集群节点的 NodePort 作为后端。此场景存在以下问题：

- 负载均衡的后端数量有数量限制。
- 负载均衡会对每一个 NodePort 进行健康检查，所有健康检查都会请求到后端的工作负载上。

此类问题可通过以下方式进行解决：

在一些大规模集群的场景中，用户可以通过 service.kubernetes.io/qcloud-loadbalancer-backends-label 指定一部分节点进行绑定。 service.kubernetes.io/qcloud-loadbalancer-backends-label 的内容是一个标签选择器，用户可以通过在集群节点上标记 Label，然后在 Service 中通过该注解描述的标签选择器，选择匹配的节点进行绑定。这个同步会持续进行，当节点发生变化导致其被选择或是不再被选择时，Service Controller 会对应添加或删除负载均衡上的对应后端。

注意事项

- 当 service.kubernetes.io/qcloud-loadbalancer-backends-label 的选择器没有选取到任何节点的时候，服务的后端将会被排空，会使得服务中断。使用此功能时，需要对集群节点的 Label 有一定的管理。
- 新增符合要求的节点或变更存量节点也会触发 controller 更新。

使用场景

大规模集群下的测试应用

在一个大规模集群下，部署一个仅包含一两个 Pod 的测试应用。通过 Service 进行服务暴露时，负载均衡将对所有的后端 NodePort 进行健康检查，此健康检查的请求量对测试应用有很大影响。此时可以在集群中通过 Label 指定一小部分节点作为后端，缓解健康检查带来的压力。

示例

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.kubernetes.io/qcloud-loadbalancer-backends-label: "group=access-layer"
  name: nginx-service
spec:
  ports:
    - name: 80-80-no
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

该示例包含以下配置：

- 描述了一个公网类型负载均衡的服务暴露。
- service.kubernetes.io/qcloud-loadbalancer-backends-label 注解声明了后端选择器，仅支持集群节点上有 group=access-layer Label 的节点才会作为这个负载均衡的后端。

Service Local 模式

Kubernetes 提供了 Service 特性 ExternalTrafficPolicy 。当 ExternalTrafficPolicy 设置为 Local 时，可以避免流量通过 NAT 在节点间的转发，减少了 NAT 操作也使得源 IP 得到了保留。NodePort 仅会将流量转发到当前节点的 Pod。Local 模式特点如下：

优点：

- 避免了 NAT 与节点间转发带来的性能损失。
- 为服务端保留了请求来源 IP。

缺点：

没有工作负载的节点，NodePort 将无法提供服务。

注意事项

- Service Local 模式本身就具有节点选择能力，所以：
  - 请不要在 Service Local 模式下，使用指定接入层后端的能力。
  - 请不要在 Service Local 模式下，在 Node 上使用 node.kubernetes.io/exclude-from-external-load-balancers Label驱逐后端节点。
- 负载均衡的同步是需要时间的。当 Local 类型的服务工作负载数量很少时，工作负载的飘移或滚动更新会很快。此时后端如未来得及同步，后端的服务可能会出现不可用的情况。
- 仅适用于处理低流量、低负载的业务，不建议在生产环境中使用。

示例：Service 开启 Local 转发（externalTrafficPolicy: Local）

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  externalTrafficPolicy: Local
  ports:
    - name: 80-80-no
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Local 默认后端选择

默认情况下，当 Service 开启 Local 模式之后，仍会按默认方式挂载几乎所有节点的 NodePort 作为后端。负载均衡

会根据健康检查的结果，避免流量进入没有工作负载的后端节点。为了避免这些没有工作负载的后端被绑定，用户可以通过 service.kubernetes.io/local-svc-only-bind-node-with-pod: "true" 注解，在 Local 模式下指定绑定有工作负载节点作为后端。

示例：Service 开启 Local 转发并开启 Local 绑定

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.kubernetes.io/local-svc-only-bind-node-with-pod: "true"
  name: nginx-service
spec:
  externalTrafficPolicy: Local
  ports:
    - name: 80-80-no
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

由于 Local 模式下，进入节点的请求流量不会在节点间转发。所以当节点上的工作负载数量不一致的时候，同样的后端权重可能会使得每一个节点上的负载不平均。此时用户可以通过 service.finance.cloud.tencent.com/local-svc-weighted-balance: "true" 进行加权平衡。使用此注解时，NodePort 后端的权重将由节点上工作负载的数量决定，从而避免不同节点上工作负载数量不同带来的负载不均的问题。其中，Local 加权平衡必须和 Local 绑定同时使用。示例如下：

示例：Service 开启 Local 转发，并开启 Local 绑定与 Local 加权平衡

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.kubernetes.io/local-svc-only-bind-node-with-pod: "true"
    service.finance.cloud.tencent.comm/local-svc-weighted-balance: "true"
  name: nginx-service
spec:
  externalTrafficPolicy: Local
  ports:
    - name: 80-80-no
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
```

```
    app: nginx
  type: LoadBalancer
```

Service 优雅停机

简介

基于接入层直连 Pod 的场景，当后端进行滚动更新或后端 Pod 被删除时，如果直接将 Pod 从 LB 的后端摘除，则无法处理 Pod 已接收但还未处理的请求。
特别是长链接的场景，例如会议业务，如果直接更新或删除工作负载的 Pod，此时会议会直接中断。

应用场景

- 更新工作负载时，Pod 的优雅退出，使客户端不会感受到更新时产生的抖动和错误。
- 当 Pod 需要被删除时，Pod 能够处理完已接收到的请求，此时入流量关闭，但出流量仍能走通。直到处理完所有已有请求和 Pod 真正删除时，出入流量才进行关闭。

> 注意：
>
> i. 仅针对 直连场景 生效，请检查您的集群是否支持直连模式。
> ii. 如果是多 Service 复用相同的 CLB，且同时更新多个 Service 对应的后端服务，可能导致优雅停机功能受影响，详情见 多 Service 复用 CLB 中的说明事项。

操作步骤

步骤1：使用 Annotation 标明使用优雅停机

以下为使用 Annotation 标明使用优雅停机示例，完整 Service Annotation 说明可参见 Service Annotation 说明。

```
kind: Service
apiVersion: v1
metadata:
  annotations:
    service.finance.cloud.tencent.com/direct-access: "true" ## 开启直连 Pod 模式
    service.finance.cloud.tencent.com/enable-grace-shutdown: "true"  # 表示使用优雅停机
  name: my-service
spec:
  selector:
    app: MyApp
```

步骤2：使用 preStop 和 terminationGracePeriodSeconds

步骤2为在需要优雅停机的工作负载里配合使用 preStop 和 terminationGracePeriodSeconds。

容器终止流程

以下为容器在 Kubernetes 环境中的终止流程：

1. Pod 被删除，此时 Pod 里有 DeletionTimestamp，且状态置为 Terminating。此时调整 CLB 到该 Pod 的权重为 0。
2. kube-proxy 更新转发规则，将 Pod 从 service 的 endpoint 列表中摘除掉，新的流量不再转发到该 Pod。
3. 如果 Pod 配置了 preStop Hook ，将会执行。
4. kubelet 将对 Pod 中各个 container 发送 SIGTERM 信号，以通知容器进程开始优雅停止。
5. 等待容器进程完全停止，如果在 terminationGracePeriodSeconds 内 (默认30s) 还未完全停止，将发送 SIGKILL 信号强制停止进程。
6. 所有容器进程终止，清理 Pod 资源。

具体操作步骤

1. 使用 preStop
   要实现优雅终止，务必在业务代码里处理 SIGTERM 信号。主要逻辑是不接受新的流量进入，继续处理存量流量，所有连接全部断开才退出。
   若您的业务代码中未处理 SIGTERM 信号，或者您无法控制使用的第三方库或系统来增加优雅终止的逻辑，也可以尝试为 Pod 配置 preStop，在其实现优雅终止的逻辑，示例如下：

```
apiVersion: v1
kind: Pod
metadata:
  name: lifecycle-demo
spec:
  containers:
  - name: lifecycle-demo-container
    image: nginx
    lifecycle:
      preStop:
        exec:
          command:
          - /clean.sh
...
```

   更多关于 preStop 的配置请参见 Kubernetes API 文档。
   在某些极端情况下，Pod 被删除的一小段时间内，仍然可能有新连接被转发过来，因为 kubelet 与 kube-proxy 同时 watch 到 Pod 被删除，kubelet 有可能在 kube-proxy 同步完规则前就已经停止容器，这时可能导致一些新的连接被转发到正在删除的 Pod，而通常情况下，当应用受到 SIGTERM 后都不再接受新连接，只保持存量连接继续处理，因此可能导致 Pod 删除的瞬间部分请求失败。
   针对上述情况，可以利用 preStop 先 sleep 短暂时间，等待 kube-proxy 完成规则同步再开始停止容器内进

程。示例如下：

```
apiVersion: v1
kind: Pod
metadata:
  name: lifecycle-demo
spec:
  containers:
  - name: lifecycle-demo-container
    image: nginx
    lifecycle:
      preStop:
        exec:
          command:
          - sleep
          - 5s
```

2. 使用 terminationGracePeriodSeconds 调整优雅时长
   如果需要优雅终止时间较长 (preStop + 业务进程停止可能超过30s)，可根据实际情况自定义 terminationGracePeriodSeconds，避免过早的被 SIGKILL 停止，示例如下：

```
apiVersion: v1
kind: Pod
metadata:
  name: grace-demo
spec:
  terminationGracePeriodSeconds: 60 # 优雅停机默认30s，您可以设置更长的时间
  containers:
  - name: lifecycle-demo-container
    image: nginx
    lifecycle:
      preStop:
        exec:
          command:
          - sleep
          - 5s
...
```

相关能力

优雅停机只是在 Pod 删除时，才把 CLB 后端的权重置为 0。若 Pod 在运行的过程中，出现了不健康的情况，此时将该后端的权重置为 0，可以减少服务不可用的风险。

您可以使用 Annotation： service.finance.cloud.tencent.com/enable-grace-shutdown-tkex: "true" 实现这样优雅退出的能力。
该 Annotation 会根据 Endpoint 对象中 endpoints 是否 not-ready，将 not-ready 的 CLB 后端权重置为 0。

Pod 优雅删除

操作背景

在 CLB 直连场景下，当用户发起工作负载滚动更新时，会同时触发两个并行的工作流：

1. Service 控制器调整 CLB 后端 Pod 的权重为 0，并在后续摘除该 Pod。
2. 运行时（kubelet）终止并删除 Pod。
   在某些极端场景，比如有多个 Service 复用同一个 CLB，或者一个 Service 下有非常多的 Pod，可能出现某些 Pod 已经被运行时删除了，但 Service 控制器还没有完成 CLB 上该 Pod 的权重调整，进而导致流量还会发往不存在的 Pod IP，导致业务异常。

应用场景

TKE Service 的优雅删除特性，会将上述两个并行的工作流，协调为串行，也就是先让 Service 控制器完成 CLB 后端 Pod 权重调整为 0，再让运行时（kubelet）删除该 Pod。

> 注意：
>
> 1. 仅适用于 直连场景。
> 2. 对节点 kubelet 版本有要求，确保集群节点版本支持 kubelet 支持 pod 删除保护功能。
> 3. 目前只支持 Service，暂不支持 ingress。该功能对 TKE Service 组件版本有要求，检查集群是否支持该功能。

操作步骤

步骤1：确保集群节点版本和 Service 组件版本支持该功能

kubelet 版本：确保集群节点版本支持 kubelet 的 Pod 删除保护功能。

步骤2：使用注解对指定的 Service 开启 Pod 优雅删除

YAML 配置示例如下：

```
apiVersion: v1
kind: Service
metadata:
  name: anyserver
  annotations:
    service.finance.cloud.tencent.com/direct-access: "true"
    service.finance.cloud.tencent.com/enable-grace-deletion: "true" # 表示开启 Pod 优雅删除
```

使用 LoadBalancer 直连 Pod 模式 Service

操作场景

原生 LoadBalancer 模式 Service 可自动创建负载均衡 CLB，并通过集群的 Nodeport 转发至集群内，再通过 iptable 或 ipvs 进行二次转发。该模式下的 Service 能满足大部分使用场景 ，但在以下场景中更推荐使用直连 Pod 模式 Service：

- 有获取来源 IP 需求时（非直连模式必须另外开启 Local 转发）。
- 要求具备更高转发性能时（非直连模式下 CLB 和 Service 本身存在两层 CLB，性能有一定损失）。
- 需使用完整的健康检查和会话保持到 Pod 层级时（非直连模式下 CLB 和 Service 本身存在两层 CLB，健康检查及会话保持功能较难配置）。

> 说明：
>
> - 若您的集群是 Serverless 集群，则默认为直连 Pod 模式，您无需任何操作。
> - 当前 GlobalRouter 和 VPC-CNI 容器网络模式均支持直连 Pod 模式，您可以在集群列表中单击集群 ID 进入集群详情页面，在集群的“基本信息”页面中查看当前集群使用的网络插件。

容器网络模式为 VPC-CNI

使用限制

- 集群 Kubernetes 版本需要高于 1.12。
- 集群网络模式必须开启 VPC-CNI 弹性网卡模式。
- 直连模式 Service 使用的工作负载需使用 VPC-CNI 弹性网卡模式。
- 默认 CLB 的后端数量会有限制，具体限制数量与 CLB 所在地域有关，默认限制通常是100或200个。
- 满足 CLB 本身绑定弹性网卡的功能限制，详情请参见 绑定弹性网卡。
- 开启直连 Pod 模式的工作负载更新时，将会根据 CLB 的健康检查状态进行滚动更新，会对更新速度造成一定影响。

> 注意：
>
> 从 NodePort 接入方案迁移至直连时，需要注意以下事项：
>
> - 确认弹性网卡工作负载的安全组配置，确认出入流量放通。
> - 确认来源 IP 的变化对业务没有影响。（直连避免了 FullNAT 转发，来源 IP 不再是节点 IP，而是请求来源 IP）。
> - 确认工作负载当前没有处于滚动更新中的状态。
> - 如果出现数据面访问的不预期，请及时参考操作步骤进行回滚。

操作步骤

控制台操作指引

1. 登录容器服务控制台。
2. 参考控制台创建 Service 步骤，进入新建 Service 页面，根据实际需求设置 Service 参数。

其中，部分关键参数信息需进行如下设置。

- 服务访问方式：选择为公网 LB 访问或内网 LB 访问。
- 网络模式：勾选采用负载均衡直连 Pod 模式。（取消勾选则为默认的 NodePort 接入方式）
- Workload 绑定：选择引用 Workload。

3. 单击创建 Service，完成创建。

YAML 操作指引

直连 Pod 模式 Service 的 YAML 配置与普通 Service YAML 配置相同，示例中的 annotation 即代表是否开启直连 Pod 模式。

```
kind: Service
apiVersion: v1
metadata:
  annotations:
    service.finance.cloud.tencent.com/direct-access: "true" ##开启直连 Pod 模式（删除注解或"false"，为关闭直连）
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

annotation 扩展

负载均衡 CLB 的相关配置可参见 TkeServiceConfig 介绍。其中相关 annotation 配置如下：

```
service.finance.cloud.tencent.com/tke-service-config: [tke-service-configName]
```

注意事项

如何保证滚动更新时的可用性保证

Kubernetes 官方提供的一个特性 ReadinessGate，主要是用来控制 Pod 的状态，集群版本需高于1.12。默认情况下，Pod 有以下 Condition：PodScheduled、Initialized、ContainersReady，当这几个状态都 Ready 的时候，Pod Ready 的 Condition 就通过了。但是在云原生场景下，Pod 的状态可能需要参考其他状态。ReadinessGate 提供了这样一个机制，允许为 Pod 的状态判断添加一个栅栏，由第三方来进行判断与控制。这样 Pod 的状态就和第三方关联起来了。

直连模式滚动更新的变化

当用户开始为应用做滚动更新的时候，Kubernetes 会根据更新策略进行滚动更新。但其判断一批 Pod 启动的标识仅包括 Pod 自身的状态，并不会考虑该 Pod 在负载均衡上是否配置健康检查且通过。如在接入层组件高负载时，不能及时对此类 Pod 进行及时调度，则滚动更新成功的 Pod 可能并没有正在对外提供服务，从而导致服务的中断。

为了关联滚动更新和负载均衡的后端状态，TKE 接入层组件引入了 Kubernetes 1.12中引入的新特性 ReadinessGate 。TKE 接入层组件仅在确认后端绑定成功并且健康检查通过时，通过配置 ReadinessGate 的状态来使 Pod 达到 Ready 的状态，从而推动整个工作负载的滚动更新。

> 注意：
>
> ReadinessGate 仅负责流量就绪检查，和 Endpoints 的职责是不一样的，所以 Pod 启动就绪之后的健康状态不会在 ReadinessGate Status 中体现。

在集群中使用 ReadinessGate

Kubernetes 集群提供了服务注册的机制，只需要将您的服务以 MutatingWebhookConfigurations 资源的形式注册至集群即可。集群会在 Pod 创建的时候按照配置的回调路径进行通知，此时可对 Pod 进行创建前的操作，即给 Pod 加上 ReadinessGate 。需注意此回调过程必须是 HTTPS，即需要在 MutatingWebhookConfigurations 中配置签发请求的 CA，并在服务端配置该 CA 签发的证书。

ReadinessGate 机制的灾难恢复

用户集群中的服务注册或证书有可能被用户删除，虽然这些系统组件资源不应该被用户修改或破坏。在用户对集群的探索或是误操作下，这类问题会不可避免的出现。因此接入层组件在启动时会检查以上资源的完整性，在完整性受到破坏时会重建以上资源，加强系统的鲁棒性。详情可参见 Kubernetes Pods ReadinessGate 特性。

容器网络模式为 GlobalRouter

使用限制

- 单个工作负载仅能运行在一种网络模式下，您可选择弹性网卡直连或 GlobalRoute 直连。
- 默认 CLB 的后端数量会有限制，具体限制数量与 CLB 所在地域有关，默认限制通常是100或200个。
- 使用 CLB 直连 Pod，需注意网络链路受云服务器的安全组限制，确认安全组配置是否放开对应的协议和端口，需要开启 CVM 上工作负载对应的端口。
- 不支持机型 BMG5n.24XLARGE384，如果有使用该机型，确保启用直连的 Pod 不被调度到该机型的节点。

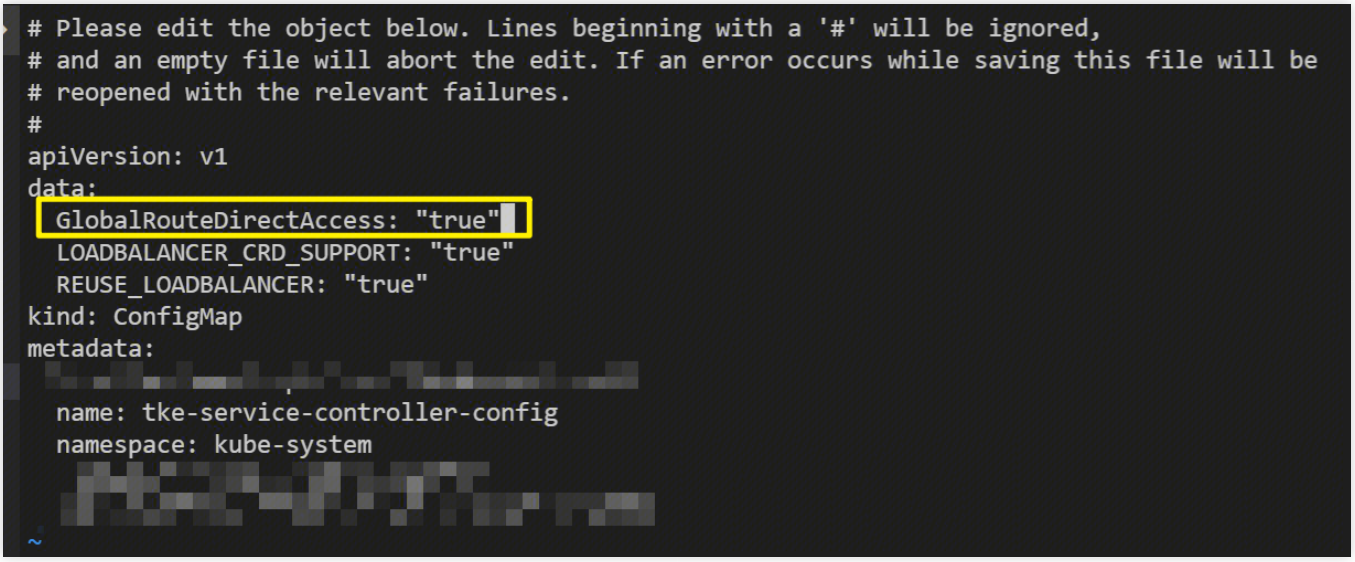
步骤1: 开启直连配置

对于 GlobalRouter 集群，需要先开启“集群”维度的直连能力，再开启 “service” 维度的直连开关。
在 kube-system/tke-service-controller-config ConfigMap 中新增 GlobalRouteDirectAccess: "true" 以开启 GlobalRoute 直连能力。
控制台操作指引

1. 登录容器服务控制台，在左侧导航栏中选择集群。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 在配置管理 > ConfigMap 页面，命名空间选择 kube-system ，搜索 tke-service-controller-config ，单击更新配置。
4. 在更新配置页面，选择手动增加，变量名填写为 GlobalRouteDirectAccess ，变量值填写为 true （注意没有引号）。
5. 单击更新 ConfigMap。

YAML 操作指引

1. 登录容器服务控制台，在左侧导航栏中选择集群。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 在配置管理 > ConfigMap 页面，单击 tke-service-controller-config 右侧的**编辑 yaml。**或通过命令行执行 kubectl edit -n kube-system cm tke-service-controller-config 进入 yaml 编辑模式。
4. 在 data 下新增行 GlobalRouteDirectAccess: "true"（注意缩进2个控制），保存即可。




```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  GlobalRouteDirectAccess: "true"
  LOADBALANCER_CRD_SUPPORT: "true"
  REUSE_LOADBALANCER: "true"
kind: ConfigMap
metadata:
  name: tke-service-controller-config
  namespace: kube-system
```

步骤2: Service 或 Ingress 中开启直连模式

2.1 Service 开启直连模式

您可以通过以下方式开启 Service 的直连模式：
方式1：

1. 登录容器服务控制台，在左侧导航栏中选择集群。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 在服务与路由 > Service 页面，单击待配置的 Service 右侧的编辑 yaml。
   方式2：
   通过命令行工具使用 kubectl edit service -n <命名空间> <service名> 进入编辑模式。
   示例：
   在 metadata.annotations 中新增以下注解： service.finance.cloud.tencent.com/direct-access: "true" ，表示开启 Service 的直连模式。如果未添加该注解或注解值为 "false"，则表示关闭直连模式。
   直连 Pod 模式 Service 的其他 YAML 配置与普通 Service YAML 配置相同。

```
kind: Service
apiVersion: v1
metadata:
  annotations:
    service.finance.cloud.tencent.com/direct-access: "true" ##开启直连 Pod 模式
```

```
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

Annotation 扩展：

有关负载均衡 CLB 的配置，请参见 TkeServiceConfig 介绍。其中相关 Annotation 配置如下：

```
service.finance.cloud.tencent.com/tke-service-config: [tke-service-configName]
```

2.2 Ingress 开启直连模式

对于 Ingress 需要使用 ingress.finance.cloud.tencent.com/direct-access: "true" 注解来开启直连模式。

除了注解名称不同外，其余步骤与 开启 Service 的直连模式 相同。

多 Service 复用 CLB

操作场景

您可通过多个 Service 复用相同负载均衡器 CLB 的能力，来支持在同一个 VIP 同时暴露 TCP 及 UDP 的相同端口。

> 注意：
>
> 其他场景下均不建议使用多个 Service 复用相同的 CLB。

说明事项

- 如果您的集群是 TKE Serverless 集群，集群默认已开启了 CLB 复用能力，但需要注意以下内容：
  i. 用于复用的 CLB 必须为用户手动购买，而非 Serverless 集群自动购买。Serverless 集群自动购买的 CLB 在复用时会报错，是为了保护复用 CLB 的 Service 的 CLB 不被 Serverless 集群回收。
  ii. CLB 购买成功后，需要在 Service 里添加两个 Annotation：
    - service.kubernetes.io/qcloud-share-existed-lb:"true"
    - service.kubernetes.io/tke-existed-lbid:lb-xxx
- Service 和 CLB 之间配置的管理和同步是由以 CLB ID 为名字的 LoadBalancerResource 类型的资源对象，请勿对该 CRD 进行任何操作，否则容易导致 Service 失效。
- 在业务发版时（滚动更新），请避免同时对这些 Service 关联的后端服务进行发版，因为同步这些复用相同 CLB 的 Service 的操作是串行的（保证同步结果的准确性），如果多个 Service 后端同时在更新，部分后端的流量摘除可能就不及时，容易造成流量异常，同时 优雅停机 的能力也不能真正生效。

使用限制

- 多个 Service 复用相同 CLB 时，Service 的端口不能重复。
- 在 Service 复用场景下，单个负载均衡管理的监听器数量由 CLB 的 TOTAL_LISTENER_QUOTA 限制。
- 在 Service 复用场景下，只能使用用户自行创建的负载均衡。因为容器服务 TKE 集群创建的负载均衡在被复用的情况下，负载均衡资源可能因为无法释放而导致泄漏。

  > 注意：
  >
  > 负载均衡资源被复用后，该 CLB 的生命周期将不由 TKE 侧控制，需要自行管理，请谨慎操作。

操作步骤

1. 参考创建负载均衡实例，创建集群所在 VPC 下的公网或内网类型的负载均衡。

2. 参考 创建 Deployment 或 创建 Service，创建 LoadBalancer 类型的 Service，选择使用已有负载均衡，并选择 步骤1 中创建的负载均衡实例。
3. 重复步骤2，即可完成通过多个 Service 复用相同负载均衡器 CLB。

Service 扩展协议

Service 默认支持的协议

Service 是 Kubernetes 暴露应用程序到集群外的一种机制与抽象，您可以通过 Service 访问集群内的应用程序。

> 注意：
>
> - 在 直连场景 下接入时，使用扩展协议没有任何限制，支持 TCP 和 UDP 协议混用。
> - 在非直连场景下，ClusterIP 和 NodePort 模式支持混用。但是，对于 LoadBalancer 类型的 Service，社区目前仅支持同类型协议的使用。
> - 当 LoadBalancer 声明为 TCP 时，端口可以使用扩展协议的能力，将负载均衡的协议变更为 HTTP 或 HTTPS。
> - 当 LoadBalancer 声明为 UDP 时，端口可以使用扩展协议的能力，将负载均衡的协议变更为 QUIC。

TKE 扩展 Service 转发协议

在原生的 Service 支持的协议规则上，存在部分场景需要在 Service 上同时支持 TCP 和 UDP 混合，且需 Service 能够支持 TCP SSL、HTTP、HTTPS 协议。TKE 针对 LoadBalancer 模式扩展了更多协议的支持。

前置说明

- 扩展协议仅对 LoadBalancer 模式的 Service 生效。
- 扩展协议通过注解 Annotation 的形式描述协议与端口的关系。
- 扩展协议与注解 Annotation 关系如下：
  - 当扩展协议注解中没有覆盖 Service Spec 中描述的端口时，Service Spec 按照用户描述配置。
  - 当扩展协议注解中描述的端口在 Service Spec 中不存在时，忽略该配置。
  - 当扩展协议注解中描述的端口在 Service Spec 中存在时，覆盖用户在 Service Spec 中声明的协议配置。

注解名称

service.finance.cloud.tencent.com/specify-protocol

扩展协议注解示例

HTTP 示例

```
{"80":{"protocol":["HTTP"],"hosts":{"a.finance.cloud.tencent.com":{},"b.finance.cloud.tencent.com":{}}}}
```

HTTPS 示例

{"80":{"protocol":["HTTPS"],"hosts":{"a.finance.cloud.tencent.com":{"tls":"cert-secret-a"},"b.finance.cloud.tencent.com":{"tls":"cert-secret-b"}}}}

TCP/UDP混合示例

```
{"80":{"protocol":["TCP","UDP"]}} # 仅直连模式支持
```

> 注意：
>
> HTTPS 中的字段 cert-secret ，表示使用该协议需要指定一个证书，证书是 Opaque 类型的 Secret，Secret 的 Key 为 qcloud_cert_id，Value 是证书 ID。详情见 Ingress 证书配置。

扩展协议使用说明

扩展协议YAML使用说明

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.finance.cloud.tencent.com/specify-protocol: '{"80":{"protocol":["HTTPS"],"tls":"cert-secret"}}'
# 若要使用别的协议，修改该键值对的值为上述内容
  name: test
    ....
```

扩展协议控制台使用说明

- 当 Service 为仅在集群内访问（ClusterIP）或主机端口访问（NodePort）模式时，支持任意协议混用。
- 直连模式，支持任意协议混用。
  扩展协议YAML使用说明

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.finance.cloud.tencent.com/specify-protocol: '{"80":{"protocol":["HTTPS"],"tls":"cert-secret"}}'
# 若要使用别的协议，修改该键值对的值为上述内容
  name: test
    ....
```

扩展协议控制台使用说明

- 当 Service 为仅在集群内访问（ClusterIP）或主机端口访问（NodePort）模式时，支持任意协议混用。
- 直连模式，支持任意协议混用。

案例说明

原生 Service 不支持协议混用，TKE 经过特殊改造后，在 直连场景 中支持混合协议的使用。
需注意的是，YAML 中仍使用相同的协议，但可以通过 Annotation 明确每个端口的协议类型。如下示例展示了 80 端口使用 TCP 协议，8080 端口使用 UDP 协议。

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.finance.cloud.tencent.com/direct-access: "true"  #TKE Serverless 集群默认是直连模式，TKE 集群请务必先参照文档开启直连模式。
    service.finance.cloud.tencent.com/specify-protocol: '{"80":{"protocol":["TCP"]},"8080":{"protocol":["UDP"]}}'  # 指定 80 端口 TCP 协议，8080 端口 UDP 协议。
  name: nginx
spec:
  externalTrafficPolicy: Cluster
  ports:
  - name: tcp-80-80
    nodePort: 32150
    port: 80
    protocol: TCP
    targetPort: 80
  - name: udp-8080-8080
    nodePort: 31082
    port: 8080
    protocol: TCP # 注意，因为 Kubernetes Service Controller 限制，只能使用同类型协议。
    targetPort: 8080
  selector:
    k8s-app: nginx
    qcloud-app: nginx
  sessionAffinity: None
  type: LoadBalancer
```

Service Annotation 说明

您可以通过以下 Annotation 注解配置 Service，以实现更丰富的负载均衡的能力。

注解使用方式

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
  name: test
.....
```

Annotation 集合

service.kubernetes.io/loadbalance-id

说明：
只读注解，提供当前 Service 引用的负载均衡 LoadBalanceId。您可以在CLB 控制台查看与集群在同一 VPC 下的 CLB 实例 ID。

service.kubernetes.io/qcloud-loadbalancer-internal-subnetid

说明：
通过该 Annotation 指定创建内网类型 CLB，取值为子网 ID。
使用示例：

```
service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx
```

service.kubernetes.io/tke-existed-lbid

说明：
使用已存在的 CLB，需注意不同使用方式对标签的影响。
使用示例：
使用方式详情见 Service 使用已有 CLB。

service.kubernetes.io/local-svc-only-bind-node-with-pod

说明：
Service Local 模式下仅绑定有 Pod 存在的节点。
使用示例：

使用方式详情见 Service Local 模式。

service.kubernetes.io/qcloud-loadbalancer-backends-label

说明：
指定标签设置负载均衡后端绑定的节点。
使用示例：
使用方式详情见 指定接入层后端。

service.kubernetes.io/loadbalance-nat-ipv6

说明：
只读注解，创建 NAT64 IPv6 负载均衡时，负载均衡的 IPv6 地址将会展示到注解中。
使用示例：
service.kubernetes.io/loadbalance-nat-ipv6: "2402:4e00:1402:7200:0:9223:5842:2a44"

service.kubernetes.io/service.extensiveParameters

说明：
该 Annotation 使用的是 CLB 创建时的参数，当前仅在创建时支持配置，创建后不支持修改，创建后修改本注解无效。
使用示例：

- 创建 NAT64 IPv6 实例：
  service.kubernetes.io/service.extensiveParameters: '{"AddressIPVersion":"IPV6"}'
- 购买电信负载均衡：
  service.kubernetes.io/service.extensiveParameters: '{"VipIsp":"CTCC"}'
- 创建时自定义 CLB 名字：
  service.kubernetes.io/service.extensiveParameters: '{"LoadBalancerName":"my_cutom_lb_name"}'

service.kubernetes.io/qcloud-loadbalancer-internet-charge-type

说明：
负载均衡的付费类型，当前仅在创建时支持配置，创建后不支持修改付费类型，创建后修改本注解无效。
指定创建负载均衡时，负载均衡的付费类型。请配合 service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out 注解一起使用。
可选值：

- BANDWIDTH_POSTPAID_BY_HOUR 按带宽按小时后计费
- TRAFFIC_POSTPAID_BY_HOUR 按流量按小时后计费
  使用示例：
  service.kubernetes.io/qcloud-loadbalancer-internet-charge-type："TRAFFIC_POSTPAID_BY_HOUR"

service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out

说明：
CLB 带宽设置，当前仅在创建时支持配置，创建后不支持修改带宽，创建后修改本注解无效。
指定创建负载均衡时，负载均衡的最大出带宽，仅对公网属性的 LB 生效。需配合 service.kubernetes.io/qcloud-loadbalancer-internet-charge-type 注解一起使用。
可选值：
范围支持1到2048，单位 Mbps。
使用示例：
service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out: "2048"

配置
ConfigMap管理

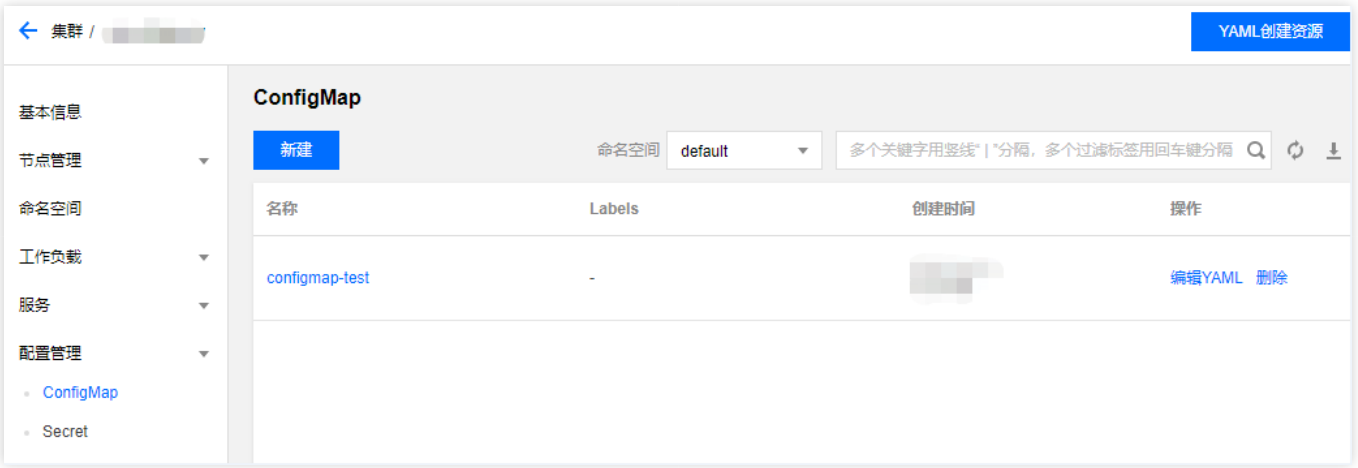
简介

通过 ConfigMap 您可以将配置和运行的镜像进行解耦，使得应用程序有更强的移植性。ConfigMap 是有 key-value 类型的键值对，您可以通过控制台的 Kubectl 工具创建对应的 ConfigMap 对象，也可以通过挂载数据卷、环境变量或在容器的运行命令中使用 ConfigMap。

ConfigMap 控制台操作指引

创建 ConfigMap

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建 ConfigMap 的集群【ID/名称】，进入待创建 ConfigMap 的集群管理页面。
4. 选择【配置管理】 > 【ConfigMap】，进入“ConfigMap”信息页面。

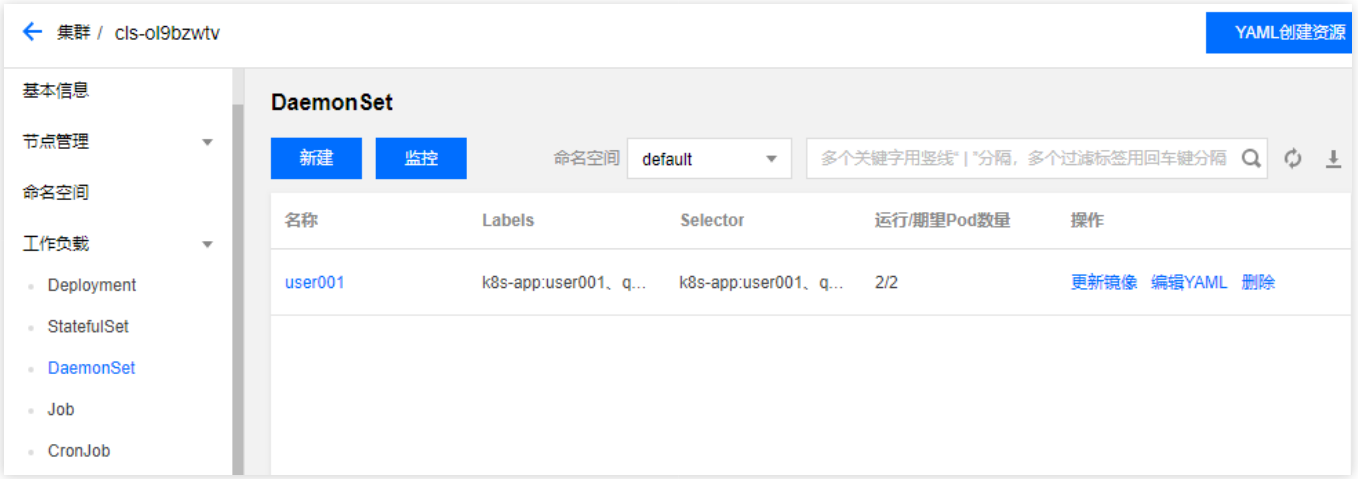


5. 单击【新建】，进入“新建ConfigMap”页面。
6. 根据实际需求，设置 ConfigMap 参数。关键参数信息如下：
   - 名称：自定义。
   - 命名空间：根据实际需求进行选择命名空间类型，定义变量名和变量值。
     也可以选择导入文件的方式进行创建。
7. 单击【创建ConfigMap】，完成创建。

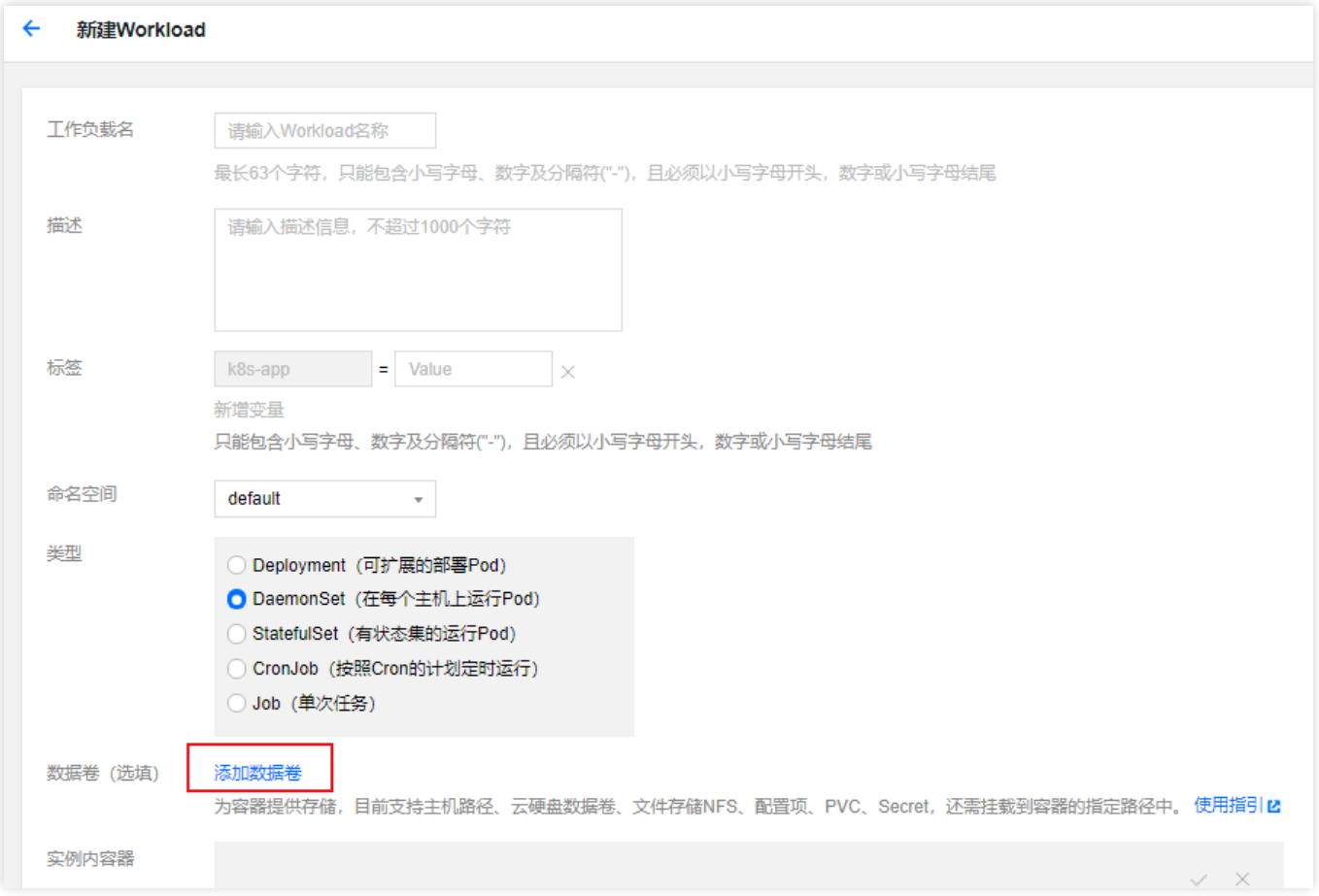
使用 ConfigMap

方式一：数据卷使用 ConfigMap 类型

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要部署 Workload 的集群【ID/名称】，进入待部署 Workload 的集群管理页面。
4. 在【工作负载】下，任意选择 Workload 类型，进入对应的信息页面。例如，选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。



5. 单击【新建】，进入“新建Workload”页面。
6. 根据页面信息，设置工作负载名、命名空间等信息。并在【数据卷】中，单击【添加数据卷】，添加数据卷。



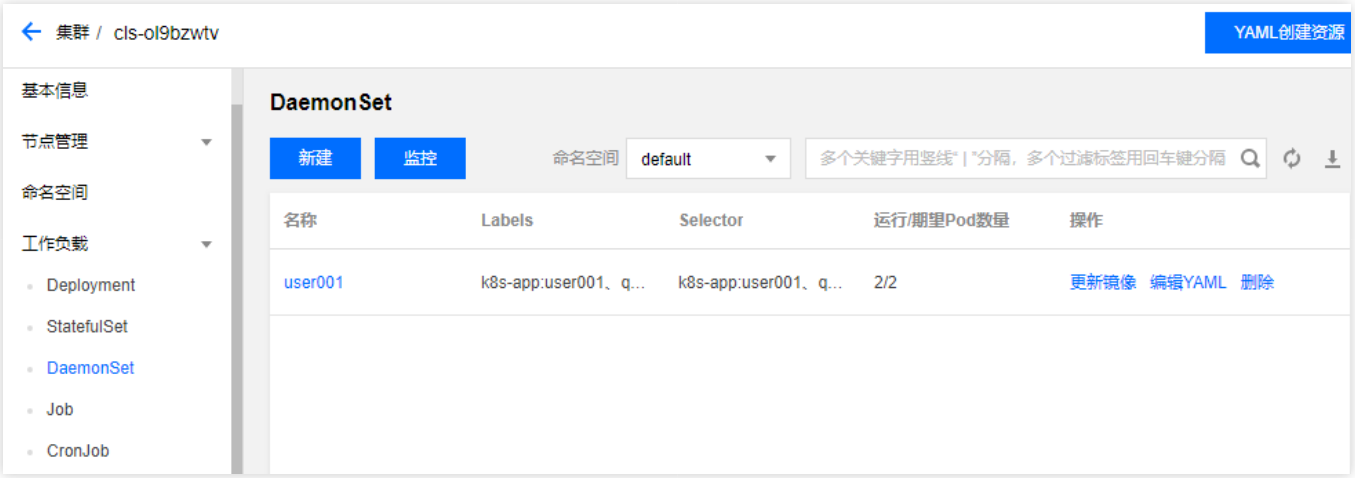
7. 选择【使用ConfigMap】方式，填写名称，单击【选择配置项】。



8. 在弹出的“设置ConfigMap”窗口中，配置挂载点，单击【确认】。
9. 单击【创建Workload】，完成创建。

方式二： 环境变量中使用 ConfigMap 类型

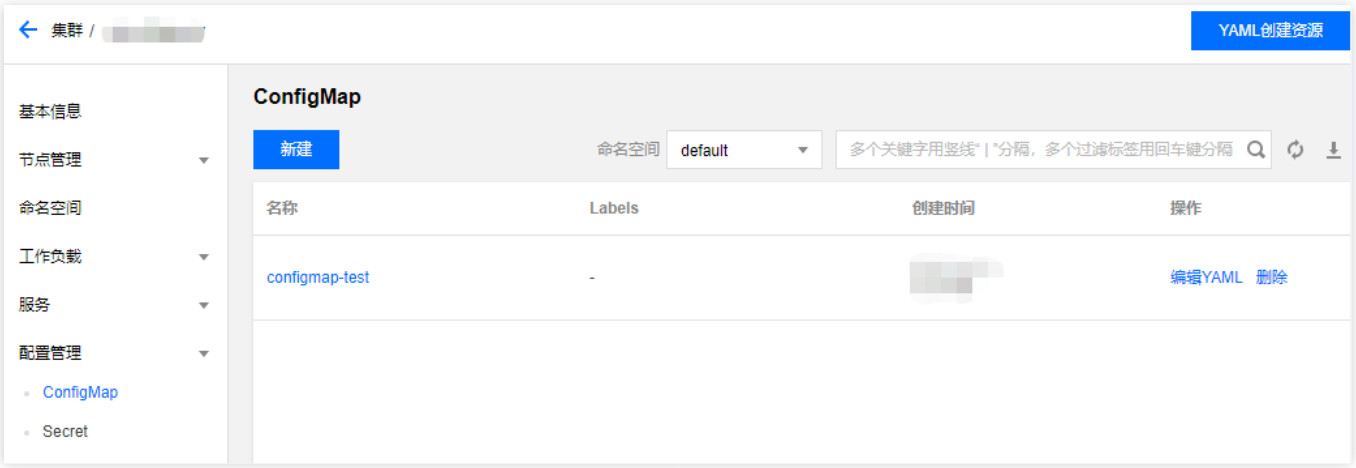
1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要部署 Workload 的集群【ID/名称】，进入待部署 Workload 的集群管理页面。
4. 在【工作负载】下，任意选择 Workload 类型，进入对应的信息页面。例如，选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。



5. 单击【新建】，进入“新建Workload”页面。
6. 根据页面信息，设置工作负载名、命名空间等信息。
7. 选择 “ConfigMap” 环境变量方式，并根据实际需求选择资源。
8. 单击【创建Workload】，完成创建。

更新 ConfigMap

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 YAML 的集群【ID/名称】，进入待更新 YAML 的集群管理页面。
4. 选择【配置管理】 > 【ConfigMap】，进入“ConfigMap”信息页面。



5. 在需要更新 YAML 的 ConfigMap 行中，单击【编辑YAML】，进入“更新 ConfigMap”页面。
6. 在“更新ConfigMap”页面，编辑 YAML，单击【完成】，即可更新 YAML。 > 说明：
   如需修改 key-values，编辑 YAML 中 data 的参数值，单击【完成】，即可完成更新。

Kubectl 操作 ConfigMap 指引

YAML 示例

```
apiVersion: v1
data:
  key1: value1
  key2: value2
  key3: value3
kind: ConfigMap
metadata:
  name: test-config
  namespace: default
```

- data：ConfigMap 的数据，以 key-value 形式呈现。
- kind：标识 ConfigMap 资源类型。
- metadata：ConfigMap 的名称、Label等基本信息。
- metadata.annotations：ConfigMap 的额外说明，可通过该参数设置TKE 的额外增强能力。

创建 ConfigMap

方式一：通过 YAML 示例文件方式创建

1. 参考 YAML 示例，准备 ConfigMap YAML 文件。

2. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。

3. 执行以下命令，创建 ConfigMap YAML 文件。

   ```
   kubectl create -f ConfigMap YAML 文件名称
   ```

   例如，创建一个文件名为 web.yaml 的 ConfigMap YAML 文件，则执行以下命令：

   ```
   kubectl create -f web.yaml
   ```

4. 执行以下命令，验证创建是否成功。

   ```
   kubectl get configmap
   ```

   返回类似以下信息，即表示创建成功。

   ```
   NAME          DATA      AGE
   test          2         39d
   test-config   3         18d
   ```

方式二：通过执行命令方式创建

执行以下命令，在目录中创建 ConfigMap。

```
kubectl create configmap <map-name> <data-source>
```

- <map-name>：表示 ConfigMap 的名字。
- <data-source>：表示目录、文件或者字面值。
  更多参数详情可参见 Kubernetes configMap 官方文档。

使用 ConfigMap

方式一：数据卷使用 ConfigMap 类型

YAML 示例如下：

```
apiVersion: v1
 kind: Pod
 metadata:
   name: nginx
 spec:
```

```
    containers:
      - name: nginx
        image: nginx:latest
        volumeMounts:
          name: config-volume
          mountPath: /etc/config
    volumes:
        name: config-volume
        configMap:
          name: test-config ## 设置 ConfigMap 来源
          ## items:  ## 设置指定 ConfigMap 的 Key 挂载
          ##   key: key1  ## 选择指定 Key
          ##   path: keys ## 挂载到指定的子路径
    restartPolicy: Never
```

方式二：环境变量中使用 ConfigMap 类型

YAML 示例如下：

```
  apiVersion: v1
  kind: Pod
  metadata:
    name: nginx
  spec:
    containers:
      - name: nginx
        image: nginx:latest
        env:
          - name: key1
            valueFrom:
              configMapKeyRef:
                name: test-config ## 设置来源 ConfigMap 文件名
                key: test-config.key1  ## 设置该环境变量的 Value 来源项
    restartPolicy: Never
```

Secret管理

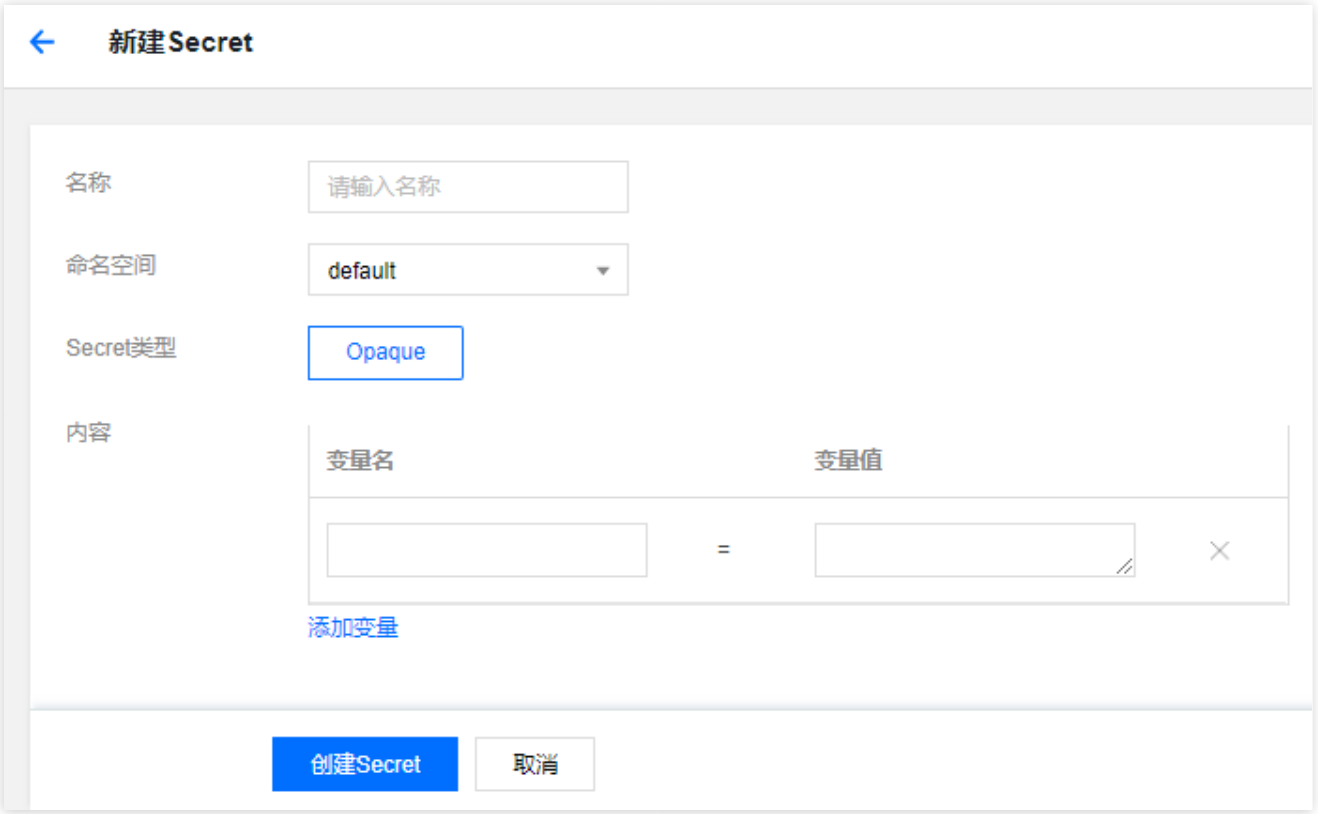
简介

Secret 可用于存储密码、令牌、密钥等敏感信息，降低直接对外暴露的风险。 Secret 是有 key-value 类型的键值对，您可以通过控制台的 Kubectl 工具创建对应的 Secret 对象，也可以通过挂载数据卷、环境变量或在容器的运行命令中使用 Secret。

Secret 控制台操作指引

创建 Secret

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建 Secret 的集群【ID/名称】，进入待创建 Secret 的集群管理页面。
4. 选择【配置管理】 > 【Secret】，进入“Secret”信息页面。
5. 单击【新建】，进入“新建Secret”页面。



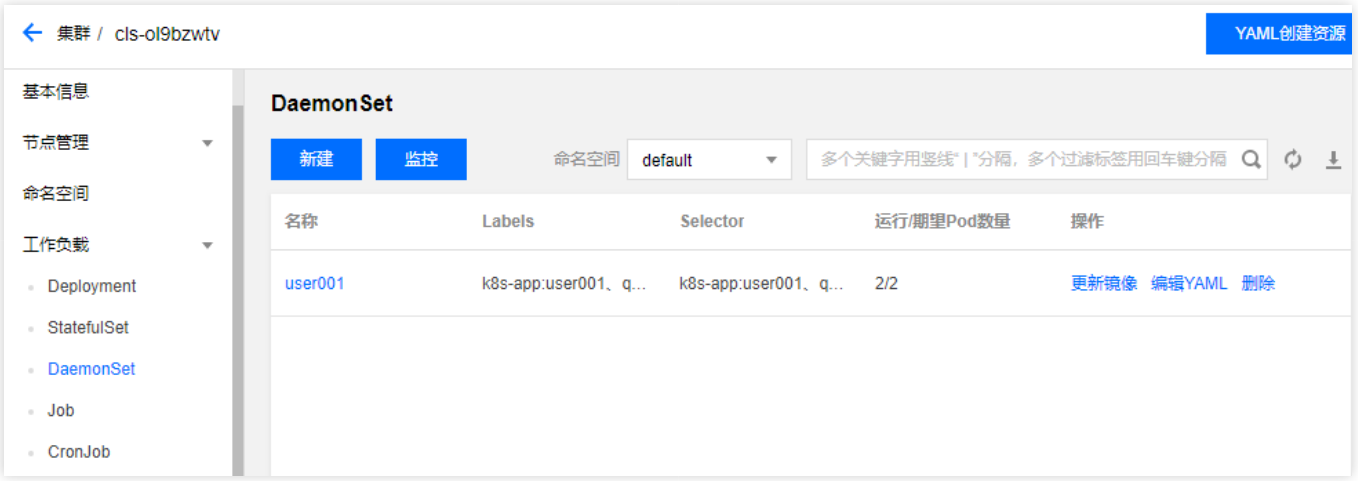
6. 根据实际需求，设置 Secret 参数。关键参数信息如下：
   - 名称：自定义。
   - 命名空间：根据实际需求进行选择命名空间类型。
   - 内容：根据实际需求，设置变量名和变量值。

7. 单击【创建Secret】，完成创建。

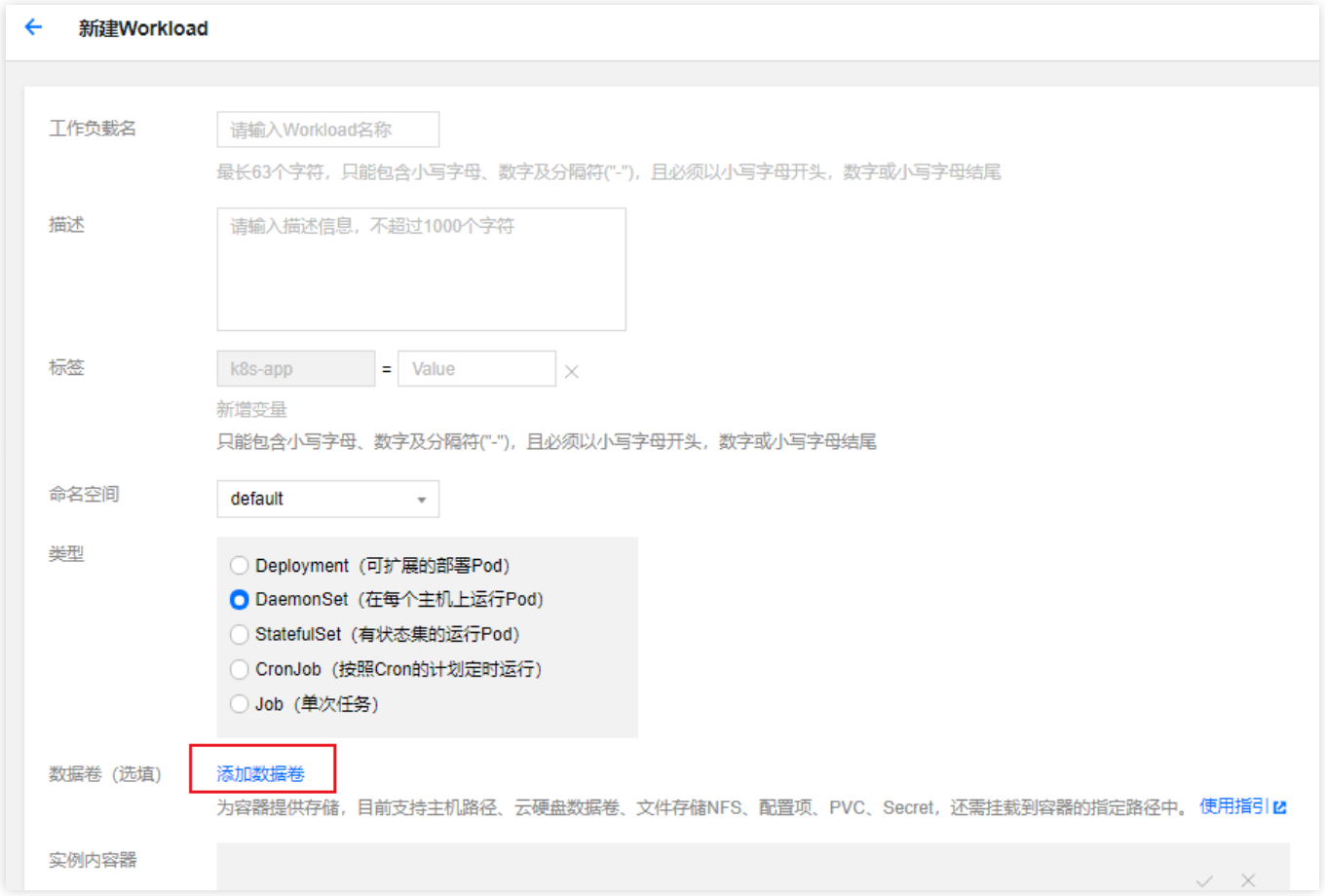
使用 Secret

方式一：数据卷使用 Secret 类型

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要部署 Workload 的集群【ID/名称】，进入待部署 Workload 的集群管理页面。
4. 在 “工作负载” 下，任意选择 Workload 类型，进入对应的信息页面。例如，选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。



5. 单击【新建】，进入“新建Workload”页面。
6. 根据页面信息，设置工作负载名、命名空间等信息。并在 “数据卷” 中，单击【添加数据卷】，添加数据卷。



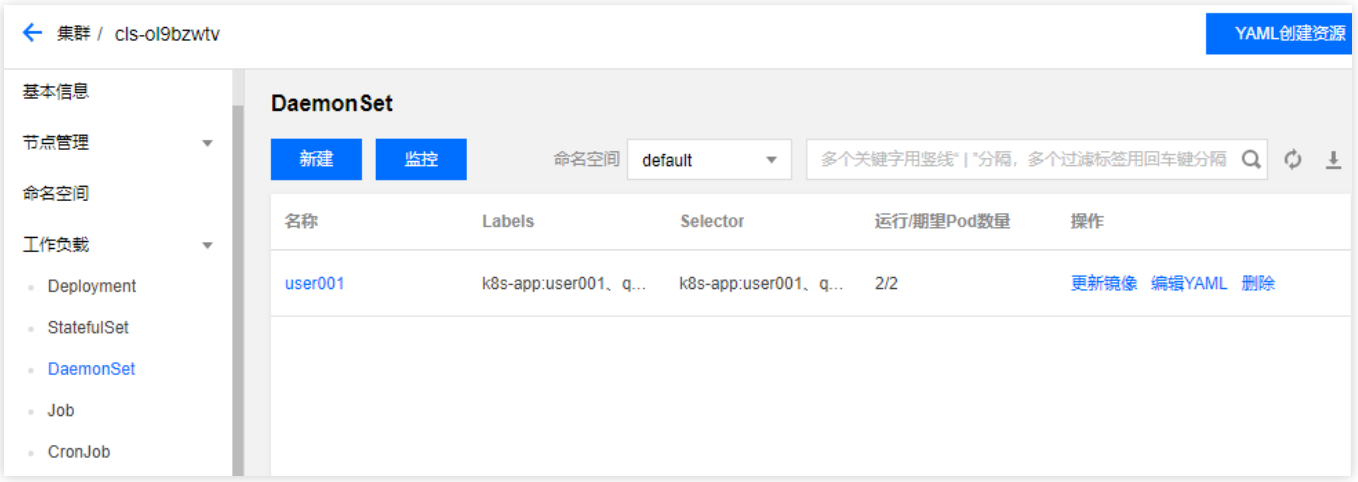
7. 选择 “使用Secret” 方式，填写名称，单击【选择Secret】。



8. 在弹出的 “设置Secret” 窗口中，配置挂载点，单击【确认】。
9. 单击【创建Workload】，完成创建。

方式二：环境变量中使用 Secret 类型

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要部署 Workload 的集群【ID/名称】，进入待部署 Workload 的集群管理页面。
4. 在 “工作负载” 下，任意选择 Workload 类型，进入对应的信息页面。例如，选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。如下图所示：



5. 单击【新建】，进入“新建Workload”页面。
6. 根据页面信息，设置工作负载名、命名空间等信息。选择 “Secret” 环境变量方式，并根据实际需求选择资源。
7. 单击【创建Workload】，完成创建。

更新 Secret

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 YAML 的集群【ID/名称】，进入待更新 YAML 的集群管理页面。
4. 选择【配置管理】 > 【Secret】，进入“Secret”信息页面。
5. 在需要更新 YAML 的 Secret 行中，单击【编辑YAML】，进入更新 Secret 页面。
6. 在“更新Secret”页面，编辑 YAML，单击【完成】，即可更新 YAML。 > 如需修改 key-values，编辑 YAML 中 data 的参数值，单击【完成】，即可完成更新。

更新 Secret 配置

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要更新 配置的集群【ID/名称】，进入待更新的集群管理页面。
4. 选择【配置管理】 > 【Secret】，进入“Secret”信息页面。
5. 在需要更新配置的 Secret 行中，单击【更新配置】，进入更新页面。
6. 在更新页面，编辑配置项，单击【完成】，即可更新。

Kubectl 操作 Secret 指引

创建 Secret

方式一：通过指定文件创建 Secret

1. 执行以下命令，获取 Pod 的用户名和密码。

```
$ echo -n 'username' > ./username.txt
$ echo -n 'password' > ./password.txt
```

2. 执行 Kubectl 命令，创建 Secret。

```
$ kubectl create secret generic test-secret --from-file=./username.txt --from-file=./password.txt
secret "testSecret" created
```

3. 执行以下命令，查看 Secret 详情。

```
kubectl describe secrets/ test-secret
```

方式二：YAML 文件手动创建

> 通过 YAML 手动创建 Secret，需提前将 Secret 的 data 进行 Base64 编码。

```
apiVersion: v1
kind: Secret
metadata:
  name: test-secret
type: Opaque
data:
  username: dXNlcm5hbWU=  ## 由echo -n 'username' | base64生成
  password: cGFzc3dvcmQ=  ## 由echo -n 'password' | base64生成
```

使用 Secret

方式一： 数据卷使用 Secret 类型

YAML 示例如下：

```
apiVersion: v1
 kind: Pod
 metadata:
   name: nginx
 spec:
   containers:
     - name: nginx
       image: nginx:latest
       volumeMounts:
        name: secret-volume
```

```
      mountPath: /etc/config
  volumes:
      name: secret-volume
      secret:
        name:  test-secret ## 设置 secret 来源
        ## items:  ## 设置指定 secret的 Key 挂载
        ##   key: username  ## 选择指定 Key
        ##   path: group/user ## 挂载到指定的子路径
        ##   mode: 256  ## 设置文件权限
  restartPolicy: Never
```

方式二： 环境变量中使用 Secret 类型

YAML 示例如下：

```
apiVersion: v1
 kind: Pod
 metadata:
   name: nginx
 spec:
   containers:
     - name: nginx
       image: nginx:latest
       env:
         - name: SECRET_USERNAME
           valueFrom:
             secretKeyRef:
               name: test-secret ## 设置来源 Secret 文件名
               key: username  ## 设置该环境变量的 Value 来源项
  restartPolicy: Never
```

Ingress 管理
Nginx 类型 Ingress
概述

Nginx-ingress 介绍

Nginx 可以用作反向代理、负载平衡器和 HTTP 缓存。
Nginx-ingress 是使用 Nginx 作为反向代理和负载平衡器的 Kubernetes 的 Ingress 控制器。您可以部署 Nginx-ingress 组件，在集群中使用 Nginx-ingress。容器服务 TKE 提供了产品化的能力，帮助您在集群内安装和使用 Nginx-ingress。

Nginx-ingress 名词解释

- Nginx-ingress 组件：在 TKE 中使用 Nginx-ingress 的入口，您可以在集群的组件页面一键安装部署 Nginx-ingress。
- Nginx-ingress 实例：一个集群中可部署多个 Nginx-ingress（例如一个用于公网，一个用于内网）。在 Kubernetes 中对应一个 CRD，创建一个 Nginx-ingress 实例会在集群中自动创建 Nginx-ingress-controller、service、configmap 等 Kubernetes 资源。
- Nginx-ingress-controller：实际 Nginx 负载，同时 controller 会 watch kubernetes ingress 对象的变化更新在集群中，Nginx 负载的转发配置即 nginx.conf 文件。

Nginx-ingress 相关操作

Nginx-Ingress 相关操作及功能如下，您可参考以下文档进一步了解：

- Nginx-ingress 安装
- 使用 Nginx-ingress 对象接入集群外部流量

安装 Nginx-ingress 实例

安装 NginxIngress 组件

1. 登录容器服务控制台，选择左侧导航栏中的集群。
2. 在集群管理页面，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航中的组件管理，在组件管理页面，单击新建。
4. 在新建组件页面中勾选 NginxIngress。
5. 单击完成即可安装组件。安装完成后，您可以在组件管理中查看组件详情。

使用 Nginx-ingress 对象接入集群外部流量

前提条件

- 已登录容器服务控制台。
- 集群内已部署 NginxIngress 组件。
- 已安装并创建业务需要的 Nginx-ingress 实例。

使用方法

Nginx-ingress 控制台操作指引

1. 登录容器服务控制台，在左侧导航栏中单击集群。
2. 进入集群管理页面，单击已安装 Nginx-ingress 组件的集群 ID，进入集群详情页。
3. 选择组件管理，单击更新配置，进入更新页面。
   根据实际需求，设置 Ingress 参数。
   - Ingress 类型：选择 Nginx Ingress Controller。
   - Class：选择一个 Ngixn Ingress 实例，若无则单击右侧的立即创建 Nginx 负载均衡器。
   - 转发规则：需自行设置。
4. 单击创建 Ingress。

Kubectl 操作 Nginx-ingress 指引

在 Kubernetes 中引入 IngressClass 资源和 ingressClassName 字段之前，Ingress 类由 Ingress 中的 kubernetes.io/ingress.class 注解指定。
示例如下：

```
metadata:
  name:
  annotations:
    kubernetes.io/ingress.class: "nginx-pulic". ## 对应 TKE 集群 Nginx-ingress 组件中的 Nginx-ingress 实例名称
```

相关操作

为 Nginx 类型 Ingress 对象可配置注解。

Nginx-ingress 对象使用模型

当多个 Ingress 对象作用于一个 Nginx 实体时：

- 按 CreationTimestamp 字段对 Ingress 规则排序，即先按旧规则。
- 如果在多个 Ingress 中为同一主机定义了相同路径，则最早的规则将获胜。
- 如果多个 Ingress 包含同一主机的 TLS 部分，则最早的规则将获胜。
- 如果多个 Ingress 定义了一个影响 Server 块配置的注释，则最早的规则将获胜。
- 按每个 hostname 创建 NGINX Server。
- 如果多个 Ingress 为同一 host 定义了不同的路径，则 ingress-controller 合并这些定义。
- 多个 Ingress 可以定义不同的注释。这些定义在 Ingress 之间不共享。
- Ingress 的注释将应用于 Ingress 中的所有路径。

触发更新 nginx.conf 机制

以下内容描述了需要重新加载 nginx.conf 的情况：

- 创建新的 Ingress 对象。
- 为 Ingress 添加新的 TLS。
- Ingress 注解的更改不仅影响上游配置，而且影响更大。例如 load-balance 注释不需要重新加载。
- 为 Ingress 添加/删除路径。
- 删除 Ingress、Ingress 的 Service、Secret。
- Ingress 关联的对象状态不可知，例如 Service 或 Secret。
- 更新 Secret。

CLB 类型 Ingress

概述

Service 提供了基于四层网络的集群内容器服务的暴露能力，Service 暴露类型（例如 ClusterIP、NodePort 或 LoadBalancer）均基于四层网络服务的访问入口，缺少基于七层网络的负载均衡、SSL 或基于名称的虚拟主机等七层网络能力。Ingress 提供七层网络下 HTTP、 HTTPS 协议服务的暴露，及七层网络下的常见能力。

Ingress 基本概念

Ingress 是允许访问到集群内 Service 规则的集合，您可以通过配置转发规则，实现不同 URL 可以访问到集群内不同的 Service。为了使 Ingress 资源正常工作，集群需运行 Ingress Controller，容器服务在集群内默认启用了基于负载均衡器实现的 TKE Ingress Controller。

Ingress 生命周期管理

Ingress 对外服务的能力依赖于负载均衡所提供的资源，因此服务资源管理也是 Ingress 的重要工作之一。Ingress 在资源的生命周期管理上会使用以下标签：

标签	描述
tke-createdBy-flag = yes	- 标识该资源是容器服务创建，拥有该标签的 Ingress 会在销毁时删除对应资源。 - 如果没有该标签，Ingress 会在销毁时，仅删除负载均衡内的监听器资源，而不删除负载均衡自身。
tke-clusterId = <clusterId>	- 标识该资源被哪一个 Cluster 所使用。 - Ingress 会在销毁时，删除对应标签（ClusterId 需正确）。
tke-lb-ingress-uuid = <Ingress UUID>	- 标识该资源被哪一个 Ingress 所使用。 - Ingress 目前不支持复用，当用户指定 Ingress 使用已有负载均衡时，标签的值若不正确会被拒绝。 - Ingress 会在销毁时，删除对应标签（Ingress UUID 需正确）。

> 说明
>
> 上述标签对用户只读，请不要修改或删除上述标签，否则可能导致资源泄漏。
>
> 若用户使用了已有负载均衡，则 Service 仅会使用该负载均衡，而不会删除该负载均衡。

Ingress Controller 使用方法

除了服务提供的 TKE Ingress Controller 以外，Kubernetes 社区还有各种类型的第三方 Ingress Controller，这些 Ingress 控制器均为完成服务的七层网络暴露。Kubernetes 社区基本支持使用 kubernetes.io/ingress.class 注解用于区分各种 Ingress 控制器，以确定当前 Ingress 资源应被哪一个控制器处理。TKE Ingress Controller 也支持使用该注解，具体规则及使用建议如下：

- 当 Ingress 资源没有描述注解 kubernetes.io/ingress.class 时，TKE Ingress Controller 会管理当前 Ingress 资源。
- 当 Ingress 资源有注解 kubernetes.io/ingress.class 且值为 qcloud 时，TKE Ingress Controller 会管理当前 Ingress 资源。
- 当 Ingress 资源修改注解 kubernetes.io/ingress.class 的内容时，TKE Ingress Controller 会根据注解内容将其纳入或脱离管理范围，其操作会涉及到资源的创建与释放。
- 当您确认完全不需要使用 TKE Ingress Controller 时，可以将集群中的 Deployment（kube-system:l7-lb-controller）的工作副本数量调整为0，从而关闭 TKE Ingress Controller 功能。

  > 说明
  >
  > - 关闭该功能前，请确保集群中没有被 TKE Ingress Controller 管理的 Ingress 资源，避免出现负载均衡资源释放失败的情况。
  > - 若用户在负载均衡上面开启了删除保护，或者使用私有连接，则删除 Service 时，不会删除该负载均衡。

Ingress 基本功能

简介

Ingress 是允许访问到集群内 Service 的规则的集合，您可以通过配置转发规则，实现不同 URL 可以访问到集群内不同的 Service。
为了使 Ingress 资源正常工作，集群必须运行 Ingress-controller。TKE 服务在集群内默认启用了基于负载均衡器实现的 l7-lb-controller ，支持 HTTP、HTTPS，同时也支持在集群内自建其他 Ingress 控制器，您可以根据您的业务需要选择不同的 Ingress 类型。

注意事项

- 确保您的容器业务不和 CVM 业务共用一个 CLB。
- 不支持您在 CLB 控制台操作 TKE 管理的 CLB 的监听器、转发路径、证书和后端绑定的服务器，您的更改会被 TKE 自动覆盖。
- 使用已有的 CLB 时：
  - 只能使用通过 CLB 控制台创建的负载均衡器，不支持复用由 TKE 自动创建的 CLB。
  - 不支持多个 Ingress 复用 CLB。
  - 不支持 Ingress 和 Service 共用 CLB。
  - 删除 Ingress 后，复用 CLB 绑定的后端云服务器需要自行解绑，同时会保留一个 tag tke-clusterId: cls-xxxx ，需自行清理。
  - Ingress 和 CLB 之间配置的管理和同步是由以 CLB ID 为名字的 LoadBalancerResource 类型的资源对象，请勿对该 CRD 进行任何操作，否则容易导致 Ingress 失效。

Ingress 控制台操作指引

创建 Ingress

1. 登录 容器服务控制台，单击左侧导航栏中的【集群】。
2. 在集群管理页面，单击需要创建 Ingress 的集群 ID。
3. 在集群详情页，选择【服务】 > 【Ingress】。
4. 在 Ingress 页面，单击【新建】。
5. 在新建 Ingress 页面，根据实际需求，设置 Ingress 参数。关键参数信息如下：
   - Ingress 名称：自定义。
   - Ingress 类型：根据实际需求进行选择。
   - 网络类型：默认为“公网”，请根据实际需求进行选择。

- IP 版本：提供 IPv4 和 IPv6 NAT64 两种版本，请根据实际需求进行选择。
- 负载均衡器：可自动创建或使用已有 CLB。
- 命名空间：根据实际需求进行选择。
- 运营商类型：支持 BGP（多线）运营商类型。

  转发配置：协议默认为 Http，请根据实际情况进行选择。如果协议选择 Https 则需绑定服务器证书，以保证访问安全。

  证书详情请参见 SSL 证书格式要求及格式转换说明。

6. 单击【创建Ingress】，完成创建。

更新 Ingress

更新 YAML

1. 登录容器服务控制台，单击左侧导航栏中的【集群】。
2. 在集群管理页面，单击集群 ID，进入待更新 YAML 的集群管理页面。
3. 在集群详情页，选择【服务】 > 【Ingress】。
4. 在需要更新 YAML 的 Ingress 行中，单击【编辑 YAML】。
5. 在更新 Ingress 页面，编辑 YAML，单击【完成】。

更新转发规则

1. 登录容器服务控制台，单击左侧导航栏中的【集群】。
2. 在集群管理页面，单击集群 ID，进入待更新 YAML 的集群管理页面。
3. 在集群详情页，选择【服务】 > 【Ingress】。
4. 在需要更新 YAML 的 Ingress 行中，单击【更新转发配置】。

   在更新转发配置页面，根据实际需求，修改转发配置。
5. 单击【更新转发配置】，即可完成更新。

Kubectl 操作 Ingress 指引

YAML 示例

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: qcloud ## 可选值：qcloud（CLB类型ingress），nginx（nginx-ingress），tr
aefik
    ## kubernetes.io/ingress.existLbId: lb-xxxxxxxx        ##指定使用已有负载均衡器创建公网/内网访问的In
gress
    ## kubernetes.io/ingress.subnetId: subnet-xxxxxxxx  ##若是创建CLB类型内网ingress需指定该条annota
tion
```

```
  name: my-ingress
  namespace: default
spec:
  rules:
  - host: localhost
    http:
      paths:
      - backend:
          serviceName: non-service
          servicePort: 65535
        path: /
```

- kind：标识 Ingress 资源类型。
- metadata：Ingress 的名称、Label 等基本信息。
- metadata.annotations：Ingress 的额外说明，可通过该参数设置 TKE 的额外增强能力。
- spec.rules：Ingress 的转发规则，配置该规则可实现简单路由服务、基于域名的简单扇出路由、简单路由默认域名、配置安全的路由服务等。

annotations: 使用已有负载均衡器创建公网/内网访问的 Ingress

如果您已有的应用型 CLB 为空闲状态，需要提供给 TKE 创建的 Ingress 使用，或期望在集群内使用相同的 CLB ，您可以通过以下 annotations 进行设置：

> 说明
>
> 请了解 注意事项 后开始使用。

```
metadata:
  annotations:
    kubernetes.io/ingress.existLbId： lb-6swtxxxx
```

annotations: 创建 CLB 类型内网 Ingress

如果您需要使用内网负载均衡，可以通过以下 annotations 进行设置：

```
metadata:
  annotations:
    kubernetes.io/ingress.subnetId: subnet-xxxxxxxx
```

说明事项

如果您使用的是 IP 带宽包账号，在创建公网访问方式的服务时需要指定以下两个 annotations 项：

- kubernetes.io/ingress.internetChargeType 公网带宽计费方式，可选值有：

- TRAFFIC_POSTPAID_BY_HOUR（按使用流量计费）
- BANDWIDTH_POSTPAID_BY_HOUR（按带宽计费）

- kubernetes.io/ingress.internetMaxBandwidthOut 带宽上限，范围：[1,2000] Mbps。

例如：

```
metadata:
annotations:
  kubernetes.io/ingress.internetChargeType: TRAFFIC_POSTPAID_BY_HOUR
  kubernetes.io/ingress.internetMaxBandwidthOut: "10"
```

创建 Ingress

1. 参考 YAML 示例，准备 Ingress YAML 文件。
2. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
3. 执行以下命令，创建 Ingress YAML 文件。

   ```
   kubectl create -f Ingress YAML 文件名称
   ```

   例如，创建一个文件名为 my-ingress.yaml 的 Ingress YAML 文件，则执行以下命令：

   ```
   kubectl create -f my-ingress.yaml
   ```

4. 执行以下命令，验证创建是否成功。

   ```
   kubectl get ingress
   ```

   返回类似以下信息，即表示创建成功。

   ```
   NAME          HOSTS       ADDRESS   PORTS     AGE
   clb-ingress   localhost             80        21s
   ```

更新 Ingress

方法一

执行以下命令，更新 Ingress。

kubectl edit ingress/[name]

方法二

1. 手动删除旧的 Ingress。
2. 执行以下命令，重新创建 Ingress。

   kubectl create/apply

Ingress 使用已有 CLB

容器服务 TKE 具备通过 kubernetes.io/ingress.existLbId: <LoadBalanceId> 注解使用已有负载均衡的功能，您可使用该注解指定 Ingress 关联的负载均衡实例。

> 说明
>
> Ingress 与 Service 的区别：Ingress 不支持多个实例使用同一个负载均衡实例，即不支持复用功能。

注意事项

- 请确保您的容器业务不与云服务器 CVM 业务共用一个负载均衡资源。
- 不支持在负载均衡控制台操作 Ingress Controller 管理的负载均衡监听器以及后端绑定的服务器，更改会被 Ingress Controller 自动覆盖。
- 使用已有负载均衡时：
  - 不支持多个 Ingress 复用同一个负载均衡。
  - 指定的负载均衡不能存在任何已有监听器。如已存在，请提前删除。
  - 仅支持使用通过负载均衡控制台创建的负载均衡器，不支持使用由 Service Controller 自动创建和管理的负载均衡，即 Service 和 Ingress 不能混用同一个负载均衡。
  - Ingress Controller 不负责负载均衡的资源管理，即在 Ingress 资源删除时，负载均衡资源不会被删除回收。

使用场景

使用负载均衡对外提供服务

Ingress Controller 管理负载均衡生命周期时，仅支持购买按量计费的资源。
通过注解控制 Ingress 使用已有负载均衡，将负载均衡的生命周期管理从 Ingress Controller 中剥离。示例如下：

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.existLbId: lb-mgzu3mpx
  name: nginx-ingress
spec:
  rules:
    - http:
        paths:
          - backend:
              serviceName: nginx-service
```

```
      servicePort: 80
    path: /
```

kubernetes.io/ingress.existLbId: lb-mgzu3mpx 注解表明了该 Ingress 将使用已有负载均衡 lb-mgzu3mpx 进行 Ingress 服务配置。

Ingress 证书配置

操作场景

本文档介绍 Ingress 证书使用相关的内容，您可在以下场景中进行 Ingress 证书配置：

- 创建 Ingress 选用 HTTPS 监听协议时，选用合适的服务器证书能够确保访问安全。
- 为所有的 HTTPS 域名绑定同一个证书，简化配置 Ingress 下所有 HTTPS 规则的证书，使更新操作更加便捷。
- 为不同的域名绑定不同的证书，改善服务器与客户端 SSL/TLS。

注意事项

- 需提前创建需配置的证书，详情请参见下文通过控制台新建服务器证书。
- 需使用 Secret 形式来设置 Ingress 证书。云容器服务 TKE Ingress 会默认创建同名 Secret，其内容包含证书 ID。
- 若您需要更换证书，建议在证书平台新建一个证书，然后更新 Secret 的证书 ID。因为集群中组件的同步会以 Secret 的声明为准，若您直接在其他证书服务、负载均衡服务上更新的证书，将会被 Secret 里的内容还原。
- Secret 证书资源需和 Ingress 资源放置在同一个 Namespace 下。
- 由于控制台默认会创建同名 Secret 证书资源，若同名 Secret 资源已存在，则 Ingress 将无法创建。
- 通常情况下，在创建 Ingress 时，不会复用 Secret 关联的证书资源。但仍支持在创建 Ingress 复用 Secret 关联的证书资源，更新 Secret 时，会同步更新所有引用该 Secret 的 Ingress 的证书。
- 为域名增加匹配证书后，将同步开启负载均衡 CLB SNI 功能（不支持关闭）。若删除证书对应的域名，则该证书将默认匹配 Ingress 所对应的 HTTPS 域名。
- 传统型负载均衡不支持基于域名和 URL 的转发，由传统型负载均衡创建的 Ingress 不支持配置多证书。

示例

TKE 支持通过 Ingress 中的 spec.tls 的字段，为 Ingress 创建的 CLB HTTPS 监听器配置证书。其中，secretName 为包含证书 ID 的 Kubernetes Secret 资源。示例如下：

Ingress

通过 YAML 创建：

```
spec:
  tls:
  - hosts:
    - www.abc.com
```

```
    secretName: secret-tls-2
```

Secret

通过YAML 创建

```
apiVersion: v1
stringData:
  qcloud_cert_id: Xxxxxxxx ## 配置证书 ID 为 Xxxxxxxx
  qcloud_ca_cert_id: Xxxxxxxx ## 配置证书 ID 为 Xxxxxxxx。仅配置双向证书时需要，见下面的“注意”。
kind: Secret
metadata:
  name: abcd-com-cert
  namespace: default
type: Opaque
```

通过控制台创建

您可以通过容器服务控制台创建，操作详情可参考 创建 Secret。
在“新建Secret” 页面，Secret 主要参数配置如下：

- 名称：自定义，本文以 cos-secret 为例。
- Secret类型：选择 Opaque，该类型适用于保存密钥证书和配置文件，Value 将以 Base64 格式编码。
- 生效范围：按需选择，需确保与 Ingress 在同一 Namespace 下。
- 内容：变量名设置为 qcloud_cert_id ，变量值配置为服务器证书所对应的证书 ID。

  > 注意
  >
  > 若您需要配置双向证书，则 Secret 除了要添加“服务器证书”外，还需要添加“客户端CA证书”。此时该 Secret 还需要额外添加一个键值对：变量名为： qcloud_ca_cert_id ，变量值配置为“客户端CA证书”所对应的证书ID。

Ingress 证书配置行为

- 仅配置单个 spec.secretName 且未配置 hosts 的情况下，将会为所有的 HTTPS 的转发规则配置该证书。示例如下：

  ```
  spec:
    tls:
    - secretName: secret-tls
  ```

- 支持配置一级泛域名统配。 示例如下：

```
spec:
  tls:
  - hosts:
    - *.abc.com
    secretName: secret-tls
```

- 若同时配置证书与泛域名证书，将优先选择一个证书。 示例如下， www.abc.com 将会使用 secret-tls-2 中描述的证书。

```
spec:
  tls:
  - hosts:
    - *.abc.com
    secretName: secret-tls-1
  - hosts:
    - www.abc.com
    secretName: secret-tls-2
```

- 对已使用多个证书的 Ingress 进行更新时，TKE Ingress controller 将进行以下行为判断：
  - HTTPS 的 rules.host 无任何匹配时，若判断不通过，则不能提交更新。
  - HTTPS 的 rules.host 匹配中单个 TLS 时，可提交更新，并为该 host 配置对 Secret 中对应的证书。
  - 修改 TLS 的 SecretName 时仅校验 SecretName 的存在性，而不校验 Secret 内容， Secret 存在即可提交更新。

    > 注意
    >
    > 请确保 Secret 中证书 ID 符合要求。

操作步骤

通过控制台新建服务器证书

> 说明
>
> 若您已具备需配置的证书，则请跳过此步骤。

1. 登录负载均衡控制台，选择左侧导航栏中的证书管理。
2. 在“证书管理”页面中，单击【新建】。
3. 在弹出的“新建证书”窗口中，参考以下信息进行设置。
   - 证书名称：自定义设置。
   - 证书类型：选择“服务器证书”。服务器证书即 SSL 证书（SSL Certificates）。基于 SSL 证书，可将站

点由 HTTP（Hypertext Transfer Protocol）切换到 HTTPS（Hyper Text Transfer Protocol over Secure Socket Layer），即基于安全套接字层（SSL）进行安全数据传输的加密版 HTTP 协议。
    - 证书内容：根据实际情况填写证书内容。
    - 密钥内容：仅当证书类型选择为“服务器证书”时，该选项才会显示。
4. 单击提交即可完成创建。

创建使用证书的 Ingress 对象

注意事项：

- 当控制台创建的 Ingress 开启 HTTPS 服务，会先创建同名的 Secret 资源用于存放证书 ID，并在 Ingress 中使用并监听该 Secret。
- TLS 配置域名与证书的对应关系如下：
    - 可以使用一级泛域名统配。
    - 若域名匹配中多个不同的证书，将随机选择一个证书，不建议相同域名使用不同证书。
    - 需为所有 HTTPS 域名配置证书，否则会创建不通过。

操作步骤：

参考 创建 Ingress 完成 Ingress 新建，其中监听端口勾选 Https:443。

修改证书

注意事项：

- 如果您需要修改证书， 请确认所有使用该证书的 Ingress。如用户的多个 Ingress 配置使用同一个 Secret 资源，那么这些 Ingress 对应 CLB 的证书会同步变更。
- 证书需要通过修改 Secret 进行修改， Secret 内容中包含您使用的云证书的 ID。

操作步骤：

1. 执行以下命令，使用默认编辑器打开需修改的 Secret。其中， [secret-name] 需更换为需修改的 Secret 的名称。

```
kubectl edit secrets [secret-name]
```

2. 修改 Secret 资源，将 qcloud_cert_id 的值修改为新的证书 ID。
与创建 Secret 相同，修改 Secret 证书 ID 需要进行 Base64 编码，请根据实际需求选择 Base64 手动编码或者指定 stringData 进行 Base64 自动编码。

更新 Ingress 对象

通过控制台更新

1. 登录容器服务控制台 ，选择左侧导航栏中的集群。
2. 在“集群管理”页面，选择需修改 Ingress 的集群 ID。
   在集群详情页，选择左侧【服务】 > 【Ingress】。
3. 单击目标 Ingress 所在行右侧的【更新转发配置】。
4. 在“更新转发配置”页面中，根据实际情况进行转发配置规则更新。
5. 单击【更新转发配置】即可完成更新操作。

通过 YAML 更新

执行以下命令，使用默认编辑器打开需修改的 ingress，修改 yaml 文件并保存即可完成更新操作。

```
kubectl edit ingress <ingressname> -n <namespaces>
```

Ingress Annotation 说明

您可以通过以下 Annotation 注解配置 Ingress，以实现更丰富的负载均衡的能力。

注解使用方式

```
apiVersion:
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: "qcloud"
  name: test
.....
```

Annotation 集合

kubernetes.io/ingress.class

说明：
配置 Ingress 类型。当前组件管理未配置该注解，或注解内容为 qcloud 的 Ingress 资源。
使用示例：

```
kubernetes.io/ingress.class: "qcloud"
```

kubernetes.io/ingress.qcloud-loadbalance-id

说明：
只读注解，组件提供当前 Ingress 引用的负载均衡 LoadBalanceId。
使用示例：

```
kubernetes.io/ingress.qcloud-loadbalance-id: "lb-3imskkfe"
```

ingress.finance.cloud.tencent.com/loadbalance-nat-ipv6

说明：
只读注解，当用户配置或申请的为 NAT IPv6负载均衡时，提供 IPv6地址。

ingress.finance.cloud.tencent.com/loadbalance-ipv6

说明：
只读注解，当用户配置或申请的为 FullStack IPv6负载均衡时，提供 IPv6地址。

kubernetes.io/ingress.internetChargeType

说明：
负载均衡的付费类型，当前仅在创建时支持配置，创建后不支持修改付费类型，创建后修改本注解无效。
指定创建负载均衡时，负载均衡的付费类型。请配合 kubernetes.io/ingress.internetMaxBandwidthOut 注解一起使用。
可选值：

- TRAFFIC_POSTPAID_BY_HOUR 按流量按小时后计费。
- BANDWIDTH_POSTPAID_BY_HOUR 按带宽按小时后计费。
  使用示例：
  kubernetes.io/ingress.internetChargeType: "TRAFFIC_POSTPAID_BY_HOUR"

kubernetes.io/ingress.internetMaxBandwidthOut

说明：
CLB 带宽设置，当前仅在创建时支持配置，创建后不支持修改带宽，创建后修改本注解无效。
指定创建负载均衡时，负载均衡的最大出带宽，仅对公网属性的 LB 生效。需配合 kubernetes.io/ingress.internetChargeType 注解一起使用。
可选值：
范围支持1到2048，单位 Mbps。
使用示例：
kubernetes.io/ingress.internetMaxBandwidthOut: "2048"

kubernetes.io/ingress.extensiveParameters

说明：
该 Annotation 使用的是 CLB 创建时的参数，当前仅在创建时支持配置，创建后不支持修改，创建后修改本注解无效。
使用示例：

- 创建 NAT64 IPv6 实例：
  kubernetes.io/ingress.extensiveParameters: '{"AddressIPVersion":"IPV6"}'
- 创建 IPv6 实例：（ SubnetId 必填，而且需要分配IPv6网段。 MixIpTarget 可以提供混绑后端IPv4的能力，当您的后端不是IPv6时请添加该配置）
  kubernetes.io/ingress.extensiveParameters: '{"AddressIPVersion":"IPv6FullChain","SubnetId": "subnet-fqduxxxx"}'
  kubernetes.io/ingress.extensiveParameters: '{"AddressIPVersion":"IPv6FullChain","SubnetId": "subnet-fqduxxxx","MixIpTarget":true}'
- 购买电信负载均衡：
  kubernetes.io/ingress.extensiveParameters: '{"VipIsp":"CTCC"}'
- 指定可用区创建：
  kubernetes.io/ingress.extensiveParameters: '{"ZoneId":"ap-cityA-1"}'
- 创建时自定义 CLB 名字：

kubernetes.io/ingress.extensiveParameters: '{"LoadBalancerName":"my_custom_lb_name"}'

kubernetes.io/ingress.subnetId

说明：

指定创建内网类型的负载均衡，并指定负载均衡所属子网。

使用示例：

kubernetes.io/ingress.subnetId: "subnet-3swgntkk"

kubernetes.io/ingress.existLbId

说明：

指定使用已有负载均衡作为接入层入口资源。

> 注意
>
> 使用已有负载均衡时，需要保证其不包含其他监听器。

使用示例：

kubernetes.io/ingress.existLbId: "lb-342wppll"

Ingress Controllers 说明

Ingress Controllers 介绍

直通 CLB

直通 CLB 是基于负载均衡器 CLB 实现的 TKE Ingress Controller，可以配置实现不同 URL 访问到集群内不同的 Service。CLB 直接将流量通过 NodePort 转发至 Pod（CLB 直连 Pod 时直接转发到 Pod），一条 Ingress 配置绑定一个 CLB 实例（IP），适合仅需做简单路由管理，对 IP 地址收敛不敏感的场景。详情可参见 CLB 类型 Ingress。

Nginx Ingress Controller

Nginx Ingress Controller 是基于负载均衡器 CLB 和 Nginx 反向代理（容器化部署在集群内）的 Ingress Controller，通过 Annotations 扩展了原生 Kubernetes Ingress 的功能。CLB 后增加了一层代理（nginx），适合对接入层路由管理有更多诉求，及有 IP 地址收敛诉求的场景。详情可参见 Nginx 类型 Ingress。

Ingress Controllers 功能列表

模块	功能	应用型 CLB	Nginx Ingress Controller
流量管理	支持协议	http，https	http，https，http2，grpc，tcp，udp
	IP 管理	一条 Ingress 规则对应一个 IP（CLB）	多条 Ingress 规则对应一个 IP（CLB），IP 地址收敛
	特征路由	host，URL	更多特征支持：header、cookie 等
	流量行为	不支持	支持，重定向，重写等
	地域感知负载均衡	不支持	不支持
应用访问寻址	服务发现	单 Kubernetes 集群	单 Kubernetes 集群
安全	SSL 配置	支持	支持
	认证授权	不支持	支持
可观测性	监控指标	支持（需要在 CLB 中查看）	支持（云原生监控）
	调用追踪	不支持	不支持

模块	功能	应用型 CLB	Nginx Ingress Controller
	组件运维	关联 CLB 已托管，仅需集群内运行 TKE Ingress Controller	需集群内运行 Nginx Ingress Controller（控制面 + 数据面）

组件管理
扩展组件概述

扩展组件是容器服务 TKE 提供的扩展功能包，您可以根据业务诉求选择部署所需的扩展组件。扩展组件可帮助您管理集群的 Kubernetes 组件，包括组件部署、升级、更新配置和卸载等。

扩展组件类型

扩展组件分为基础组件和增强组件两种类型。

基础组件

基础组件是 TKE 功能依赖的软件包。例如，负载均衡组件 Service-controller、CLB-ingress-controller 及容器网络插件 tke-cni-agent 等。

> 说明：
>
> - 基础组件的升级、配置管理将由 TKE 统一进行管理维护，不建议您修改基础组件。
> - 基础组件的更新发布动态将通过邮件、短信等形式进行通知。

增强组件

增强组件是 TKE 提供的非必需部署的组件，您可以通过部署增强组件来使用 TKE 支持的增强功能，增强组件类型如下表所示：

组件名称	使用场景	组件介绍
COS（对象存储）	存储	该组件实现了 CSI 接口，可帮助容器集群使用对象存储。
CFS（文件存储）	存储	该组件实现了 CSI 接口，可帮助容器集群使用文件存储。
P2P（容器镜像加速分发）	镜像	该组件基于 P2P 技术，可应用于大规模 TKE 集群快速拉取 GB 级容器镜像，支持上千节点的并发拉取。
GpuManager（GPU 管理组件）	其他	该组件提供一个 All-in-One 的 GPU 管理器，可以在 TKE 集群中更细粒度的使用 GPU 资源。
HPC（定时修改副本数）	其他	HPC（HorizontalPodCronscaler）是一种可以对 K8s workload 副本数进行定时修改的组件，配合 HPC CRD 使用，最小支持秒级的定时任务。

扩展生命周期管理

组件安装

您可以通过以下两种方式安装增强组件：

- 集群创建页安装
- 组件管理页安装

集群创建页安装

1. 登录容器服务控制台，在左侧导航栏中选择【集群】。
2. 在“集群管理”页面，单击集群列表上方的【新建】。
3. 在“创建集群”页面，依次填写集群的【集群信息】、【选择机型】、【云服务器配置】及【组件配置】。
   您可以根据业务部署情况，按需选择合适的组件安装。点击每个组件卡片的【查看详情】可以查看该组件的介绍，部分组件需要您先完成【参数配置】。
   说明：
   - 组件安装为集群创建的非关键路径，安装失败不会影响集群的创建。
   - 组件安装需要占用集群的一定资源，不同组件的资源占用情况不同，单击【查看详情】查看每个组件的详细信息。
4. 单击【下一步】，检查并确认集群配置信息。
5. 单击【完成】，即可完成创建。

组件管理页安装

1. 登录 容器服务控制台，在左侧导航栏中选择【集群】。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的【组件管理】，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择【新建】，进入组件安装页面。
5. 选择需要安装的组件并单击【完成】即可。

组件卸载

1. 登录 容器服务控制台，在左侧导航栏中选择【集群】。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的【组件管理】，进入 “组件列表” 页面。
4. 在“组件列表”页面中，单击需要删除组件所在行右侧的【删除】。
5. 在弹出的 “删除资源” 窗口中，单击【确认】即可完成组件卸载。

COS 说明

简介

组件介绍

该COS组件实现 了CSI 的接口，可帮助您在容器集群中使用TCloudFinanceZone对象存储 COS。

使用场景

对象存储（Cloud Object Storage，COS）是TCloudFinanceZone提供的一种存储海量文件的分布式存储服务，用户可通过网络随时存储和查看数据。COS 使所有用户都能使用具备高扩展性、低成本、可靠和安全的数据存储服务。

限制条件

- 支持 Kubernetes 1.10 以上版本的集群。
- Kubernetes 1.12 版本的集群需要增加 kubelet 配置： --feature-gates=KubeletPluginsWatcher=false 。
- 在 TKE 中使用 COS，需要在集群内安装该扩展组件，将占用一定的系统资源。

使用方法

安装 COS 扩展组件

1. 登录容器服务控制台，在左侧导航栏中选择【集群】。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的【组件管理】，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择【新建】，并在“新建组件”页面中勾选 COS。
5. 单击【完成】即可创建组件。

使用对象存储 COS

您可在 TKE 集群中为工作负载挂载对象存储。

CFS 说明

简介

组件介绍

该CFS 组件实现了 CSI 的接口，可帮助您在容器集群中使用TCloudFinanceZone文件存储。

> 注意
> 1.12 集群需要修改 kubelet 配置，增加 \--feature-gates=KubeletPluginsWatcher=false\ 。

部署在集群内的 Kubernetes 对象

kubernetes对象名称	类型	默认占用资源	所属Namespaces
csi-provisioner-cfsplugin	StatefulSet	-	kube-system
csi-nodeplugin-cfsplugin	DaemonSet	-	kube-system
csi-provisioner-cfsplugin	Service	1C2G	kube-system

使用场景

文件存储 CFS 提供了可扩展的共享文件存储服务，可与 CVM、容器服务 TKE、批量计算等服务搭配使用。CFS 提供了标准的 NFS 及 CIFS/SMB 文件系统访问协议，为多个 CVM 实例或其他计算服务提供共享的数据源，支持弹性容量和性能的扩展，现有应用无需修改即可挂载使用，是一种高可用、高可靠的分布式文件系统，适合于大数据分析、媒体处理和内容管理等场景。
CFS 接入简单，您无需调节自身业务结构，或者是进行复杂的配置。只需三步即可完成文件系统的接入和使用：创建文件系统，启动服务器上文件系统客户端，挂载创建的文件系统。通过 CFS-CSI 扩展组件，您可以快速在容器集群中通过标准原生 Kubernetes 使用 CFS，详情请参见 CFS 使用场景。

限制条件

- CFS 自身限制可参见 CFS 系统限制。
- 在 TKE 中使用 CFS，需要在集群内安装该扩展组件，这将占用一定的系统资源。

操作步骤

安装并设置 CFS 扩展组件

1. 登录容器服务控制台，在左侧导航栏中选择【集群】。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的【组件管理】，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择【新建】，并在“新建组件”页面中勾选 CFS。
5. 单击【完成】即可创建组件。

创建 CFS 类型 StroageClass

1. 在“集群管理”页面单击使用 CFS 的集群 ID，进入集群详情页。
2. 在左侧导航栏中选择【存储】>【StorageClass】，单击【新建】进入 “新建StorageClass” 页面。
3. 根据实际需求，创建 CFS 类型的 StorageClass。
4. 单击【创建StorageClass】，完成创建。

创建 PersistentVolumeClaim

1. 在“集群管理”页面单击使用 CFS 的集群 ID，进入集群详情页。
2. 在左侧导航栏中选择【存储】>【PersistentVolumeClaim】，单击【新建】进入 “新建 PersistentVolumeClaim” 页面。
3. 根据实际需求，创建 CFS 类型 PersistentVolumeClaim，选择上述步骤创建的 StorageClass。
4. 单击【创建PersistentVolumeClaim】，完成创建。

创建工作负载

1. 在“集群管理”页面单击使用 CFS 的集群 ID，进入集群详情页。
2. 在左侧导航栏中选择【工作负载】>【Deployment】，单击【新建】进入 “新建Workload” 页面。
3. 根据实际需求，数据卷选择【使用已有PVC】，并选择上述已创建的 PVC。
4. 挂载到容器的指定路径后，单击【创建Workload】完成创建。

版权所有：TCloudFinanceZone

CBS 说明
CBS 简介

操作场景

支持 TKE 集群通过控制台快捷选择存储类型，并创建对应块存储云硬盘类型的 PV 和 PVC。本文提供 CBS-CSI 组件功能特性等说明并介绍几种常见示例用法。

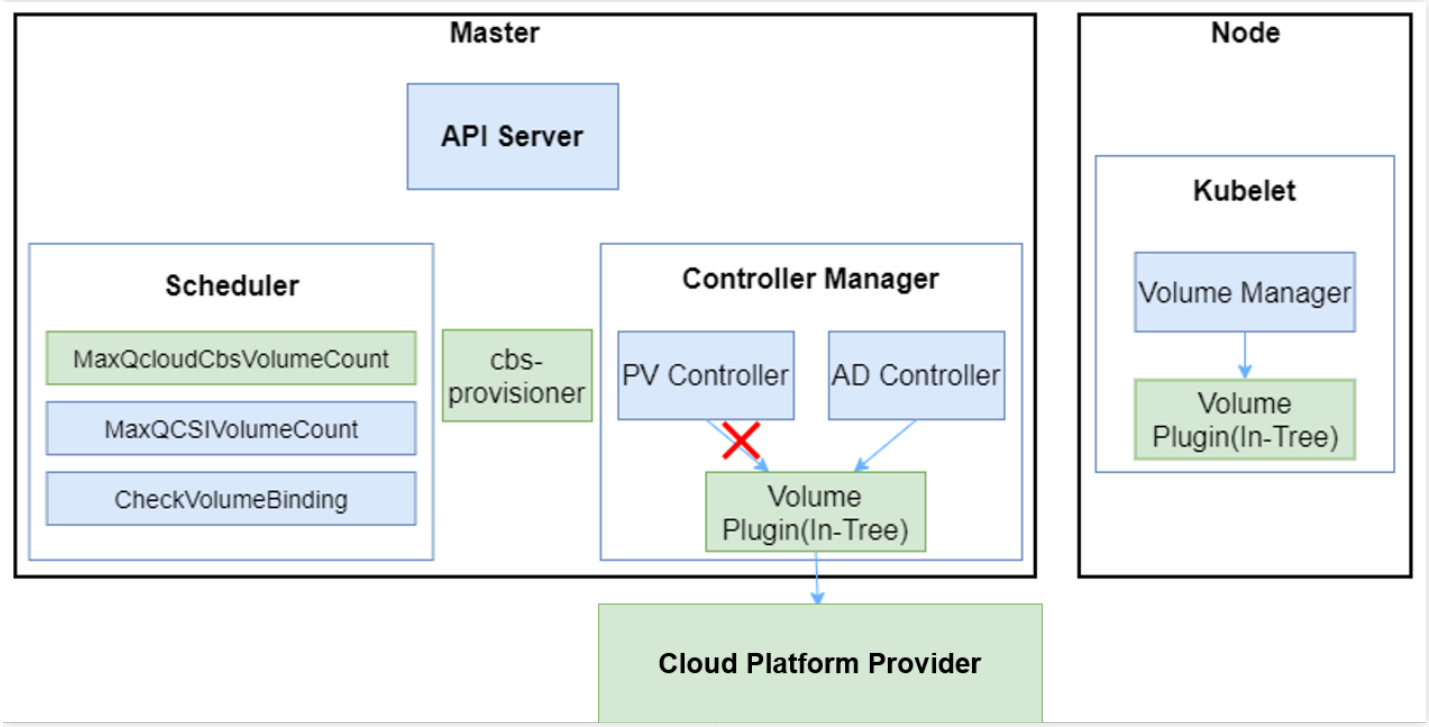
功能特性

功能	说明
静态数据卷	支持手动创建 Volume、PV 对象及 PVC 对象
动态数据卷	支持通过 StorageClass 配置、创建和删除 Volume 及 PV 对象
存储拓扑感知	云硬盘不支持跨可用区挂载，在多可用区集群中，CBS-CSI 组件将先调度 Pod，后调度 Node 的 zone 创建 Volume
调度器感知节点 maxAttachLimit	单个云服务器上默认最多挂载20块云硬盘，调度器调度 Pod 时将过滤超过最大可挂载云硬盘数量的节点
卷在线扩容	支持通过修改 PVC 容量字段，实现在线扩容（仅支持云硬盘类型）
卷快照和恢复	支持通过快照创建数据卷

组件说明

CBS-CSI 组件在集群内部署后，包含以下组件：

- DaemonSet：每个 Node 提供一个 DaemonSet，简称为 NodePlugin。由 CBS-CSI Driver 和 node-driver-registrar 两个容器组成，负责向节点注册 Driver，并提供挂载能力。
- StatefulSet 和 Deployment：简称为 Controller。由 Driver 和多个 Sidecar（external-provisioner、external-attacher、external-resizer、external-snapshotter、snapshot-controller）一起构成，提供创删卷、attach、detach、扩容、快照等能力。
  示例图如下：



限制条件

- 使用 CBS-CSI 组件后，才可在 TKE 集群中为云硬盘在线扩容和创建快照。
- 已经使用 QcloudCbs（In-Tree 插件）的 TKE 集群，可以继续正常使用。（后续将通过 Volume Migration 统一到 CBS CSI）

使用示例

- 通过 CBS-CSI 避免云硬盘跨可用区挂载
- 在线扩容云硬盘
- 创建快照和使用快照来恢复卷

通过CBS避免云硬盘跨可用区挂载

操作场景

云硬盘不支持跨可用区挂载到节点，在跨可用区的集群环境中，推荐通过 CBS 拓扑感知特性来避免跨可用区挂载问题。

实现原理

拓扑感知调度需要多个 Kubernetes 组件配合完成，包括 Scheduler、PV controller、external-provisioner。具体流程如下：

1. PV controller 观察 PVC 对象，检查 Storageclass 的 VolumeBindingMode 是否为 WaitForFirstConsumer，如是，则不会立即处理该 PVC 的创建事件，等待 Scheduler 处理。
2. Scheduler 调度 Pod 后，会将 nodeName 以 annotation 的方式加入到 PVC 对象上 volume.kubernetes.io/selected-node: 10.0.0.72 。
3. PV controller 获取到 PVC 对象的更新事件后，将开始处理 annotation（ volume.kubernetes.io/selected-node ），根据 nodeName 获取 Node 对象，传入到 external-provisioner 中。
4. external-provisioner 根据传过来的 Node 对象的 label 获取可用区（ failure-domain.beta.kubernetes.io/zone ）后在对应可用区创建 PV，达到和 Pod 相同可用区的效果，避免云硬盘和 Node 在不同可用区而无法挂载问题。

前提条件

- 已将 CBS 组件更新为最新版本。

操作步骤

使用以下 YAML，在 Storageclass 中设置 volumeBindingMode 为 WaitForFirstConsumer。示例如下：

```
kind: StorageClass
metadata:
  name: cbs-topo
parameters:
  type: cbs
provisioner: com.abcd.cloud.csi.cbs
reclaimPolicy: Delete
```

volumeBindingMode: WaitForFirstConsumer

> 说明
>
> CBS-CSI 和 In-Tree 组件均支持该操作。

在线扩容云硬盘

操作场景

TKE 支持在线扩容 PV、对应的云硬盘及文件系统，即不需要重启 Pod 即可完成扩容。为确保文件系统的稳定性，建议在云硬盘文件系统处于未挂载状态时进行操作。

前提条件

- 已将 CBS 更新为最新版本。
- （可选）为避免扩容失败导致数据丢失，可以在扩容前使用快照备份数据。

操作步骤

创建允许扩容的 StorageClass

使用以下 YAML 创建允许扩容的 StorageClass，在 Storageclass 中设置 allowVolumeExpansion 为 true 。示例如下：

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cbs-csi-expand
parameters:
  diskType: CLOUD_PREMIUM
provisioner: com.abcd.cloud.csi.cbs
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

在线扩容

提供以下两种扩容方式：

扩容方式	说明
重启 Pod 的情况下在线扩容	待扩容的云硬盘文件系统未被挂载，能够避免扩容出错以及方式2存在的问题。推荐使用该方式进行扩容。

扩容方式	说明
不重启 Pod 的情况下在线扩容	在节点上挂载着待扩容的云硬盘文件系统，如果存在 I/O 进程，将可能出现文件系统扩容错误。

1. 执行以下命令，确认扩容前 PV 和文件系统状态。示例如下，PV 和文件系统大小均为30G：

```
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/vdd       30832548 44992  30771172   1% /usr/share/nginx/html
$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS   REASON   AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   30Gi       RWO            Delete           Bound    defaul
t/www1-ivantestweb-0   cbs-csi                 20h
```

2. 执行以下命令，为 PV 对象打上一个非法 zone 标签，旨在下一步重启 Pod 后，使 Pod 无法调度到某个节点上。示例如下：

```
$ kubectl label pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c failure-domain.beta.kubernetes.i
o/zone=nozone
```

3. 执行以下命令重启 Pod，重启后由于 Pod 对应的 PV 的标签表明的是非法 zone，Pod 将处于 Pending 状态。示例如下：

```
$ kubectl delete pod ivantestweb-0
$ kubectl get pod ivantestweb-0
NAME           READY   STATUS    RESTARTS   AGE
ivantestweb-0   0/1     Pending   0          25s
$ kubectl describe pod ivantestweb-0
Events:
Type     Reason            Age               From               Message
----     ------            ----              ----               -------
Warning  FailedScheduling  40s (x3 over 2m3s)  default-scheduler  0/1 nodes are available: 1 nod
e(s) had no available volume zone.
```

4. 执行以下命令，修改 PVC 对象中的容量，将容量扩容至40G。示例如下：

```
kubectl patch pvc www1-ivantestweb-0 -p '{"spec":{"resources":{"requests":{"storage":"40Gi"}}}}'
```

> 注意

> 扩容后的PVC对象容量的大小必须为10的倍数。

5. 执行以下命令，去除 PV 对象之前打上的标签， 标签去除之后 Pod 即可调度成功。示例如下：

```
$ kubectl label pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c failure-domain.beta.kubernetes.io/zone-
persistentvolume/pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c labeled
```

6. 执行以下命令，可以查看到 Pod 状态为 Running、对应的 PV 和文件系统扩容成功，从30G扩容到40G。示例如下：

```
$ kubectl get pod ivantestweb-0
NAME           READY   STATUS    RESTARTS   AGE
ivantestweb-0   1/1     Running   0          17m
$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS   REASON   AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   40Gi       RWO            Delete           Bound    defaul
t/www1-ivantestweb-0   cbs-csi                 20h
$ kubectl get pvc www1-ivantestweb-0
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAG
ECLASS   AGE
www1-ivantestweb-0   Bound    pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   40Gi       RWO
cbs-csi        20h
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/vdd        41153760 49032  41088344   1% /usr/share/nginx/html
```

创建快照和使用快照来恢复卷

操作场景

如需为 PVC 数据盘创建快照来备份数据，或者将备份的快照数据恢复到新的 PVC 中，可以通过 CBS-CSI 插件来实现，本文将介绍如何利用 CBS-CSI 插件实现 PVC 的数据备份与恢复。

前提条件

- 已安装最新版的 CBS 组件。

操作步骤

备份PVC

创建 VolumeSnapshotClass

1. 使用以下 YAML，创建 VolumeSnapshotClass 对象。示例如下：

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: cbs-snapclass
driver: com.abcd.cloud.csi.cbs
deletionPolicy: Delete
```

2. 执行以下命令查看 VolumeSnapshotClass 是否创建成功。示例如下：

```
$ kubectl get volumesnapshotclass
NAME            DRIVER                    DELETIONPOLICY   AGE
cbs-snapclass   com.abcd.cloud.csi.cbs    Delete           17m
```

创建 PVC 快照 VolumeSnapshot

1. 本文以 new-snapshot-demo 快照名为例，使用以下 YAML 创建 VolumeSnapshot 对象。示例如下：

```
apiVersion: snapshot.storage.k8s.io/v1beta1
```

```
kind: VolumeSnapshot
metadata:
  name: new-snapshot-demo
spec:
  volumeSnapshotClassName: cbs-snapclass
  source:
  persistentVolumeClaimName: csi-pvc
```

2. 执行以下命令，查看 Volumesnapshot 和 Volumesnapshotcontent 对象是否创建成功，若 READYTOUSE 为 true，则创建成功。示例如下：

```
$ kubectl get volumesnapshot
NAME                READYTOUSE   SOURCEPVC          SOURCESNAPSHOTCONTENT   RESTORESIZE
SNAPSHOTCLASS   SNAPSHOTCONTENT                                   CREATIONTIME   AGE
new-snapshot-demo   true         www1-ivantestweb-0                    10Gi          cbs-snapclass
snapcontent-ea11a797-d438-4410-ae21-41d9147fe610   22m            22m
```

```
$ kubectl get volumesnapshotcontent
NAME                                       READYTOUSE   RESTORESIZE   DELETIONPOLICY   DRIVER
VOLUMESNAPSHOTCLASS   VOLUMESNAPSHOT      AGE
snapcontent-ea11a797-d438-4410-ae21-41d9147fe610   true         10737418240   Delete           c
om.abcd.cloud.csi.cbs   cbs-snapclass         new-snapshot-demo   22m
```

3. 执行以下命令，可以获取 Volumesnapshotcontent 对象的快照 ID，字段是 status.snapshotHandle（如下为 snap-e406fc9m），可以根据快照 ID 在 云服务控制台 > 快照列表 确认快照是否存在。示例如下：

```
$ kubectl get volumesnapshotcontent snapcontent-ea11a797-d438-4410-ae21-41d9147fe610 -o
yaml
```

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotContent
metadata:
  creationTimestamp: "2020-11-04T08:58:39Z"
  finalizers:
  - snapshot.storage.kubernetes.io/volumesnapshotcontent-bound-protection
  name: snapcontent-ea11a797-d438-4410-ae21-41d9147fe610
  resourceVersion: "471437790"
  selfLink: /apis/snapshot.storage.k8s.io/v1beta1/volumesnapshotcontents/snapcontent-ea11a79
7-d438-4410-ae21-41d9147fe610
  uid: 70d0390b-79b8-4276-aa79-a32e3bdef3d6
spec:
  deletionPolicy: Delete
  driver: com.abcd.cloud.csi.cbs
```

```
  source:
    volumeHandle: disk-7z32tin5
  volumeSnapshotClassName: cbs-snapclass
  volumeSnapshotRef:
    apiVersion: snapshot.storage.k8s.io/v1beta1
    kind: VolumeSnapshot
    name: new-snapshot-demo
    namespace: default
    resourceVersion: "471418661"
    uid: ea11a797-d438-4410-ae21-41d9147fe610
status:
  creationTime: 1604480319000000000
  readyToUse: true
  restoreSize: 10737418240
  snapshotHandle: snap-e406fc9m
```

从快照恢复数据到新 pvc

1. 本文以上述步骤中创建的 VolumeSnapshot 的对象名为 new-snapshot-demo 为例，使用以下 YAML 从快照恢复卷。示例如下：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: restore-test
spec:
  storageClassName: cbs-csi
  dataSource:
  name: new-snapshot-demo
  kind: VolumeSnapshot
  apiGroup: snapshot.storage.k8s.io
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

2. 执行以下命令，查看恢复的 PVC 已成功创建，从 PV 中可以查看到对应的 diskid（如下为 disk-gahz1kw1）。示例如下：

```
$ kubectl get pvc restore-test
NAME           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECL
ASS   AGE
```

```
restore-test   Bound    pvc-80b98084-29a3-4a38-a96c-2f284042cf4f   10Gi       RWO            cbs-csi        97s

$ kubectl get pv pvc-80b98084-29a3-4a38-a96c-2f284042cf4f -oyaml

apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: com.abcd.cloud.csi.cbs
  creationTimestamp: "2020-11-04T12:08:25Z"
  finalizers:
  - kubernetes.io/pv-protection
  name: pvc-80b98084-29a3-4a38-a96c-2f284042cf4f
  resourceVersion: "474676883"
  selfLink: /api/v1/persistentvolumes/pvc-80b98084-29a3-4a38-a96c-2f284042cf4f
  uid: 5321df93-5f21-4895-bafc-71538d50293a
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: restore-test
    namespace: default
    resourceVersion: "474675088"
    uid: 80b98084-29a3-4a38-a96c-2f284042cf4f
  csi:
    driver: com.abcd.cloud.csi.cbs
    fsType: ext4
    volumeAttributes:
      diskType: CLOUD_PREMIUM
      storage.kubernetes.io/csiProvisionerIdentity: 1604478835151-8081-com.abcd.cloud.csi.cbs
    volumeHandle: disk-gahz1kw1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: topology.com.abcd.cloud.csi.cbs/zone
          operator: In
          values:
          - ap-cityA-2
  persistentVolumeReclaimPolicy: Delete
  storageClassName: cbs-csi
```

```
  volumeMode: Filesystem
status:
  phase: Bound
```

> 说明
>
> 如果 StorageClass 使用了拓扑感知（先调度 Pod 再创建 PV），即指定 volumeBindingMode: WaitForFirstConsumer ，则需要先部署 Pod（需挂载 PVC）才会触发创建 PV（从快照创建新的 CBS 并与 PV 绑定）。

P2P 说明

简介

组件介绍

P2P Addon 是容器镜像服务 TCR 推出的基于 P2P 技术的容器镜像加速分发插件，可应用于大规模容器服务 TKE 集群快速拉取 GB 级容器镜像，支持上千节点的并发拉取。
该组件由 p2p-agent 、 p2p-proxy 和 p2p-tracker 组成：

- p2p-agent：部署在集群中每个节点上，代理每个节点的镜像拉取请求，并转发至 P2P 网络的各个 peer（node 节点）间。
- p2p-proxy：部署在集群部分节点上，作为原始种子连接被加速的镜像仓库。proxy 节点既需要做种，也需要从目标镜像仓库中拉取原始数据。
- p2p-tracker：部署在集群部分节点上，开源 bittorrent 协议的 tracker 服务。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	请求资源	所属 Namespace
p2p-agent	DaemonSet	每个节点0.2核 CPU，0.2G内存	kube-system
p2p-proxy	Deployment	每个节点0.5核 CPU，0.5G内存	kube-system
p2p-tracker	Deployment	每个节点0.5核 CPU，0.5G内存	kube-system
p2p-proxy	Service	-	kube-system
p2p-tracker	Service	-	kube-system
agent	Configmap	-	kube-system
proxy	Configmap	-	kube-system
tracker	Configmap	-	kube-system

使用场景

应用于大规模 TKE 集群快速拉取 GB 级容器镜像，支持上千节点的并发拉取，推荐如下使用场景：

- 集群内具备节点500 - 1000台，使用本地盘存储拉取的容器镜像。此场景下，集群内节点最高可支持100MB/s的并发拉取速度。
- 集群内具备节点500 - 1000台，使用 CBS 云盘存储拉取的容器镜像，且集群所在地域为城市A、城市B、城市C等主要地域。此场景下，集群内节点最高可支持20MB/s的并发拉取速度。

限制条件

- 在大规模集群内启用 P2P Addon 拉取容器镜像时，将对节点数据盘造成较高读写压力，可能影响集群内已有业务。若集群内节点使用 CBS 云盘存储拉取的容器镜像，请按照集群所在地域选择合适的下载限速或联系您的售后/架构师，避免因镜像拉取时云盘读写负载过高造成集群内现网业务中断现象，甚至影响该地域内其他用户的正常使用。
- 开启 P2P 插件需要预留一定的资源，P2P 组件在镜像加速拉取的过程中会占用节点的 CPU 和内存资源，加速结束后不再占用资源。其中：
- Proxy 的 limit 限制为：4核 CPU 和4G 内存。
- Agent 的 limit 限制为：4核 CPU 和2G 内存。
- Tracker 的 limit 限制为：2核 CPU 和4G 内存。
- 需要根据集群的节点规模，估算启动的 Proxy 个数。Proxy 运行节点的最低配置为4C8G，内网带宽1.5GB/s，单个 Proxy 服务可支撑200个集群节点。
- 需要主动为 Proxy 和 Tracker 组件选择部署节点，使用方式为手动为节点打 K8S 标签，详情请参见 使用方法。Proxy 和 Agent 所在的节点需要能够访问的仓库源站。
- Agent 组件将会占用节点的5004端口，以及 P2P 专用通信端口6881（Agent）和6882（Proxy）。Agent、Proxy 组件会分别创建本地工作目录 /p2p_agent_data 和 /p2p_proxy_data 用于缓存容器镜像，请提前确认节点已预留足够的存储空间。

使用方法

1. 选取合适的节点部署运行 Proxy 组件。
   可通过 kubectl label nodes XXXX proxy=p2p-proxy 命令标记节点，插件安装时将自动在这些节点中部署该组件。安装后如果需要调整 Proxy 组件的个数，可在指定节点上添加或者删除该 label 后，修改集群中 kube-system 命名空间下 p2p-proxy 工作负载的副本个数。
2. 选取合适的节点部署运行 Tracker 组件。
   可通过 kubectl label nodes XXXX tracker=p2p-tracker 命令标记节点，插件安装时将自动在这些节点中部署该组件。安装后如果需要调整 Tracker 的个数，可在指定节点上添加或者删除该 label 后，修改集群中 kube-system 命名空间下 p2p-tracker 工作负载的副本个数。
3. 节点安全组需要添加的配置为：入站规则放通 TCP 和 UDP 的30000 - 32768 端口、以及 VPC 内 IP 全放通。出站规则放通全部（TKE 集群 work 节点默认安全组已满足要求）。

4. 选择指定集群 开启 P2P Addon 插件。填写需要加速的镜像仓库域名，节点拉取限速、Proxy 个数，Tracker 个数。安装后如果需要重新调整下载的最高速度，可修改 p2p-agent configmap 中的 downloadRate 和 uploadRate。
5. 在业务命名空间内创建拉取镜像所需的 dockercfg，其中仓库域名为 localhost:5004，用户名及密码即为目标镜像仓库的原有访问凭证。
6. 修改业务 YAML，将需要加速的镜像仓库域名地址修改为 localhost:5004，如 localhost:5004/p2p-test/test:1.0，并使用新建的 dockercfg 作为 ImagePullSecret。
7. 使用业务 YAML 部署更新工作负载，并实时观察镜像拉取速度及节点磁盘读写负载，及时调整节点的下载限速以达到最好加速效果。

操作步骤

1. 登录容器服务控制台，选择左侧导航栏中的【集群】。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的【组件管理】，进入“组件列表”页面。
4. 在“组件列表”页面中选择【新建】，并在“新建组件”页面中勾选 P2P。
5. 选择“参数配置”，在弹出的“P2P组件参数设置”窗口中，填写需要加速的镜像仓库域名、节点拉取限速、Proxy 个数及 Tracker 个数。

GPU-Manager 说明

简介

组件介绍

GPU Manager 提供一个 All-in-One 的 GPU 管理器，基于 Kubernetes DevicePlugin 插件系统实现，该管理器提供了分配并共享 GPU、GPU 指标查询、容器运行前的 GPU 相关设备准备等功能，支持用户在 Kubernetes 集群中使用 GPU 设备。

组件功能

- 拓扑分配：提供基于 GPU 拓扑分配功能，当用户分配超过1张 GPU 卡的应用，可以选择拓扑连接最快的方式分配 GPU 设备。
- GPU 共享：允许用户提交小于1张卡资源的任务，并提供 QoS 保证。
- 应用 GPU 指标的查询：用户可以访问主机端口（默认为 5678）的 /metric 路径，可以为 Prometheus 提供 GPU 指标的收集功能，访问 /usage 路径可以进行可读性的容器状况查询。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	建议预留资源	所属 Namespaces
gpu-manager-daemonset	DaemonSet	每节点1核 CPU, 1Gi内存	kube-system
gpu-quota-admission	Deployment	每节点1核 CPU, 1Gi内存	kube-system

使用场景

在 Kubernetes 集群中运行 GPU 应用时，可以解决 AI 训练等场景中申请独立卡造成资源浪费的情况，让计算资源得到充分利用。

限制条件

- 该组件基于 Kubernetes DevicePlugin 实现，可直接在 Kubernetes 1.10 以上版本的集群使用。
- 每张 GPU 卡一共有100个单位的资源，仅支持0 - 1的小数卡，以及1的倍数的整数卡设置。显存资源是以256MiB为最小的一个单位的分配显存。
- 使用 GPU-Manager 要求集群内包含 GPU 机型节点。

使用方法

组件安装

1. 登录容器服务控制台，在左侧导航栏中选择【集群】。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的【组件管理】，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择【新建】，并在“新建组件”页面中勾选 GpuManager。
5. 单击【完成】即可创建组件。

创建细粒度的 GPU 工作负载

当 GpuManager 组件成功安装后，您可通过以下两种方式创建细粒度的 GPU 工作负载。

方式一：通过 TKE 控制台创建

1. 登录容器服务控制台，选择左侧导航栏中的【集群】。
2. 选择需要创建 GPU 应用的集群，进入工作负载管理页，并单击【新建】。
3. 在“新建Workload”页面根据实际需求进行配置，可在“GPU资源”配置细粒度的 GPU 工作负载。

方式二：通过 yaml 创建

说明：在提交时通过 yaml 为容器设置 GPU 的使用资源，核资源需要在 resource 上填写 finance.cloud.tencent.com /vcuda-core ，显存资源需要在 resource 上填写finance.cloud.tencent.com /vcuda-memory 。
下面给出 yaml 示例：

- 使用1张卡的 P4 设备：

```
apiVersion: v1
kind: Pod
...
spec:
containers:

name:gpu
resources:finance.cloud.tencent.com/vcuda-core:100
```

- 使用0.3张卡，5GiB 显存的应用：

```
apiVersion: v1
kind: Pod
...
spec:
```

containers:

name:gpu
resources:finance.cloud.tencent.com/vcuda-core:30 finance.cloud.tencent.com/vcuda-memory:20

HPC 说明

简介

组件介绍

HPC（HorizontalPodCronscaler）是一种可以对 K8s workload 副本数进行定时修改的组件，配合 HPC CRD 使用，最小支持秒级的定时任务。

组件功能

- 支持设置“实例范围”（关联对象为 HPA）或“目标实例数量”（关联对象为 deployment 和 statefulset）。
- 支持开关“例外时间”。例外时间的最小配置粒度是日期，支持设置多条。
- 支持设置定时任务是否只执行一次。

部署在集群内的 Kubernetes 对象

在集群内部署 HPC，将在集群内部署以下 Kubernetes 对象：

Kubernetes 对象名称	类型	默认占用资源	所属Namespaces
horizontalpodcronscalers.autoscaling.finance.cloud.tencent.com	CustomResourceDefinition	-	-
hpc-leader-election-role	Role	-	kube-system
hpc-leader-election-rolebinding	RoleBinding	-	kube-system
hpc-manager-role	ClusterRole	-	-
hpc-manager-rolebinding	ClusterRoleBinding	-	-
cronhpa-controller-manager-metrics-service	Service	-	kube-system
hpc-manager	ServiceAccount	-	kube-system
tke-hpc-controller	Deployment	100mCPU/pod、100Mi/pod	kube-system

限制条件

环境要求

您在创建集群时选择1.12.4以上版本集群，无需修改任何参数，开箱可用。

- 仅支持1.12版本以上的 kubernetes。
- 需设置 kube-apiserver 的启动参数： --feature-gates=CustomResourceSubresources=true

节点要求

- HPC 组件默认挂载主机的时区将作为定时任务的参考时间，因此要求节点存在 /etc/localtime 文件。
- HPC 默认安装2个 HPC Pod 在不同节点，因此节点数推荐为2个及以上。

被控资源要求

在创建 HPC 资源时，被控制的 workload（K8s 资源）需要存在于集群中。

操作步骤

安装 HPC

1. 登录 容器服务控制台，在左侧导航栏中选择【集群】。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的【组件管理】，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择【新建】，并在“新建组件”页面中勾选 HPC。
5. 单击【完成】即可创建组件。

创建并使用 HPC 工作负载示例

创建关联 Deployment 的定时任务资源

示例如下：

```
apiVersion: autoscaling.finance.cloud.tencent.com/v1
kind: HorizontalPodCronscaler
metadata:
  name: hpc-deployment
  namespace: default
spec:
  scaleTarget:
```

```
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
    namespace: default
  crons:
  - name: "scale-down"
    excludeDates:
      - "* * * 15 11 *"
      - "* * * * * 5"
    schedule: "30 */1 * * * *"
    targetSize: 1
  - name: "scale-up"
    excludeDates:
      - "* * * 15 11 *"
      - "* * * * * 5"
    schedule: "0 */1 * * * *"
    targetSize: 3
```

创建关联 StatefulSet 的定时任务资源

示例如下：

```
apiVersion: autoscaling.finance.cloud.tencent.com/v1
kind: HorizontalPodCronscaler
metadata:
  name: hpc-statefulset
  namespace: default
spec:
  scaleTarget:
    apiVersion: apps/v1
    kind: Statefulset
    name: nginx-statefulset
    namespace: default
  crons:
  - name: "scale-down"
    excludeDates:
      - "* * * 15 11 *"
    schedule: "0 */2 * * * *"
    targetSize: 1
  - name: "scale-up"
    excludeDates:
      - "* * * 15 11 *"
    schedule: "30 */2 * * * *"
    targetSize: 4
```

创建关联 HPA 的定时任务资源

示例如下：

```
apiVersion: autoscaling.finance.cloud.tencent.com/v1
kind: HorizontalPodCronscaler
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: hpc-hpa
spec:
  scaleTarget:
    apiVersion: autoscaling/v1
    kind: HorizontalPodAutoscaler
    name:  nginx-hpa
    namespace: default
  crons:
  - name: "scale-up"
    schedule: "30 */1 * * * *"
    minSize: 2
    maxSize: 6
  - name: "scale-down"
    schedule: "0 */1 * * * *"
    minSize: 1
    maxSize: 5
```

定时时间设置参考

字段名称	是否必选	允许值范围	允许的特殊字符
Seconds	是	0 - 59	* / , -
Minutes	是	0 - 59	* / , -
Hours	是	0 - 23	* / , -
Day of month	是	1 - 31	* / , - ?
Month	是	1 - 12 或 JAN - DEC	* / , -
Day of week	是	0 - 6 或 SUN - SAT	* / , - ?

Network Policy 说明

简介

组件介绍

Network Policy 是 Kubernetes 提供的一种资源，用于定义基于 Pod 的网络隔离策略。它描述了一组 Pod 是否可以与其他组 Pod，以及其他 Network Entities 进行通信。本组件提供了针对该资源的 Controller 实现。如果您希望在 IP 地址或端口层面（OSI 第3层或第4层）控制特定应用的网络流量，则可考虑使用本组件。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	请求资源	所属 Namespace
networkpolicy	DaemonSet	每个实例CPU:250m，Memory:250Mi	kube-system
networkpolicy	ClusterRole	-	kube-system
networkpolicy	ClusterRoleBinding	-	kube-system
networkpolicy	ServiceAccount	-	kube-system

操作步骤

1. 登录容器服务控制台，在左侧导航栏中选择【集群】。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的【组件管理】，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择【新建】，并在“新建组件”页面中勾选 NetworkPolicy。
5. 单击【完成】即可创建组件。

Nginx-ingress 说明

简介

组件介绍

Nginx 可以用作反向代理、负载平衡器和 HTTP 缓存。Nginx-ingress 组件是使用 Nginx 作为反向代理和负载平衡器的 Kubernetes 的 Ingress 控制器。您可以部署 Nginx-ingress 组件，在集群中使用 Nginx-ingress。

部署在集群内的 Kubernetes 对象

在集群内部署 Nginx-ingress Add-on，将在集群内部署以下 Kubernetes 对象：

Kubernetes 对象名称	类型	默认占用资源	所属 Namespaces
tke-ingress-nginx-controller-operator	Deployment	0.13核 CPU，128MB 内存	kube-system
ingress-nginx-controller	Deployment/ DaementSet	0.1核 CPU	kube-system
ingress-nginx-controller-hpa	HPA	-	kube-system

在集群内创建 Nginx ingress 实例后会在集群内部部署以下 Kubernetes 对象：

Kubernetes 对象名称	类型	默认占用资源	所属 Namespaces
nginx-ingress	Service	-	自定义设置
nginx-ingress	Configmap	-	自定义设置

存储
Volume管理

简介

数据卷类型

- 使用主机路径：将容器所在宿主机的文件目录挂载到容器的指定路径中（即对应 Kubernetes 的 HostPath）。您可以根据业务需求，不设置源路径（即对应 Kubernetes 的 EmptyDir）。如果不设置源路径，系统将分配主机的临时目录挂载到容器的挂载点。指定源路径的本地硬盘数据卷适用于将数据持久化存储到容器所在宿主机，EmptyDir 适用于容器的临时存储。
- 使用 NFS 盘：只需填写 NFS 路径，您可以使用文件存储 CFS，也可使用自建的文件存储 NFS。使用 NFS 数据卷适用于多读多写的持久化存储，也适用于大数据分析、媒体处理、内容管理等场景。
- 使用已有 PersistentVolumeClaim：使用已有 PersistentVolumeClaim 声明工作负载的存储，自动分配或新建 PersistentVolume 挂载到对应的 Pod 下。主要适用于 StatefulSet 创建的有状态应用。更多详情请参见 PV 和 PVC 管理。
- 使用新的 PersistentVolumeClaim：新建一个 PersistentVolumeClaim 声明工作负载的存储，自动分配或新建 PersistentVolume 挂载到对应的 Pod 下。主要适用于 StatefulSet 创建的有状态应用。更多详情请参见 PV 和 PVC 管理。
- 使用TCloudFinanceZone硬盘：TCloudFinanceZone基于 CBS 扩展的 Kubernetes 的块存储插件。您可以指定一块TCloudFinanceZone的 CBS 云硬盘挂载到容器的某一路径下，当容器迁移时，云硬盘会随之迁移。使用云硬盘数据卷适用于数据的持久化保存，可用于 Mysql 等有状态服务。设置云硬盘数据卷的服务，实例数量最大为 1。
- 使用 ConfigMap：ConfigMap 以文件系统的形式挂载到 Pod 上，支持自定义 ConfigMap 条目挂载到特定的路径。更多详情请参见 ConfigMap 管理。
- 使用 Secret：Secret 以文件系统的形式挂载到 Pod 上，支持自定义 Secret 条目挂载到特定的路径。更多详情请参见 Secret 管理。

数据卷的注意事项

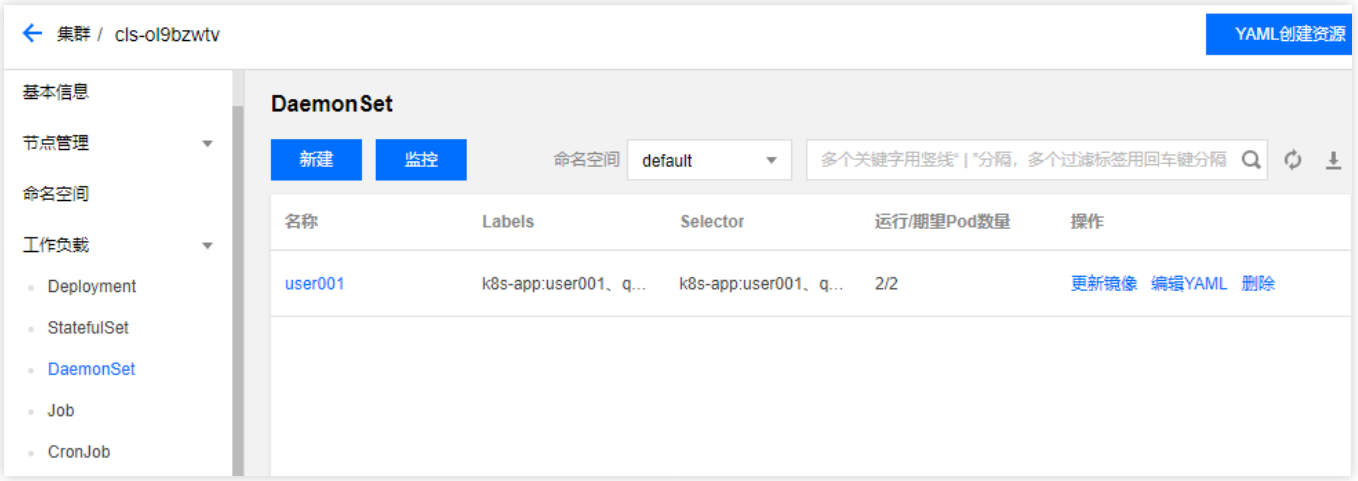
- 创建数据卷后，需设置容器的挂载点。
- 同一个服务下，数据卷的名称和容器设置的挂载点不能重复。
- 本地硬盘数据卷源路径为空时，系统将分配 /var/lib/kubelet/pods/pod_name/volumes/kubernetes.io~empty-dir 临时目录，且使用临时的数据卷生命周期与实例的生命周期保持一致。
- 数据卷挂载未设置权限，默认设置为读写权限。

Volume 控制台操作指引

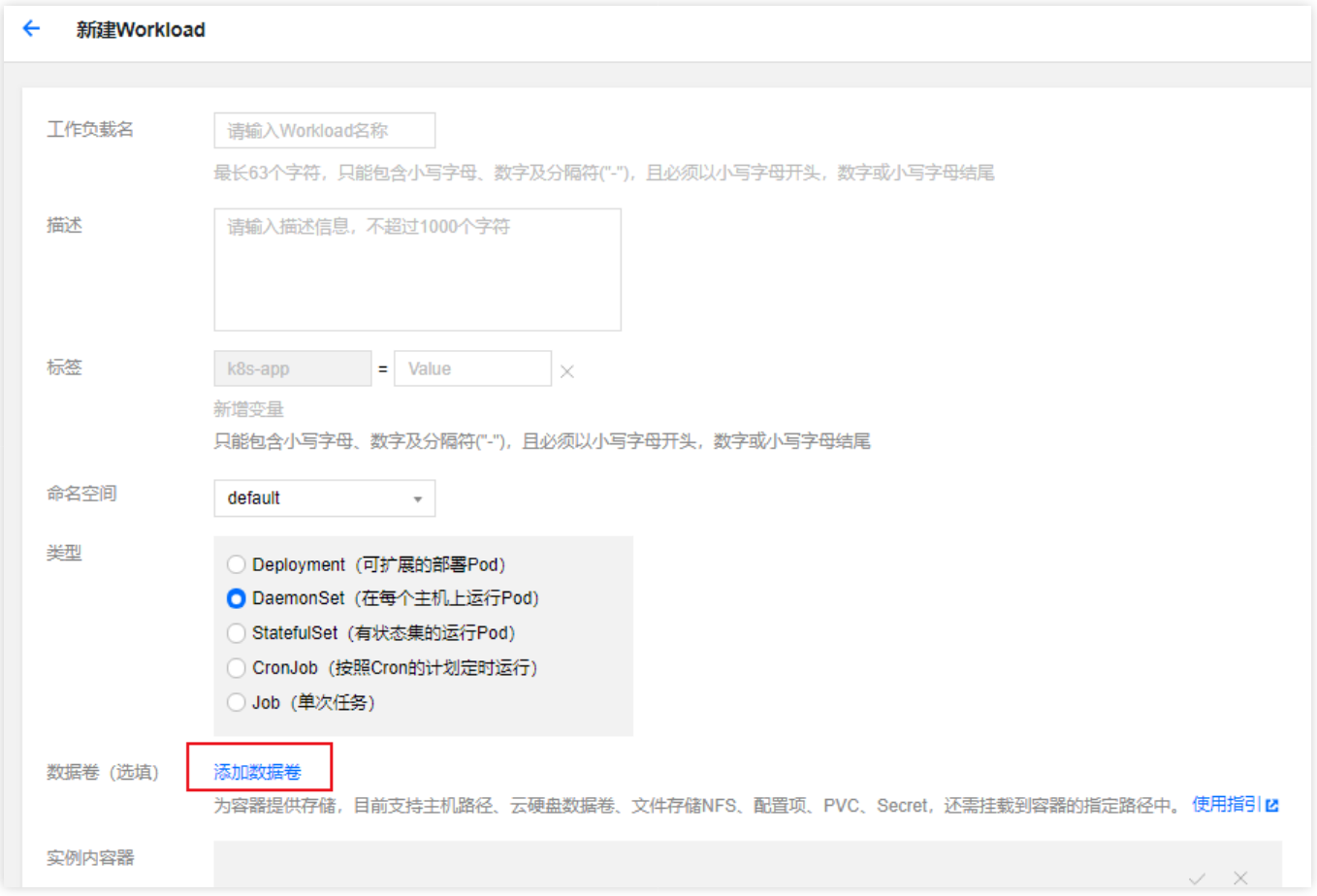
创建工作负载挂载数据卷

1. 登录 TKE 控制台。

2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要部署 Workload 的集群【ID/名称】，进入待部署 Workload 的集群管理页面。
4. 在 “工作负载” 下，任意选择 Workload 类型，进入对应的信息页面。例如，选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。



5. 单击【新建】，进入“新建Workload”页面。
6. 根据页面信息，设置工作负载名、命名空间等信息。并在 “数据卷” 中，单击【添加数据卷】，添加数据卷。



7. 根据实际需求，选择数据卷的存储方式，配置挂载点。
8. 单击【创建Workload】，完成创建。

Kubectl 操作 Volume 指引

仅提供示例文件，您可直接通过 Kubectl 进行创建操作。

Pod 挂载 Volume YAML 示例

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

- spec.volumes：设置数据卷名称、类型、数据卷的参数。
- spec.volumes.emptyDir：设置临时路径。
- spec.volumes.hostPath：设置主机路径。
- spec.volumes.nfs：设置 NFS 盘。
- spec.volumes.persistentVolumeClaim：设置已有 PersistentVolumeClaim
- spec.volumeClaimTemplates：若使用该声明，将根据内容自动创建 PersistentVolumeClaim 和 PersistentVolume。
- spec.containers.volumeMounts：填写数据卷的挂载点。

PV&PVC管理

简介

概述

PersistentVolume（PV）：集群内的存储资源，例如节点是集群的资源。PV 独立于 Pod 的生命周期，根据不同的 StorageClass 类型创建不同类型的 PV。
PersistentVolumeClaim（PVC）：集群内的存储请求。例如，PV 是 Pod 使用节点资源，PVC 则声明使用 PV 资源。当 PV 资源不足时，PVC 也可以动态创建 PV。

注意事项

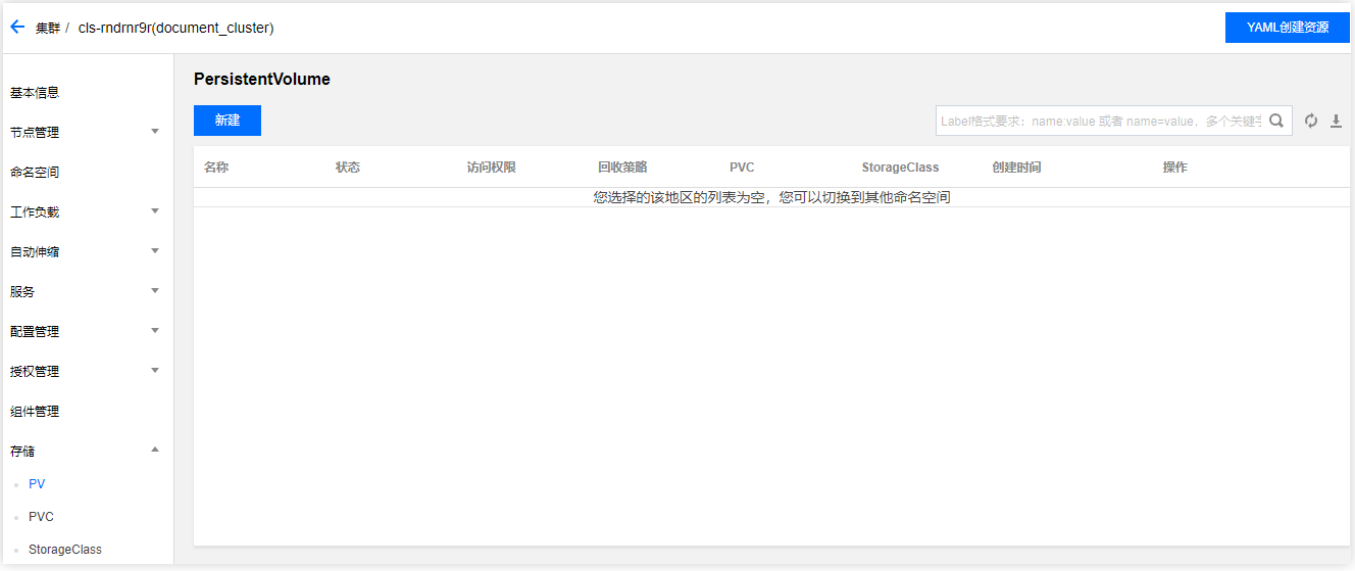
- CBS 盘不支持跨可用区挂载。若挂载 CBS 类型 PV 的 Pod 迁移到其他可用区，将会导致挂载失败。
- TKE 控制台不支持 CBS 盘扩缩容，请自行前往 CBS 控制台执行操作。

PV&PVC 控制台操作指引

静态创建 PV

静态创建 PV 适用于已有存量云盘，并在集群内使用的场景。

1. 登录 TKE 控制台。
2. 在左侧导航栏单击【集群】，进入“集群管理”页面。
3. 单击需要创建 PV 的集群【ID/名称】，进入待创建 PV 的集群管理页面。
4. 选择【存储】 > 【PV】，进入“PersistentVolume”信息页面。



5. 单击【新建】，进入“新建PersistentVolume”页面。



6. 根据实际需求，设置 PV 参数。关键参数信息如下：
   - 名称：自定义。
   - 选择云盘：根据实际需求进行选择。
   - StorageClass：根据实际需求进行选择。
7. 单击【创建PersistentVolume】，完成创建。

创建 PVC

1. 登录 TKE 控制台。
2. 在左侧导航栏单击【集群】，进入“集群管理”页面。
3. 单击需要创建 PVC 的集群【ID/名称】，进入待创建 PVC 的集群管理页面。
4. 选择【存储】 > 【PVC】，进入“PersistentVolumeClaim”信息页面。
5. 单击【新建】，进入【新建PersistentVolumeClaim】页面。

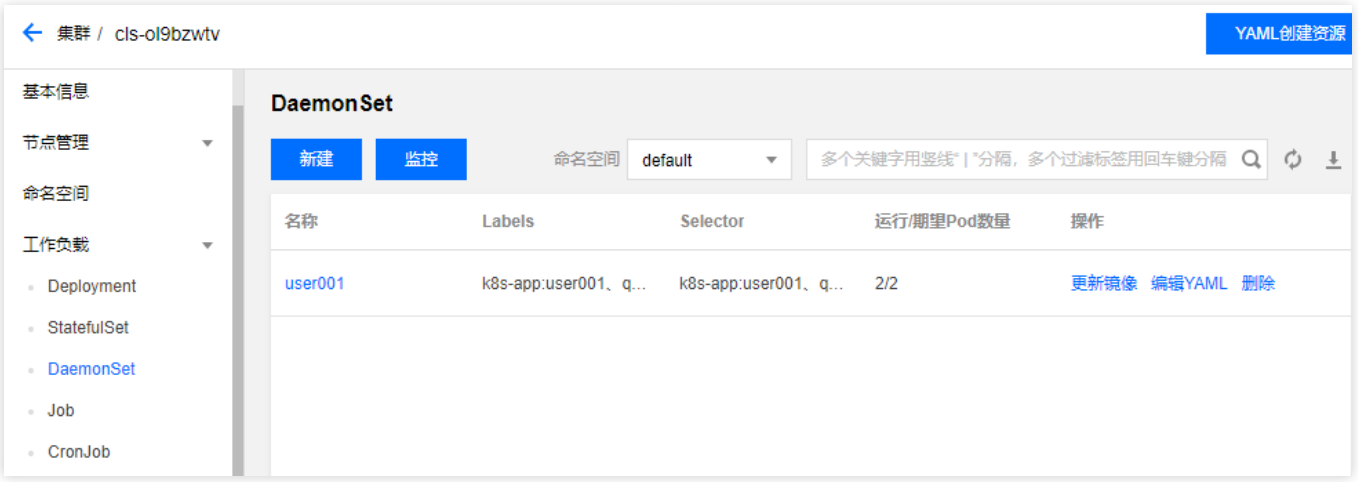


6. 根据实际需求，设置 PVC 参数。关键参数信息如下：
   - 名称：自定义。
   - 命名空间：根据实际需求进行选择命名空间类型。
   - StorageClass：根据实际需求进行选择。
   - 容量：根据实际需求进行设置。
7. 单击【创建PersistentVolumeClaim】，完成创建。

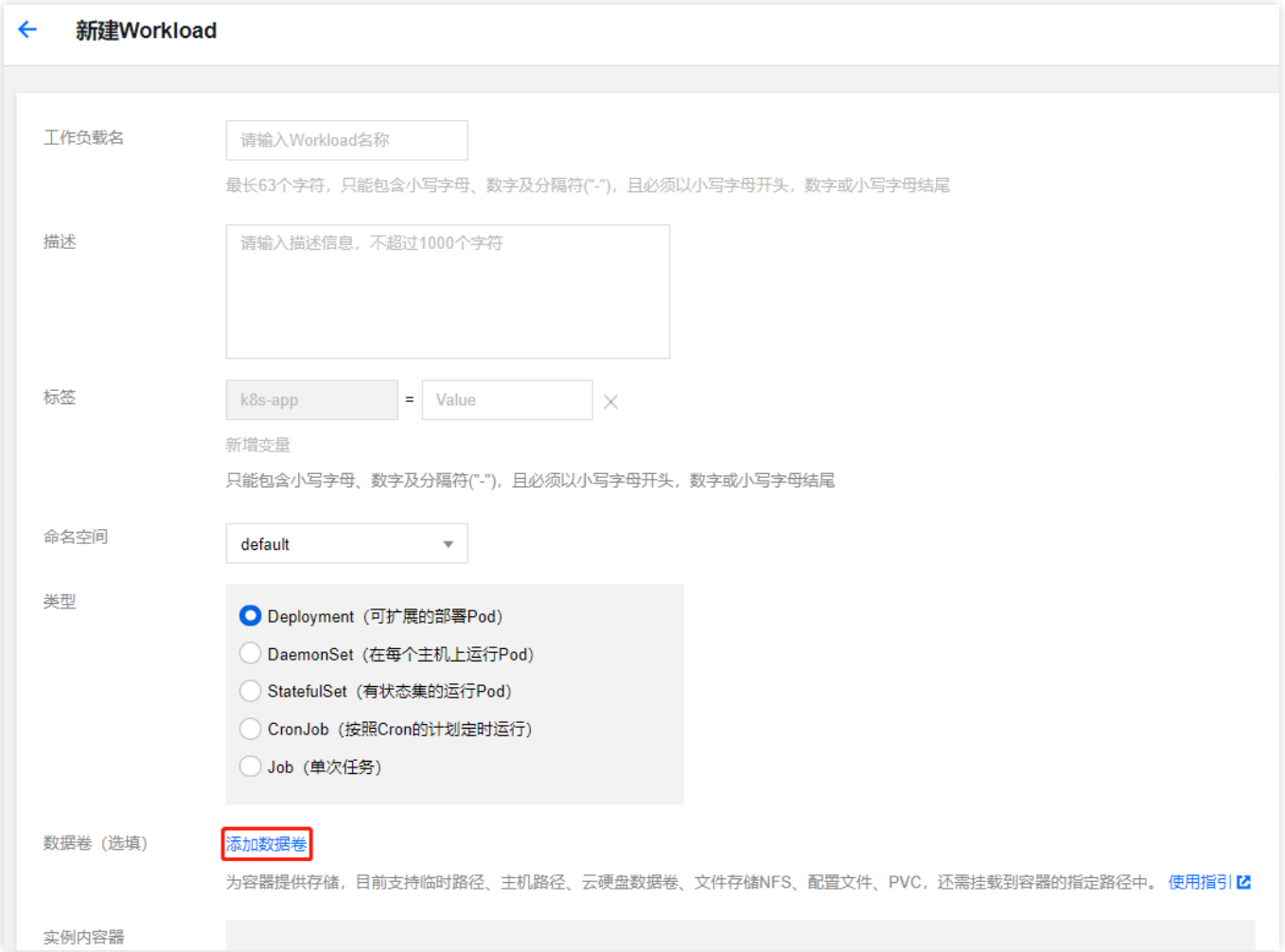
   > 若已有 PV 不足，系统将自动创建新的 PV。

创建 Workload 使用 PVC 数据卷

1. 登录 TKE 控制台。
2. 在左侧导航栏单击【集群】，进入“集群管理”页面。
3. 单击需要部署 Workload 的集群【ID/名称】，进入待部署 Workload 的集群管理页面。
4. 在【工作负载】下，任意选择 Workload 类型，进入对应的信息页面。例如，选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。



5. 单击【新建】，进入“新建Workload”页面。
6. 根据页面信息，设置工作负载名、命名空间等信息。并在【数据卷】中，单击【添加数据卷】，添加数据卷。



7. 选择【使用已有PVC】方式，填写名称，选择已有的 PVC。



8. 单击【创建Workload】，完成创建。

> 使用 PVC 挂载模式，数据卷只能挂载到一台 node 主机上

Kubectl 操作 PV&PVC 指引

当前仅支持 CBS 类型的 PV&PVC。您可通过 StorageClass 管理 指定 PV 绑定的云盘的类型。以下仅提供示例文件，您可直接通过 Kubectl 进行创建操作。

（可选）创建 PV

通过已有 CBS 创建 PV。若未创建 PV，在 创建 PVC 时，系统将自动创建对应的 PV。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nginx-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  qcloudCbs:
      cbsDiskId: disk-xxxxxxx ## 指定已有的CBS id
      fsType: ext4
  storageClassName: cbs
```

创建 PVC

若未 创建 PV，在创建 PVC 时，系统将自动创建对应的 PV。YAML 示例如下：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nginx-pv-claim
spec:
  storageClassName: cbs
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
```

```
    storage: 10Gi
```

- 普通云盘大小必须是10的倍数，最小为10。
- 高效云盘最小为50GB。
- SSD 云硬盘最小为200GB，具体策略请参见 云硬盘文档。

使用 PVC

YAML 示例如下：

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      qcloud-app: nginx-deployment
  template:
    metadata:
      labels:
        qcloud-app: nginx-deployment
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
        volumeMounts:
        - mountPath: "/opt/"
          name: pvc-test
      volumes:
      - name: pvc-test
        persistentVolumeClaim:
          claimName: nginx-pv-claim # 已经创建好的 PVC
```

StorageClass管理

简介

StorageClass 描述存储的类型，集群管理员可以为集群定义不同的存储类别。 TKE 服务默认提供块存储类型的 StorageClass，通过 StorageClass 配合 PersistentVolumeClaim 可以动态创建需要的存储资源。

StorageClass 控制台操作指引

创建 StorageClass

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击需要创建 StorageClass 的集群【ID/名称】，进入待创建 StorageClass 的集群管理页面。
4. 选择【存储】 > 【StorageClass】，进入“StorageClass”信息页面。
5. 单击【新建】，进入“新建StorageClass”页面。
6. 根据实际需求，设置 StorageClass 参数。关键参数信息如下：
    - 名称：自定义。
    - 计费模式：按量计费。
    - 可用区：根据实际需求进行设置，默认为 “随机可用区”。
    - 云盘类型：根据实际需求进行选择。
    - 回收策略：根据实际需求进行选择。
7. 单击【创建StorageClass】，完成创建。

创建 PVC 指定 StorageClass

参照 PV 和 PVC 管理 中的 “创建 PVC”，创建 PVC。并在设置 PVC 参数时，设置 StorageClass 参数。

创建 StatefulSet 挂载自动创建 PersistentVolumeClaim 类型

参照 StatefulSet 管理 中的 “创建 StatefulSet”，创建 StatefulSet。并在设置 StatefulSet 参数时，单击【添加数据卷】，选择 “使用新的 PVC” 方式，设置 PVC。



Kubectl 操作 StorageClass 指引

以下仅提供示例文件，您可直接通过 Kubectl 进行创建操作。

创建 StorageClass

如果不创建 StorageClass， 集群内将默认存在 name 为 CBS 的 StorageClass。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  # annotations:
  #   storageclass.beta.kubernetes.io/is-default-class: "true"
  #   如果有这一条，则会成为 default-class，创建 PVC 时不指定类型则自动使用此类型
  name: cloud-premium
provisioner: finance.cloud.tencent.com/qcloud-cbs ## TKE 集群自带的 provisioner
parameters:
  type: CLOUD_PREMIUM
  # 支持 CLOUD_BASIC,CLOUD_PREMIUM,CLOUD_SSD  如果不识别则当做 CLOUD_BASIC
  # paymode: PREPAID
  # paymode为云盘的计费模式，默认是 POSTPAID（按量计费：支持 Retain 保留和 Delete 删除策略，Retai
n 仅在高于1.8的集群版本生效）
  # aspid:asp-123
  # 支持指定快照策略，创建云盘后自动绑定此快照策略,绑定失败不影响创建
```

支持参数有：

- type：StorageClass 的类型，包括 CLOUD_BASIC、CLOUD_PREMIUM 和 CLOUD_SSD（注意全大写）。
- zone：指定 zone。如果指定，则云盘将创建到此 zone；如果不指定，则拉取所有 node 的 zone 信息，随机选取一个 zone。
- paymode：云盘的计费模式，默认设置为 POSTPAID 模式（即按量计费，支持 Retain 保留和 Delete 删除策略，Retain 仅在高于1.8的集群版本生效）。
- aspid：指定快照 ID，创建云盘后自动绑定此快照策略，绑定失败不影响创建。

创建多实例 StatefulSet

使用云硬盘 CBS 创建多实例 StatefulSet，示例如下所示：

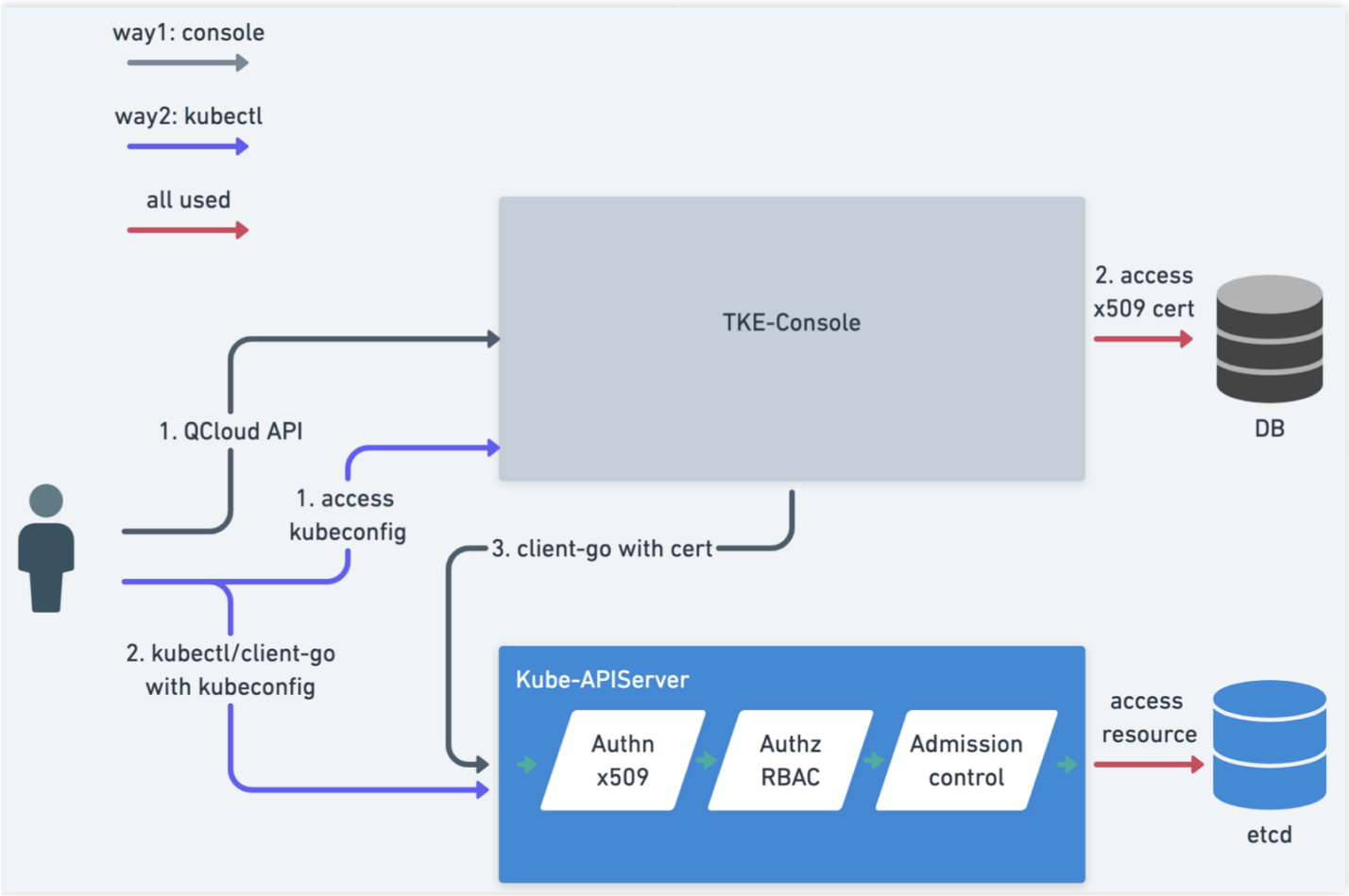
```
apiVersion: apps/v1  #不同k8s版本，api有差异，这里以1.20举例
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 1
```

```
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: ccr.baiping.fsphere.cn/tke4tce/nginx:1.20.1-stable #根据客户自己镜像修改
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:  # 自动创建pvc，进而自动创建pv
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: cloud-premium
      resources:
        requests:
          storage: 10Gi
```

授权管理

概述

TKE 提供了对接 Kubernetes RBAC 的授权模式，便于对子账号进行细粒度的访问权限控制。该授权模式下，可通过容器服务控制台及 kubectl 两种方式进行集群内资源访问。如下图所示：



名词解释

RBAC（Role-Based Access Control）

基于角色的权限控制。通过角色关联用户、角色关联权限的方式间接赋予用户权限。在 Kubernetes 中，RBAC 是通过 rbac.authorization.k8s.io API Group 实现的，即允许集群管理员通过 Kubernetes API 动态配置策略。

Role

用于定义某个命名空间的角色的权限。

ClusterRole

用于定义整个集群的角色的权限。

RoleBinding

将角色中定义的权限赋予一个或者一组用户，针对命名空间执行授权。
ClusterRoleBinding
将角色中定义的权限赋予一个或者一组用户，针对集群范围内的命名空间执行授权。

TKE Kubernetes 对象级别权限控制方案

认证方式

Kubernetes APIServer 支持丰富多样的认证策略，例如 x509 证书、bearer token、basic auth。其中，仅 bearer token 单个认证策略支持指定 known-token csv 文件的 beaer token、serviceaccount token、OIDC token、webhook token server 等多种 token 认证方式。
TKE 分析了实现复杂性及多种场景等因素，选择使用 x509 证书认证方式。其优势如下：

- 用户理解成本低。
- 对于存量集群无需进行复杂变更 。
- 按照 User 及 Group 进行划分，后续扩展性好。
  TKE 基于 x509 证书认证实现了以下功能：
- 每个子账号单独具备客户端证书，用于访问 Kubernetes APIServer。
- 当子账号在控制台访问 Kubernetes 资源时，后台默认使用该子账号的客户端证书去访问用户 Kubernetes APIServer。
- 支持子账号更新独有的客户端证书，防止凭证泄露。
- 支持主账号或使用集群 tke:admin 权限的账号进行查看、更新其他子账号的证书。

授权方式

Kubernetes 包含 RBAC 及 Webhook Server 两种主流授权模式。为给熟悉 Kubernetes 的用户提供一致性体检及需与原生 Kubernetes 结合使用，TKE 选择使用 RBAC 模式。该模式提供了预设 Role 及 ClusterRole，用户只需要在集群内创建相应的 RoleBinding 和 ClusterRoleBinding 即可实现授权变更。其优势如下：

- 亲和有 Kubernetes 基础的用户。
- 复用 Kubernetes RBAC 能力，支持 Namespace 维度、APIGroup 维度及资源维度的多种 Verb 权限控制。
- 支持用户自定义策略。
- 支持管理用户自定义的扩展 API 资源。

TKE Kubernetes 对象级别权限控制功能

通过 TKE 提供的授权管理功能，您可以进行更细粒度的权限控制。例如，仅赋予某个子账号只读权限或仅赋予某个子账号下的某个命名空间读写权限等。可参考以下文档，对子账号进行更细粒度的权限控制：
使用预设身份授权
自定义策略授权

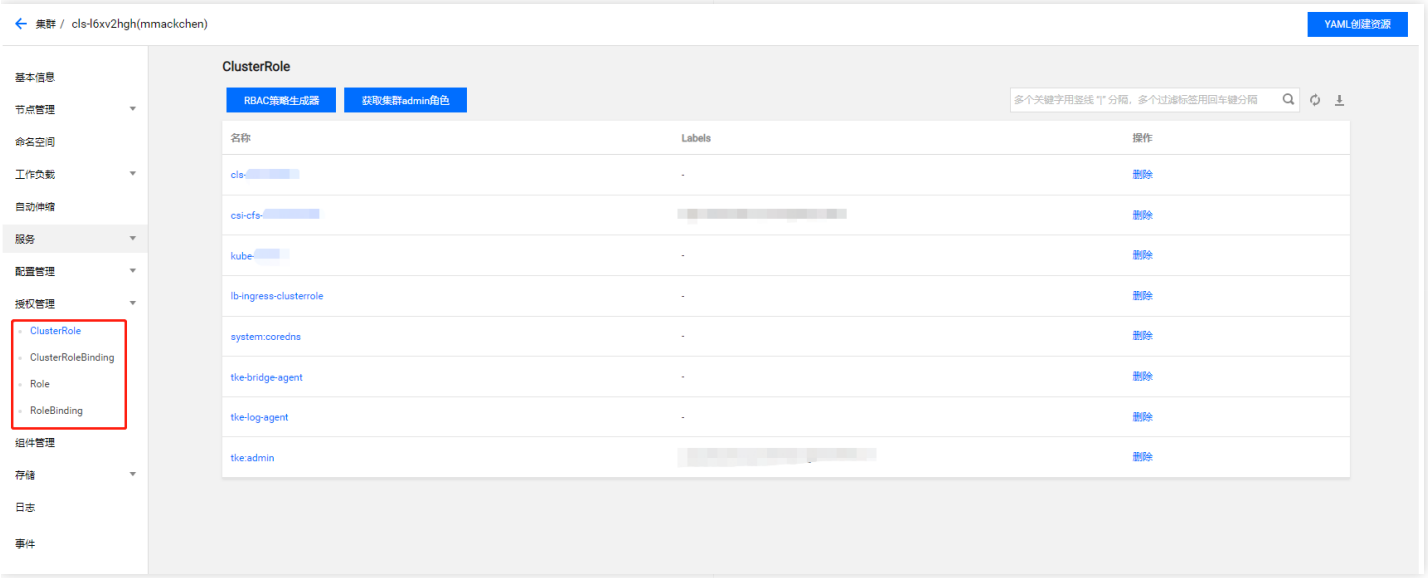
使用预设身份授权

预设角色说明

容器服务控制台通过 Kubernetes 原生的 RBAC 授权策略，针对子账号提供了细粒度的 Kubernetes 资源权限控制。同时提供了预设角色： Role 及 ClusterRole，详细说明如下：

Role 说明

容器服务控制台提供授权管理页，默认主账号及集群创建者具备管理员权限。可对其他拥有该集群 DescribeCluster Action 权限的子账号进行权限管理。



ClusterRole 说明

管理员（tke:admin）：对所有命名空间下资源的读写权限，具备集群节点、存储卷、命名空间、配额的读写权限，可配置子账号的读写权限。
运维人员（tke:ops）：对所有命名空间下控制台可见资源的读写权限，具备集群节点、存储卷、命名空间、配额的读写权限。
开发人员（tke:dev）：对所有命名空间或所选命名空间下控制台可见资源的读写权限。
只读人员（tke:ro）：对所有命名空间或所选命名空间下控制台可见资源的只读权限。
自定义：用户自定义 ClusterRole。

操作步骤

获取凭证

容器服务默认会为每个子账号创建独立的凭证，用户只需访问集群详情页或调用云 API 接口 DescribeClusterKubeconfig，即可获取当前使用账号的凭证信息 Kubeconfig 文件。通过控制台获取步骤如下：

1. 登录容器服务控制台，选择左侧导航栏中的【集群】。
2. 在“集群管理”页面中，单击所需目标集群 “操作”列【查看集群凭证】，即可显示凭证相关内容。

集群管理员可以访问凭证管理页，进行查看并更新所有账号下集群的凭证。

授权

> 说明：
>
> 请联系集群管理员（主账号、集群创建者或拥有 admin role 的用户）进行授权。

1. 在“集群管理”页面中，选择目标集群 ID。
2. 在集群详情页面中，单击【授权管理】 > 【ClusterRoleBinding】。
3. 在 “ClusterRoleBinding” 管理页面中，单击【RBAC策略生成器】。
4. 在“管理权限”页面的选择账号步骤中，勾选需授权的子账号并单击下一步。
5. 在“集群RBAC设置”步骤中，按照以下指引进项权限设置：
   - Namespace列表：按需指定权限生效的 Namespace 范围。
   - 权限：请参考界面中的“权限说明”，按需设置权限。

     您还可以单击【添加权限】，继续进行权限自定义设置。

     鉴权

     登录子账号，确认该账号已获得所授权限，则表示授权成功。

自定义策略授权

本文介绍如何通过自行编写 Kubernetes 的 ClusterRole 和 Role 以授予子账号特定权限，您可根据业务诉求进行对应操作。

策略语法说明

您可自行编写策略语法，或通过访问管理 CAM 策略生成器创建自定义策略。YAML 示例如下：

Role：命名空间维度

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: testRole
  namespace: default
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - create
  - delete
  - deletecollection
  - get
  - list
  - patch
  - update
  - watch
```

ClusterRole：集群维度

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: testClusterRole
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
```

- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch

操作步骤

> 说明：
>
> 该步骤以为子账号绑定自定义 ClusterRole 为例，与绑定 Role 的步骤基本一致，您可结合实际需求进行操作。

1. 登录容器服务控制台 ，选择左侧导航栏中的 集群。
2. 在“集群管理”页面中，选择需的集群 ID。
3. 在集群详情页面中，单击【授权管理】 > 【ClusterRole】。
4. 在 “ClusterRole” 管理页面中，单击右上角的【YAML创建资源】。
5. 在编辑界面输入自定义策略的 YAML 内容，单击完成即可创建 ClusterRole。
   该步骤以 ClusterRole：集群维度 YAML 为例，创建完成后，可在 “ClusterRole” 管理页面中查看自定义权限 “testClusterRole”。
6. 在 “ClusterRoleBinding” 管理页面中，单击【RBAC策略生成器】。
7. 在“管理权限”页面的“选择账号”步骤中，勾选需授权的子账号并单击下一步。
8. 进入“集群RBAC设置”界面，按照以下指引进项权限设置。
   - Namespace列表：按需指定权限生效的 Namespace 范围。
   - 权限：选择“自定义”，并单击选择自定义权限。按需在自定义权限列表中进行权限选择，本文以选择已创建的自定义权限 “testClusterRole” 为例。

     > 说明：
     >
     > 您还可以单击添加权限，继续进行权限自定义设置。
9. 单击完成即可完成授权操作。

监控
监控概述

概述

良好的监控环境为容器服务高可靠性、高可用性和高性能提供重要保证。您可以方便为不同资源收集不同维度的监控数据，能方便掌握资源的使用状况，轻松定位故障。
容器服务提供集群、节点、工作负载、Pod、Container 5个层面的监控数据收集和展示功能。
收集监控数据有助于您建立容器集群性能的正常标准。通过在不同时间、不同负载条件下测量容集群的性能并收集历史监控数据，您可以较为清楚的了解容器集群和服务运行时的正常性能，并能快速根据当前监控数据判断服务运行时是否处于异常状态，及时找出解决问题的方法。例如，您可以监控服务的 CPU 利用率、内存使用率和磁盘 I/O。

监控

容器服务的监控功能使用指引请参见 查看监控数据。
目前覆盖的监控指标请参见 监控及告警指标列表。

> 说明：
> 容器服务提供的监控功能主要覆盖 Kubernetes 对象的核心指标或事件，请结合 云监控 提供的基础资源监控（如云服务器、块存储、负载均衡等）使用，以保证更细的指标覆盖。

查看监控数据

操作场景

容器服务默认为所有集群提供基础监控功能，您可以通过以下方式查看容器服务的监控数据。

- 查看集群指标
- 查看节点指标
- 查看节点内 Pod 指标
- 查看工作负载指标
- 查看工作负载内 Pod 指标
- 查看 Pod 内 Container 指标

前提条件

已登录 TKE 控制台，并进入【集群】的管理页面。

操作步骤

查看集群指标

在需要查看监控数据的集群行中，单击 ，即可查看该集群监控信息页面。



查看节点指标

您可以通过以下操作查看节点和 Master&Etcd 节点的监控信息。

1. 选择需要查看的集群【ID/名称】，进入该集群的管理页面。
2. 展开【节点管理】，即可查看节点和 Master&Etcd 节点的监控信息。
    - 选择【节点】>【监控】，即可进入“节点监控”页面，查看监控信息。



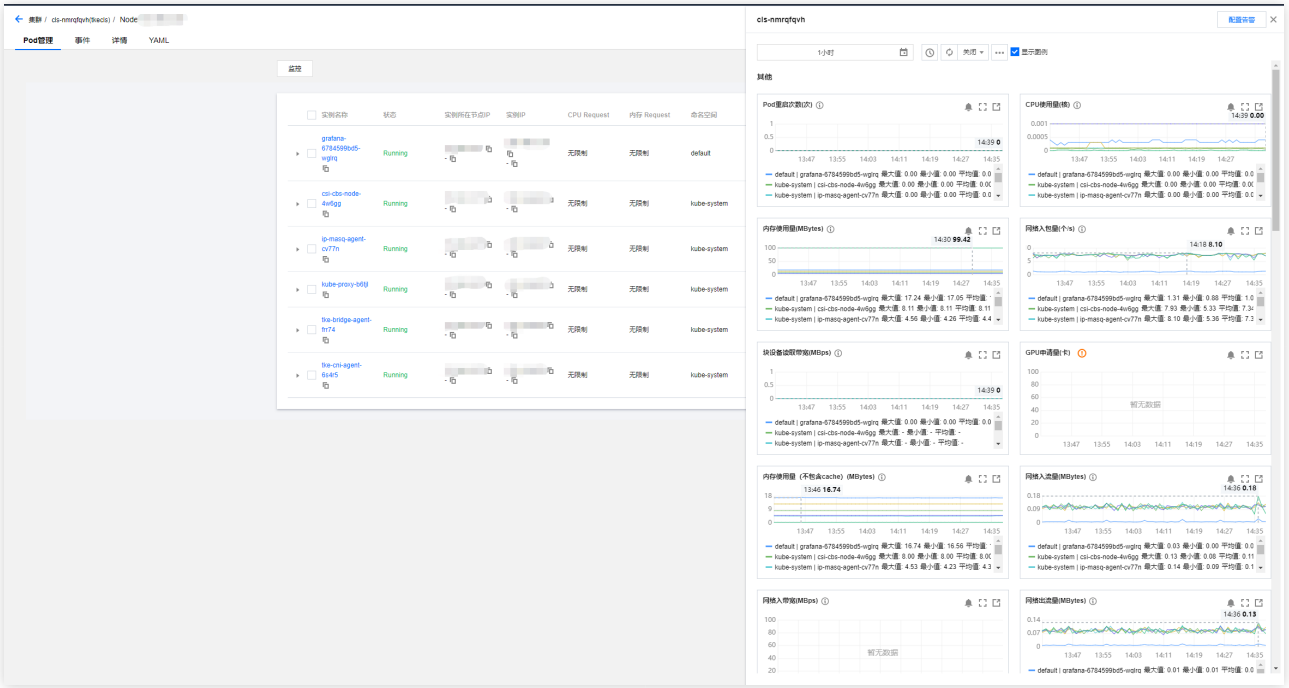
- 选择【Master&Etcd】>【监控】，即可进入“Master&Etcd 监控”页面，查看监控信息。



查看节点内 Pod 指标

您可通过以下两种方式查看节点内 Pod 指标。

1. 选择需要查看的集群【ID/名称】，进入该集群的管理页面。
2. 选择【节点管理】>【节点】，进入节点列表页面。
3. 根据实际需求，单击【ID/名称】选择所需查看节点，进入【Pod管理】详情页。
4. 单击【监控】，进入“节点监控”页面，即可查看到该节点内 Pod 的监控指标对比图。



查看工作负载指标

1. 选择需要查看的集群【ID/名称】，进入该集群的管理页面。
2. 选择【工作负载】>【任意类型工作负载】。例如，选择【Deployment】，进入 Deployment 管理页面。

3. 单击【监控】，即可查看该工作负载的监控信息。

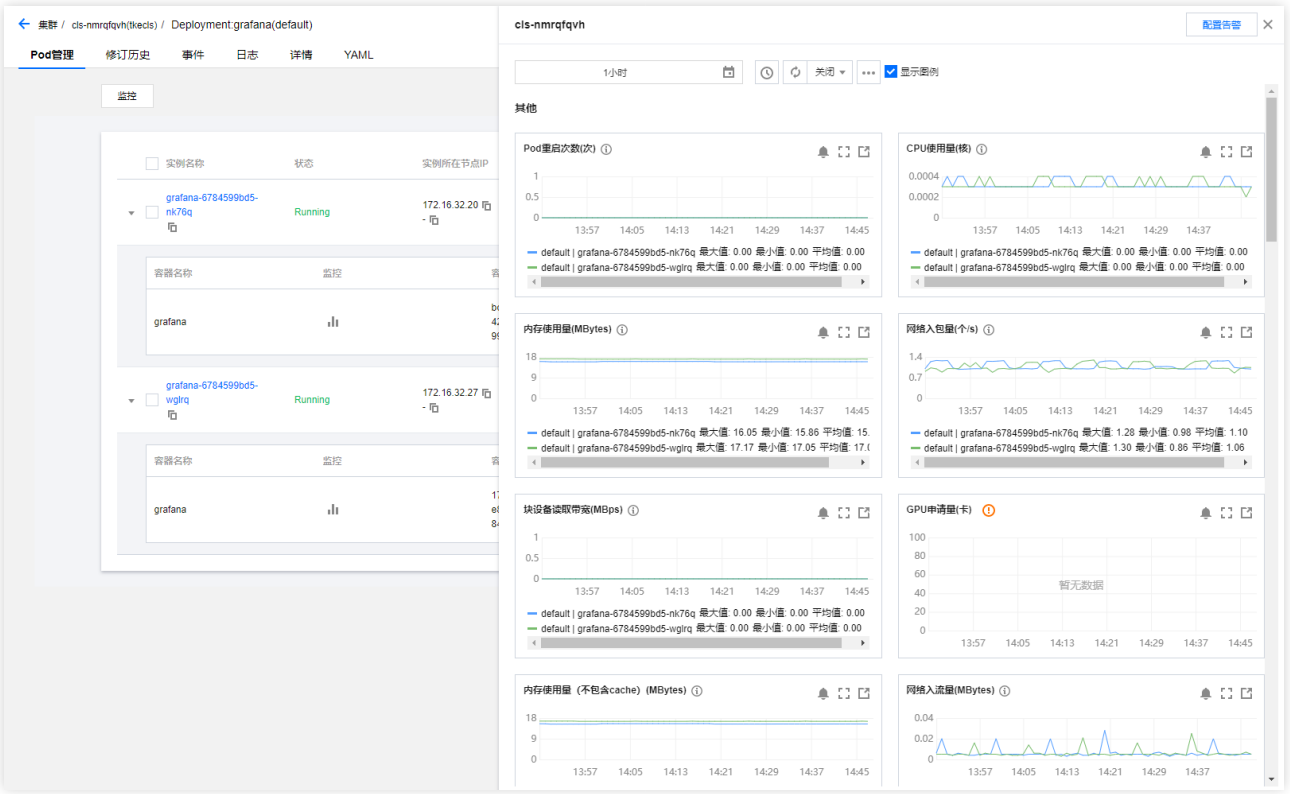
查看工作负载内 Pod 指标

1. 选择需要查看的集群【ID/名称】，进入该集群的管理页面。
2. 选择【工作负载】>【任意类型工作负载】。例如，选择【Deployment】，进入 Deployment 管理页面。
3. 选择需要查看的工作负载名称，进入该工作负载的管理页面。
4. 选择【Pod 管理】页签，单击【监控】，即可查看该工作负载内所有 Pod 的监控指标对比图。



查看 Pod 内 Container 指标

1. 参照查看工作负载内 Pod 指标 的[步骤1]-[步骤3]，进入工作负载详情页。
2. 选择【Pod 管理】页签，单击【监控】，即可查看该 Pod 内 Container 的监控指标对比图。
3. 单击实例名称前的三角形图标，即可查看该Pod的container相关信息。



监控及告警指标列表

监控

目前容器服务提供了以下维度的监控指标，所有指标均为统计周期内的平均值。

集群监控指标

监控指标	单位	说明
CPU利用率	%	集群整体的CPU利用率
内存利用率	%	集群整体的内存利用率

Master&Etcd 和普通节点监控指标

监控指标	单位	说明
Pod重启次数	次	节点内所有 Pod 的重启次数之和
异常状态	-	节点的状态，正常或异常
CPU利用率	%	节点内所有 Pod 的 CPU 使用量占节点总量之比
内存利用率	%	节点内所有 Pod 的内存使用量占节点总量之比
内网入带宽	bps	节点内所有 Pod 的内网入方向带宽之和
内网出带宽	bps	节点内所有 Pod 的内网出方向带宽之和
外网入带宽	bps	节点内所有 Pod 的外网入方向带宽之和
外网出带宽	bps	节点内所有 Pod 的外网出方向带宽之和
TCP连接数	个	节点保持的 TCP 连接数

集群节点更详细的监控指标请参考云服务器监控。
集群节点数据盘更详细的监控指标请参考云硬盘监控。

工作负载监控指标

监控指标	单位	说明
Pod 重启次数	次	工作负载内所有 Pod 的重启次数之和

监控指标	单位	说明
CPU 使用量	核	工作负载内所有 Pod 的 CPU 使用量
CPU 利用率（占集群）	%	工作负载内所有 Pod 的 CPU 使用量占集群总量之比
内存使用量	B	工作负载内所有 Pod 的内存使用量
内存利用率（占集群）	%	工作负载内所有 Pod 的内存使用量占集群总量之比
网络入带宽	bps	工作负载内所有 Pod 的入方向带宽之和
网络出带宽	bps	工作负载内所有 Pod 的出方向带宽之和
网络入流量	B	工作负载内所有 Pod 的入方向流量之和
网络出流量	B	工作负载内所有 Pod 的出方向流量之和
网络入包量	个/s	工作负载内所有 Pod 的入方向包数之和
网络出包量	个/s	工作负载内所有 Pod 的出方向包数之和

如果工作负载对集群外部提供服务，绑定的 Service 更详细的网络监控指标请参考负载均衡监控。

Pod 监控指标

监控指标	单位	说明
异常状态	-	Pod 的状态，正常或异常
CPU 使用量	核	Pod 的 CPU 使用量
CPU 利用率（占节点）	%	Pod 的 CPU 使用量占节点总量之比
CPU 利用率（占 Request）	%	Pod 的 CPU 使用量和设置的 Request 值之比
CPU 利用率（占 Limit）	%	Pod 的 CPU 使用量和设置的 Limit 值之比
内存使用量	B	Pod 的内存使用量，含缓存
内存使用量（不包含 Cache）	B	Pod 内所有 Container 的真实内存使用量（不含缓存）
内存利用率（占节点）	%	Pod 的内存使用量占节点总量之比
内存利用率（占节点，不包含 Cache）	%	Pod 内所有 Container 的真实内存使用量（不含缓存）占节点总量之比
内存利用率（占 Request）	%	Pod 的内存使用量和设置的 Request 值之比

监控指标	单位	说明
内存利用率（占 Request，不包含 Cache）	%	Pod 内所有 Container 的真实内存使用量（不含缓存）和设置的 Request 值之比
内存利用率（占 Limit）	%	Pod 的内存使用量和设置的 Limit 值之比
内存利用率（占 Limit，不包含 Cache）	%	Pod 内所有 Container 的真实内存使用量（不含缓存）和设置的 Limit 值之比
网络入带宽	bps	Pod 的入方向带宽之和
网络出带宽	bps	Pod 的出方向带宽之和
网络入流量	B	Pod 的入方向流量之和
网络出流量	B	Pod 的出方向流量之和
网络入包量	个/s	Pod 的入方向包数之和
网络出包量	个/s	Pod 的出方向包数之和

Container 监控指标

监控指标	单位	说明
CPU 使用量	核	Container 的 CPU 使用量
CPU 利用率（占节点）	%	Container 的 CPU 使用量占节点总量之比
CPU 利用率（占 Request）	%	Container 的 CPU 使用量和设置的 Request 值之比
CPU 利用率（占 Limit）	%	Container 的 CPU 使用量和设置的 Limit 值之比
内存使用量	B	Container 的内存使用量，含缓存
内存使用量（不包含 Cache）	B	Container 的真实内存使用量（不含缓存）
内存利用率（占节点）	%	Container 的内存使用量占节点总量之比
内存利用率（占节点，不包含 Cache）	%	Container 的真实内存使用量（不含缓存）占节点总量之比
内存利用率（占 Request）	%	Container 的内存使用量和设置的 Request 值之比
内存利用率（占 Request，不包含 Cache）	%	Container 的真实内存使用量（不含缓存）和设置的 Request 值之比

监控指标	单位	说明
内存利用率（占 Limit）	%	Container 的内存使用量和设置的 Limit 值之比
内存利用率（占 Limit，不包含 Cache）	%	Container 的真实内存使用量（不含缓存）和设置的 Limit 值之比
块设备读带宽	B/s	Container 从硬盘读取数据的吞吐量
块设备写带宽	B/s	Container 把数据写入硬盘的吞吐量
块设备读 IOPS	次/s	Container 从硬盘读取数据的 IO 次数
块设备写 IOPS	次/s	Container 把数据写入硬盘的 IO 次数

告警

目前容器服务提供了以下维度的告警指标，所有指标均为统计周期内的平均值。

集群告警指标

监控指标	单位	说明
CPU 利用率	%	集群整体的 CPU 利用率
内存利用率	%	集群整体的内存利用率
CPU 分配率	%	集群所有容器设置的 CPU Request 之和与集群总可分配 CPU 之比
内存分配率	%	集群所有容器设置的内存 Request 之和与集群总可分配内存之比
Apiserver 正常	-	Apiserver 状态，默认 False 时告警，仅独立集群支持该指标
Etcd 正常	-	Etcd 状态，默认 False 时告警，仅独立集群支持该指标
Scheduler 正常	-	Scheduler 状态，默认 False 时告警，仅独立集群支持该指标
Controll Manager 正常	-	Controll Manager 状态，默认 False 时告警，仅独立集群支持该指标

节点告警指标

监控指标	单位	说明
CPU 利用率	%	节点内所有 Pod 的 CPU 使用量占节点总量之比
内存利用率	%	节点内所有 Pod 的内存使用量占节点总量之比

监控指标	单位	说明
节点上 Pod 重启次数	次	节点内所有 Pod 重启次数之和
Node Ready	-	节点状态，默认 False 时告警

集群节点更详细的指标告警请参考云服务器监控和云监控创建告警策略。
集群节点数据盘更详细的指标告警请参考云硬盘监控和云监控创建告警策略。

Pod 告警指标

监控指标	单位	说明
CPU利用率（占Request）	%	Pod 的 CPU 使用量和设置的 Request 值之比
内存利用率（占Request）	%	Pod 的内存使用量和设置的 Request 值之比
实际内存利用率（占 Request）	%	Pod 内所有 Container 的真实内存使用量（不含缓存）设置的 Request 总量之比
CPU 利用率（占节点）	%	Pod 的 CPU 使用量占节点总量之比
内存利用率（占节点）	%	Pod 的内存使用量占节点总量之比
实际内存利用率（占节点）	%	Pod 内所有 Container 的真实内存使用量（不含缓存）占节点总量之比
CPU 利用率（占 Limit）	%	Pod 的CPU使用量和设置的 Limit 值之比
内存利用率（占 Limit）	%	Pod 的内存使用量和设置的 Limit 值之比
实际内存利用率（占 Limit）	%	Pod 内所有 Container 的真实内存使用量（不含缓存）和设置的 Limit 值之比
Pod 重启次数	次	Pod 的重启次数
Pod Ready	-	Pod 的状态，默认 False 时告警
CPU 使用量	核	Pod 的 CPU 使用量
内存使用量	MB	Pod 的内存使用量，含缓存
实际内存使用量	MB	Pod 内所有 Container 的真实内存使用量之和，不含缓存

HelmChart
概述

Helm 是管理 Kubernetes 应用程序的打包工具。更多详情请查看 Helm 官网文档。
容器服务平台集成 Helm 相关功能，提供了 Helm Chart 在指定集群内图形化的增删改查。

Helm实列

操作场景

您可以通过控制台管理 Helm 应用的更新、删除、回滚等操作。

前提条件

- 已有TKE 集群，且拥有0.28核 CPU，180MiB内存的资源。
- 仅支持 Kubernetes 1.8 以上版本的集群。

操作步骤

查看 Helm 实例详情

1. 登录 TKE 控制台。
2. 在左侧导航栏中，选择【HelmChart】>【Helm实列】，进入“Helm实列”页面。
3. 在“Helm实列”页面中，选择需要更新的 Helm 应用，单击需要更新的 Helm 应用的应用名，进入应用详情页面。
    - 在“应用详情”页签可查看工作负载、Service和其他的YAML。
    - 在“版本历史”页签可查看版本历史。
    - 在“参数配置”页签可查看参数配置。

更新 Helm 应用

1. 登录 TKE 控制台。
2. 在左侧导航栏中，选择【HelmChart】>【Helm实列】，进入“Helm实列”页面。
3. 在“Helm实列”页面中，选择需要更新的 Helm 应用，单击【更新应用】。
4. 在弹出的【更新Helm应用】窗口中，选择需要更新的版本，并根据业务需求填写自定义参数。
5. 单击【更新】。

回滚 Helm 应用

1. 登录 TKE 控制台。
2. 在左侧导航栏中，选择【HelmChart】>【Helm实列】，进入“Helm实列”页面。
3. 在“Helm实列”页面中，选择需要更新的 Helm 应用，单击需要更新的 Helm 应用的应用名，进入应用详情页面。
4. 选择【版本历史】页签，在需要回滚的版本行中，单击【回滚】。

删除 Helm 应用

1. 登录 TKE 控制台。
2. 在左侧导航栏中，选择【HelmChart】>【Helm实列】，进入“Helm实列”页面。
3. 在“Helm实列”页面中，选择需要更新的 Helm 应用，单击【删除】。
4. 在弹出的提示框中，单击【确认】。

Helm仓库

操作场景

您可以通过控制台上传 Chart 到 Helm 仓库，查看 Chart 详情，下载 Chart，删除 Chart。

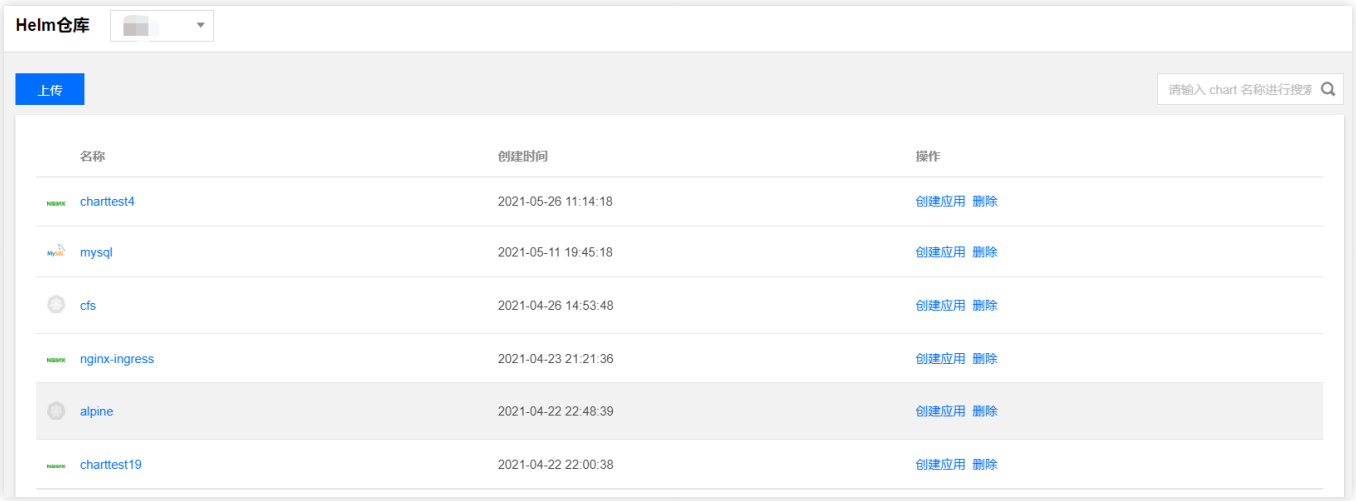
前提条件

- 已有TKE 集群，且拥有0.28核 CPU，180MiB内存的资源。
- 仅支持 Kubernetes 1.8 以上版本的集群。

操作步骤

上传 Helm Chart

1. 登录 TKE 控制台。
2. 在左侧导航栏中，选择【HelmChart】>【Helm仓库】，进入“Helm仓库”页面。



3. 在“Helm仓库”页面单击【上传】，弹出“上传Helm Chart”窗口。



4. 单击【Chart包】后面的框，在弹出的文件选择框中选择Chart包，单击【打开】。
5. 单击【上传】。

查看 Chart 详情

1. 登录 TKE 控制台。
2. 在左侧导航栏中，选择【HelmChart】>【Helm仓库】，进入“Helm仓库”页面。
3. 单击Chart的名称，进入详情页。
   展示“版本管理”、“基本信息”。

下载 Chart

1. 登录 TKE 控制台。
2. 在左侧导航栏中，选择【HelmChart】>【Helm仓库】，进入“Helm仓库”页面。
3. 单击Chart名称，进入详情页。
4. 在“版本管理”页签选择对应的版本，单击后面的【下载】。

删除 Chart

1. 登录 TKE 控制台。
2. 在左侧导航栏中，选择【HelmChart】>【Helm仓库】，进入“Helm仓库”页面。
3. 单击Chart名称，进入详情页。
4. 在“版本管理”页签选择对应的版本，单击后面的【删除】。

创建 Helm 应用

1. 登录 TKE 控制台。
2. 在左侧导航栏中，选择【HelmChart】>【Helm仓库】，进入“Helm仓库”页面。
3. 单击Chart后面的【创建应用】，进入【新建 Helm 应用】页面。
4. 输入“名称”、“集群”、“Namespace”、“参数”，选择Chart版本。

5. 单击【创建】。

健康检查

操作场景

集群健康检查功能是容器服务（TKE）为集群提供检查各个资源状态及运行情况的服务，检查报告将详细展示组件、节点、工作负载的状态和配置的检查内容。若出现异常项，可进行异常详情描述，并自动分析异常级别、异常原因、异常影响和修复建议等。

> 在健康检查过程中，您的集群内会自动新建 namespace tke-cluster-inspection，并安装一个 Daemonset 进行节点信息采集，检查结束后均会被自动删除。

主要检查项目

检查类别	检查项	检查内容	仅独立集群
资源状态	kube-apiserver 的状态	检测组件是否正在运行，如果组件以 Pod 形式运行，则检测其24小时内是否重启过。	是
	kube-scheduler 的状态	检测组件是否正在运行，如果组件以 Pod 形式运行，则检测其24小时内是否重启过。	是
	kube-controller-manager 的状态	检测组件是否正在运行，如果组件以 Pod 形式运行，则检测其24小时内是否重启过。	是
	etcd 的状态	检测组件是否正在运行，如果组件以 Pod 形式运行，则检测其24小时内是否重启过。	是
	kubelet 的状态	检测组件是否正在运行，如果组件以 Pod 形式运行，则检测其24小时内是否重启过。	否
	kube-proxy 的状态	检测组件是否正在运行，如果组件以 Pod 形式运行，则检测其24小时内是否重启过。	否
	dockerd 的状态	检测组件是否正在运行，如果组件以 Pod 形式运行，则检测其24小时内是否重启过。	否
	master 节点的状态	检测节点状态是否 Ready 且无其他异常情况，如内存不足，磁盘不足等。	是
	worker 节点的状态	检测节点状态是否 Ready 且无其他异常情况，如内存不足，磁盘不足等。	否
	各个工作负载的状态	检测工作负载当前可用 Pod 数是否符合其期望目标 Pod 数。	否
运行情况	kube-apiserver 的参数配置	根据 master 节点配置检测以下参数： - max-requests-inflight：给定时间内运行的变更类请求的最大值。 - max-mutating-requests-inflight：给定时间内运行的非变更类请求的最大值。 	是

检查类别	检查项	检查内容	仅独立集群
	kube-scheduler 的参数配置	根据 master 节点配置检测以下参数： - kube-api-qps：请求 kube-apiserver 使用的 QPS。 - kube-api-burst：和 kube-apiserver 通信的时候最大 burst 值。	是
	kube-controller-manager 的参数配置	根据 master 节点配置检测以下参数： - kube-api-qps：请求 kube-apiserver 使用的 QPS。 - kube-api-burst：和 kube-apiserver 通信的时候最大 burst 值。 	是
	etcd 的参数配置	根据 master 节点配置检测以下参数： quota-backend-bytes：存储大小。	是
	master 节点的配置合理性	检测当前 master 节点配置是否足以支撑当前的集群规模。	是
	node 高可用	检测目前集群是否是单节点集群； 检测当前集群节点是否支持多可用区容灾。 即当一个可用区不可用后，其他可用区的资源总和是否足以支撑当前集群业务规模。	否
	工作负载的 Request 和 Limit 配置	检测工作负载是否有未设置资源限制的容器，配置资源限制有益于完善资源规划、Pod 调度、集群可用性等。	否
	工作负载的反亲和性配置	检测工作负载是否配置了亲和性或者反亲和性，配置反亲和性有助于提高业务的高可用性。	否
	工作负载的 PDB 配置	检测工作负载是否配置了 PDB，配置 PDB 可避免您的业务因驱逐操作而不可用。	否
	工作负载的健康检查配置	检测工作负载是否配置了健康检查，配置健康检查有助于发现业务异常。	否
	HPA-IP 配置	当前集群剩余的 Pod IP 数目是否满足 HPA 扩容的最大数。	否

操作步骤

1. 登录 容器服务控制台，选择左侧导航栏中的【健康检查】。
2. 进入“健康检查”页面，选择需要健康检查的集群，并为其选择合适的检查方式。 健康检查的三种方式分别为批量检查、立即检查和自动检查。
   - 批量检查：适用于同时检查多个集群。
   - 立即检查：适用于只检查一个集群。
   - 自动检查：适用于需要周期性检查的集群。选择需要周期检查的集群，单击【自动检查】。如下图所示：



在“自动检查设置”弹窗中，可根据您的需求设置开启状态、检查周期和时刻。如下图所示：



3. 选择好检查方式之后，等待检查完成，可查看检查进度。如下图所示：



4. 检查完成后，可单击【查看结果】查看检查报告。如下图所示：



在检查报告页面，选择【资源状态】和【运行情况】分别查看资源状态和异常情况，单击【检查内容】可展示具体的检查内容，单击【异常】可查看异常级别、异常描述、异常原因、异常影响和修复建议。如下图所示：



最佳实践
构建简单web应用

操作场景

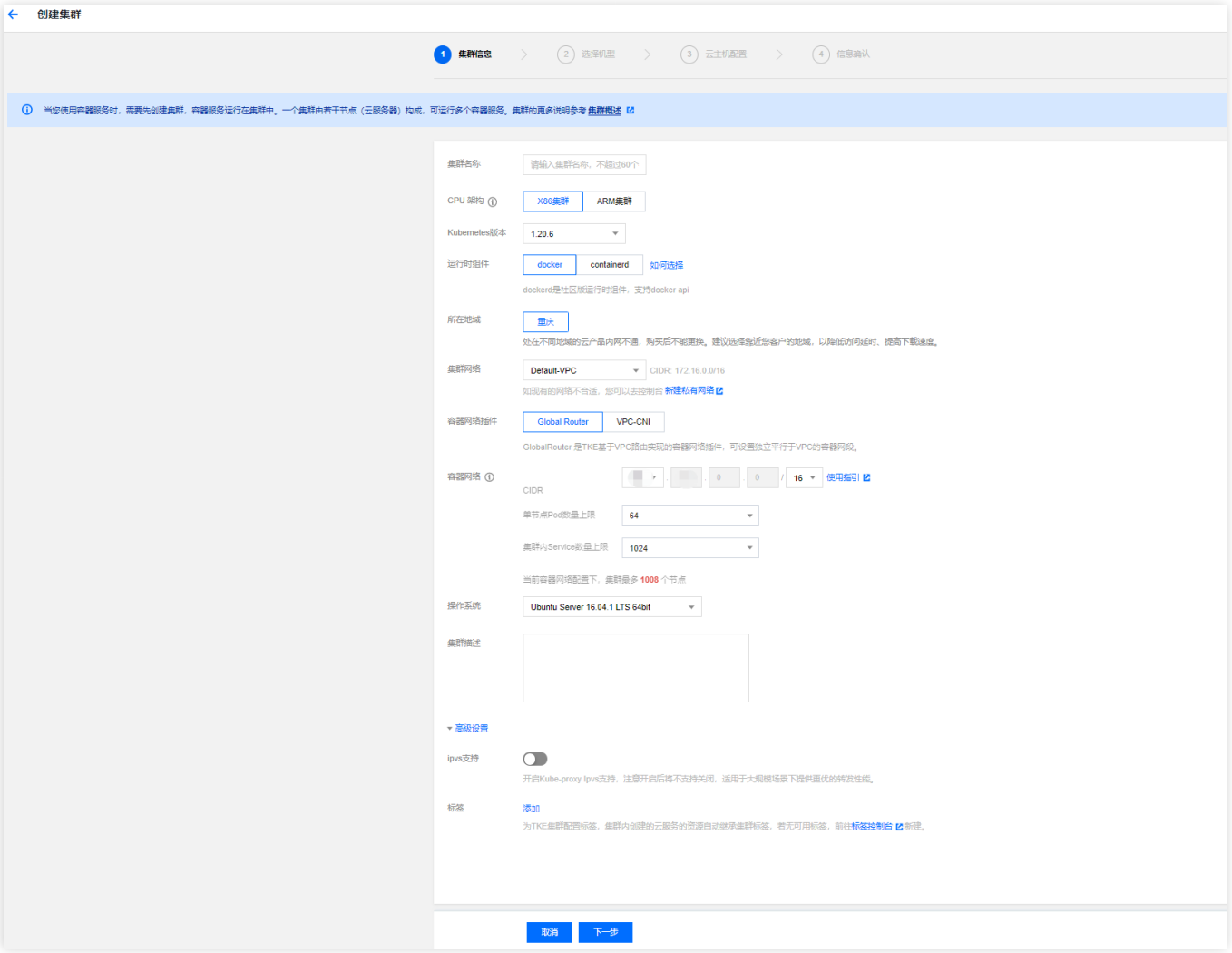
本文档指导您使用容器服务构建一个简单的 Web 应用。
Web 应用分为以下两部分：

- 前端服务，用于处理客户端的查询和写入请求。
- 数据存储服务，使用 redis，将写入的数据存放到 redis-master。通过访问 redis-slave 进行读取操作，redis-master 和 redis-slave 通过主从复制来保持数据同步。
  该应用是 kubernetes 项目自带的例子，链接地址为 https://github.com/kubernetes/kubernetes/tree/release-1.6/examples/guestbook

操作步骤

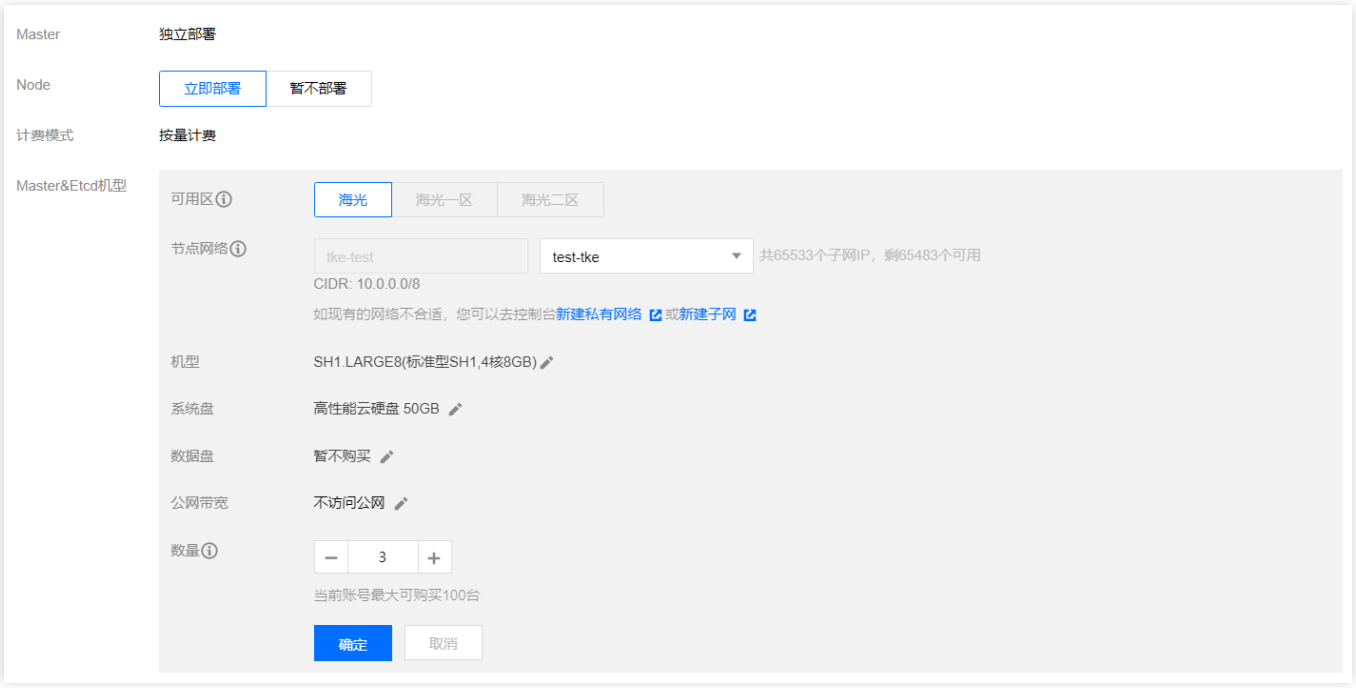
创建容器集群

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
3. 单击【新建】，进入“创建集群”页面。



4. 根据实际需求，填写以下参数。
   - 集群名称：您要创建的集群的名称。不超过60个字符。
   - CPU架构：根据需要选择对应架构。
   - Kubernetes 版本：选择 Kubernetes 版本。
   - 运行时组件：提供docker和containerd两种选择。
   - 所在地域：建议您根据所在地理位置选择靠近的地域。可降低访问延迟，提高下载速度。
   - 集群网络：如现有的网络不合适，您可以去控制台 新建私有网络。
   - 容器网络插件：选择容器网络插件。
   - 容器网络：为集群内容器分配在容器网络地址范围内的 IP 地址，单节点Pod数量上限和集群内Service数量上限。
5. 单击【下一步】。

   根据实际需求，为集群节点选择 Master 方式、计费模式，选择机型，配置机型、系统盘、公网带宽等。如下图所示：





6. 单击【下一步】。
7. 根据实际需求，选择操作系统、安全组，并选择登录方式和设置密码等。如下图所示：



8. 单击【下一步】。
9. 确认信息，单击【完成】。稍等几分钟即可创建成功。

创建 Web 应用

创建 redis-master 服务

1. 在左侧导航栏中，选择【服务】 > 【Service】，进入“Service”页面。
2. 单击【新建】，进入“新建服务”页面。
3. 根据实际需求，填写以下参数。
    - 服务名称：命名为 redis-master。
    - 所在地域：选择集群所在地域。
    - 运行集群：选择新创建的集群。
    - 运行容器
        - 名称：命名为 master。
        - 镜像：指定 master 容器的镜像。
        - 镜像版本：latest。
        - CPU/内存限制：（可选）设置 CPU 和内存资源的上限。

- 实例数量：选择【手动调节】，设置为1个。
- 服务访问方式：选择【仅在集群内访问】。
- 端口映射：选择 TCP 协议，并将服务端口和容器端口都设置为6379。其它服务可以通过服务名称 redis-master 以及端口6379访问到 master 容器。

4. 单击【创建服务】。

创建 redis-slave 服务

1. 单击【新建】，进入“新建服务”页面。
2. 根据实际需求，填写以下参数。
   - 服务名称：命名为 redis-slave。
   - 所在地域：选择集群所在地域。
   - 运行集群：选择新创建的集群。
   - 运行容器
     - 名称：命名为 slave。
     - 镜像：指定 master 容器的镜像为 ccr.ccs.finance.cloud.tencent.com/library/gb-redisslave 。
     - 镜像版本：latest。
     - CPU/内存限制：（可选）设置 CPU 和内存资源的上限。
   - 环境变量：添加一个名称为：GET_HOSTS_FROM，值为：dns 的环境变量。
   - 实例数量：选择【手动调节】，设置为1个。
   - 服务访问方式：选择【仅在集群内访问】。
   - 端口映射：选择 TCP 协议，并将服务端口和容器端口都设置为6379。其它服务可以通过服务名称 redis-slave 以及端口6379访问到 slave 容器。
3. 单击【创建服务】。

创建 frontend 服务

1. 单击【新建】，进入“新建服务”页面。
2. 根据实际需求，填写以下参数。
   - 服务名称：命名为 frontend。
   - 所在地域：选择集群所在地域。
   - 运行集群：选择新创建的集群。
   - 运行容器
     - 名称：命名为 frontend。
     - 镜像：指定 master 容器的镜像为 ccr.ccs.finance.cloud.tencent.com/library/gb-frontend 。
     - 镜像版本：latest。
     - CPU/内存限制：（可选）设置 CPU 和内存资源的上限。
   - 环境变量：添加一个名称为：GET_HOSTS_FROM，值为：dns 的环境变量。
   - 实例数量：选择【手动调节】，设置为1个。
   - 服务访问方式：选择【提供公网访问】。

- 端口映射：选择 TCP 协议，并将服务端口和容器端口都设置为80。用户通过浏览器访问负载均衡 IP 即可访问到 frontend 容器。

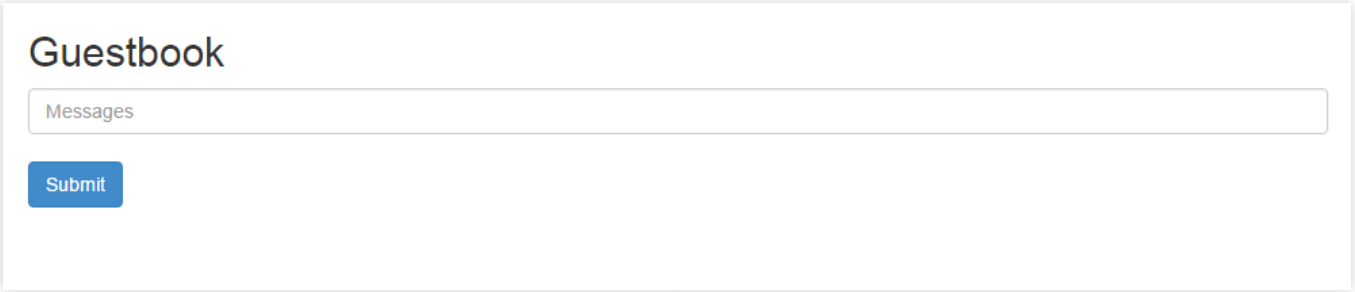
3. 单击【创建服务】。

查看服务

在左侧导航栏中，单击【服务】，即可查看新建的三个服务。如下图所示：



- 在创建 redis-master 和 redis-slave 服务时，因设置了 “仅在集群内访问” 的访问方式，这两个服务的 IP 地址只有一个内网 IP，且只能在集群内被其它服务访问。
- 在创建 frontend 服务时，因设置了 “提供公网访问” 的访问方式，该服务的 IP 地址有两个，分别为外网 IP（即公网负载均衡的 IP）和内网 IP，可以进行公网访问。由于 frontend 服务的访问端口为80，您可以在浏览器直接输入该外网 IP 即可访问页面。
  返回如下图所示页面，即表示可以正常访问 frontend 服务。



在输入框中输入任意的字符串，即可发现输入的记录已被保存，并且展示在页面下方。关闭并重新打开浏览器，重新该外网 IP 地址，原输入的数据依然存在，即表示输入的字符串已经保存到 redis 中。

开发实践

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);
require 'Predis/Autoloader.php';
Predis\Autoloader::register();
if (isset($_GET['cmd']) === true) {
  $host = 'redis-master';
```

```
  if (getenv('GET_HOSTS_FROM') == 'env') {
    $host = getenv('REDIS_MASTER_SERVICE_HOST');
  }
  header('Content-Type: application/json');
  if ($_GET['cmd'] == 'set') {
    $client = new Predis\Client([
      'scheme' => 'tcp',
      'host'   => $host,
      'port'   => 6379,
    ]);
    $client->set($_GET['key'], $_GET['value']);
    print('{"message": "Updated"}');
  } else {
    $host = 'redis-slave';
    if (getenv('GET_HOSTS_FROM') == 'env') {
      $host = getenv('REDIS_SLAVE_SERVICE_HOST');
    }
    $client = new Predis\Client([
      'scheme' => 'tcp',
      'host'   => $host,
      'port'   => 6379,
    ]);
    $value = $client->get($_GET['key']);
    print('{"data": "' . $value . '"}');
  }
} else {
  phpinfo();
} ?>
```

该实例是 Guestbook App 的 frontend 服务的完整代码。frontend 服务收到一个 HTTP 请求后，判断是否是 set 命令。

- 如果是 set 命令，则取出参数中的 key，value，连接到 redis-master 服务，并将 key, value 设置到 redis-master 中。
- 如果不是 set 命令，则连接到 redis-slave 服务，获取参数 key 对应的 value 值，并返回到客户端进行展示。

通过示例的 Web App，需要注意以下两点：

1. frontend 访问 redis-master 和 redis-slave 服务时，连接服务名和端口，集群自带 dns 服务将服务名解析成对应的服务 IP，并根据服务 IP 进行负载均衡。例如，redis-slave 服务有三个实例，在访问 redis-slave 服务时，直接连接 redis-slave 和6379，dns 会自动将 redis-slave 解析成 redis-slave 的服务 IP（即一个浮动 IP，类似于负载均衡的 IP），并根据 redis-slave 的服务 IP 自动进行负载均衡，将请求发往某个 redis-slave 服务的实例中。
2. 您可以为容器设置环境变量。在本例中，frontend 容器运行时，会读取 GET_HOSTS_FROM 环境变量，如果环境变量的值为 dns，则直接通过服务名来连接（推荐做法），否则需通过另一个环境变量来获取 redis-master 或者 redis-slave 的域名。

docker run 参数适配

docker run 参数适配

本文将介绍如何把在本地的 docker 中已经调试完毕的容器，在迁移到TCloudFinanceZone容器服务平台中运行时，对于 docker run 中的参数如何跟容器控制台的参数进行对应。这里我们以创建一个简单的 gitlab 服务为例。

gitlab容器的参数示例

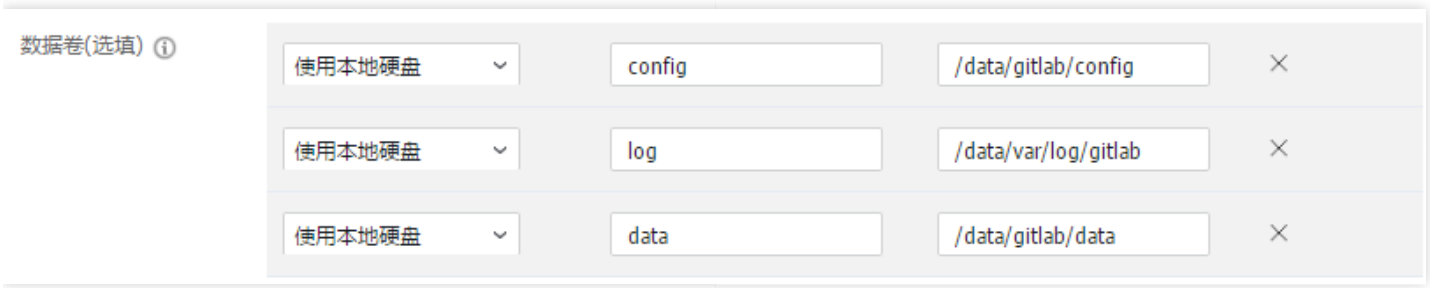
执行以下 docker run 命令创建出一个 gitlab 容器：

```
docker run \
-d \
-p 20180:80 \
-p 20122:22 \
--restart always \
-v /data/gitlab/config:/etc/gitlab \
-v /data/var/log/gitlab:/var/log/gitlab \
-v /data/gitlab/data:/var/opt/gitlab \
--name gitlab \
gitlab/gitlab-ce:8.16.7-ce.0
```

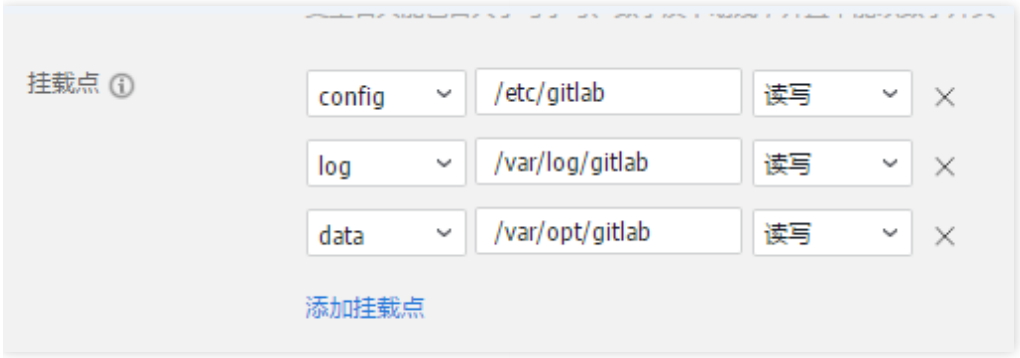
-d :容器在后台运行。容器平台都是以后台的形式来运行容器，所以本参数不需要在容器控制台指定。
-p :指定端口映射。我们这里映射了两个端口，容器端口分别是80和22，对外暴露的端口分别是20180和20122，对应到控制台，我们添加两条端口映射规则，并填写对应的容器端口和服务端口。由于我们的gitlab需要提供外网访问，我们选择了提供公网访问访问方式。



--restart :本参数用于指定在容器退出时，是否重启容器。容器平台创建的所有容器退出时，都会重启容器，所以本参数不需要在容器控制台指定。
-v :本参数用于指定容器卷。上面的命令指定了三个卷，对应到容器控制台，我们也需要添加三个数据卷，并在容器的高级设置里将这三个卷挂载到容器里。
首先我们创建三个卷：



接着我们在容器的高级设置里面，将三个卷分别挂载到容器里：



这里要注意的是，我们的数据卷的类型选择的是 使用本地硬盘 ，容器在运行过程中生产的数据会被保存到容器所在的节点上。如果容器被调度到其他的节点，数据就会丢失。我们可以使用 云硬盘 类型数据卷，容器的数据会保存到云硬盘中，即使容器被调度到其他的节点，容器卷的数据也不会丢失。

--name :容器运行的名字。这个参数，对应到容器控制台就是服务名，当然容器名也可以跟服务名用相同的名字。

其它参数

这里介绍我们执行 docker run 时，其它常见的参数：

-i :交互式执行容器。我们容器控制台只支持后台运行容器，所以本参数不支持。

-t :跟-i参数一样，本参数也不支持。

-e :容器运行的环境变量。比如用户执行以下的docker run命令：

```
docker run -e FOO='foo' -e BAR='bar' --name=container_name container_image
```

这里用户希望为容器添加两个环境变量，在我们的容器控制台创建服务时，在容器的高级设置里，可以添加容器的环境变量。在这里，变量名和变量值分别为：变量名:Foo，变量值:foo，以及变量名:BAR，变量值:bar。

Command和Args

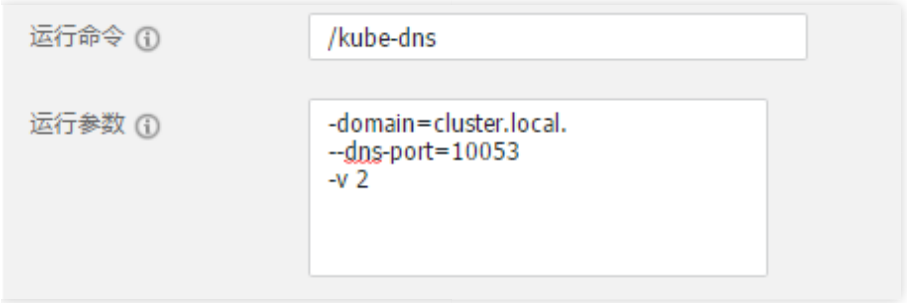
有时候我们希望在 docker run 的时候，指定进程的命令和参数，比如：

```
docker run --name=kubedns gcr.io/google_containers/kubedns-amd64:1.7 /kube-dns --domain=cluster.local. --dns-port=10053 -v 2
```

这里我们指定了容器进程的命令为： /kube-dns，并指定了三个参数： -domain=cluster.local.``--dns-

port=10053 和- v 2 。

在我们的控制台，我们可以这样指定：



单实例多容器实践

单实例多容器实践

单实例多容器优势

资源共享和通信：实例的存在使同个实例下的容器之间能更方便的共享数据和通信。同个实例下的容器使用相同的网络命名空间、IP地址和端口区间，相互之间能通过 localhost 来发现和通信。在一个无层次的共享网络中，每个实例都有一个 IP 地址用于跟别的物理主机和容器通信，实例的名字就用作容器通信时的主机名。在同个实例内运行的容器还共享一块存储卷空间，存储卷内的数据不会在容器重启后丢失，同时能被同实例下别的容器读取。
管理：相比原生的容器接口，实例通过提供更高层次的抽象，简化了应用的部署和管理。实例就像一个管理和横向部署和管理的单元，主机托管、资源共享、协调复制和依赖管理都可以自动处理。

常用单实例多容器应用场景

实例能应用于构建垂直集成应用栈，但它的主要为了集中管理一些辅助程序，如：

- 内容管理，文件和数据加载进程，本地 cache 管理进程等
- 日志压缩、rotation、备份、快照等
- 数据变化监听、日志和监控适配器，事件分发等
- 代理，网桥和适配器等
- 控制、管理、配置、升级程序等
  更多可查看实例应用场景。

访问管理
概述

如果您在云平台中使用到了容器服务（TKE），且该服务虽然由不同的人管理，但都统一使用您的云账号密钥，将存在以下问题：

- 您的密钥由多人共享，泄密风险高。
- 您无法限制其他人的访问权限，其他人误操作易造成安全风险。
  为解决以上问题，您可以通过使用子帐号来实现不同的人管理不同的业务。默认情况下，子帐号没有使用 TKE 的权限，我们需要创建策略来允许子帐号拥用他们所需要的权限。

简介

访问管理（Cloud Access Management，CAM）是TCloudFinanceZone提供的一套 Web 服务，主要用于帮助客户安全管理TCloudFinanceZone账户下的资源的访问权限。通过 CAM，您可以创建、管理和销毁用户（组），并通过身份管理和策略管理控制TCloudFinanceZone资源的使用权限。
当您使用 CAM 的时候，可以将策略与一个用户或者一组用户关联起来，策略能够授权或者拒绝用户使用指定资源完成指定任务。有关 CAM 策略的更多相关基本信息，请参照 策略语法。有关 CAM 策略的更多相关使用信息，请参照策略。
如果您不需要对子账户进行 CAM 相关资源的访问管理，您可以跳过此章节。跳过这些部分并不影响您对文档中其余部分的理解和使用。

入门

CAM 策略必须授权使用一个或多个 TKE 操作或者必须拒绝使用一个或多个 TKE 操作。同时还必须指定可以用于操作的资源（可以是全部资源，某些操作也可以是部分资源），策略还可以包含操作资源的条件。
TKE 部分 API 操作不支持资源级权限，意味着对于该类 API 操作，您不能在使用该类操作的时候指定某个具体的资源来使用，而必须要指定全部资源来使用。

服务授权相关角色权限说明

在使用容器服务（Kubernetes Engines，TKE）的过程中，为了能够使用相关云资源，会遇到多种需要进行服务授权的场景。每种场景通常对应不同的角色所包含的预设策略，其中主要涉及到 TKE_QCSRole 和 IPAMDofTKE_QCSRole 两个角色。本文档接下来将分角色展示各个授权策略的详情、授权场景及授权步骤。

TKE_QCSRole 角色

开通容器服务后，云平台会授予您的账户 TKE_QCSRole 角色的权限。该容器服务角色默认关联多个预设策略，为获取相关权限，需在特定的授权场景下执行对应的预设策略授权操作。操作完成之后，对应策略会出现在该角色的已授权策略列表中。

IPAMDofTKE_QCSRole 角色

IPAMDofTKE_QCSRole 角色为容器服务的 IPAMD 支持服务角色。

TKE镜像仓库资源级权限设置

容器镜像服务权限介绍

容器镜像的地址格式是： ccr.finance.cloud.tencent.com/${namespace}/${name}:${tag} 。
镜像仓库的权限围绕以下两个字段进行设置：

- ${namespace}：镜像仓库所属命名空间。
- ${name}：镜像仓库名称。

> 说明：
> 命名空间 ${namespace} 及镜像名字 ${name} 中不能包含斜杠 “/”。 ${tag} 字段目前只实现了删除操作鉴权，请参考 镜像 Tag 权限。

通过 ${namespace} ， ${name} 两个字段，管理者可以为协作者制定详细的权限方案，实现灵活的权限管理。
例如：

- 允许协作者 A 拉取镜像
- 禁止协作者 A 删除镜像
- 禁止协作者 B 拉取命名空间 ns1 中的镜像
  如果您不需要详细管理镜像仓库权限，可以使用 预设策略授权。如果您需要细致地管理协作者权限，请使用 自定义策略授权。容器镜像服务权限基于TCloudFinanceZone CAM 进行管理，您可以详细了解 CAM 的使用方法：
- 用户管理
- 策略管理
- 授权管理

预设策略授权

为了简化容器镜像服务权限管理，容器镜像服务内置了两个预设策略：

- 镜像仓库（CCR）全读写访问权限该预设策略配置了容器镜像服务所有权限，如果协作者关联该预设策略后，将与管理者拥有相同的镜像仓库权限。详情请查看 权限列表。
- 镜像仓库（CCR）只读访问权限该预设策略包含了容器镜像服务只读操作的权限，如果协作者在容器镜像服务中只关联了该预设策略，则以下操作将被禁止：
- docker push 推送镜像
- 新建镜像仓库命名空间
- 删除镜像仓库命名空间
- 创建镜像仓库
- 删除镜像仓库
- 删除镜像 Tag
  如果您不了解如何为协作者关联预设策略，请参考 CAM 文档：预设策略介绍、预设策略关联用户。

自定义策略授权

通过自定义策略，管理者可以为不同的协作者关联不同的权限。当您分配权限时，请考虑这些要素：

- 资源(resource)：该权限策略关联哪些镜像仓库，例如所有镜像仓库描述为 qcs::ccr:::repo/* ，详见 CAM 资源描述方式。
- 动作(action)：该权限策略对资源(resource)**进行哪些操作，如删除、新建等，通常使用接口进行描述。
- 效力(effect) ：该权限策略对协作者表现出的效果（允许/拒绝）。

一旦您规划好权限设置，就可以开始进行权限分配。下面以"允许协作者创建镜像仓库"为例进行说明：

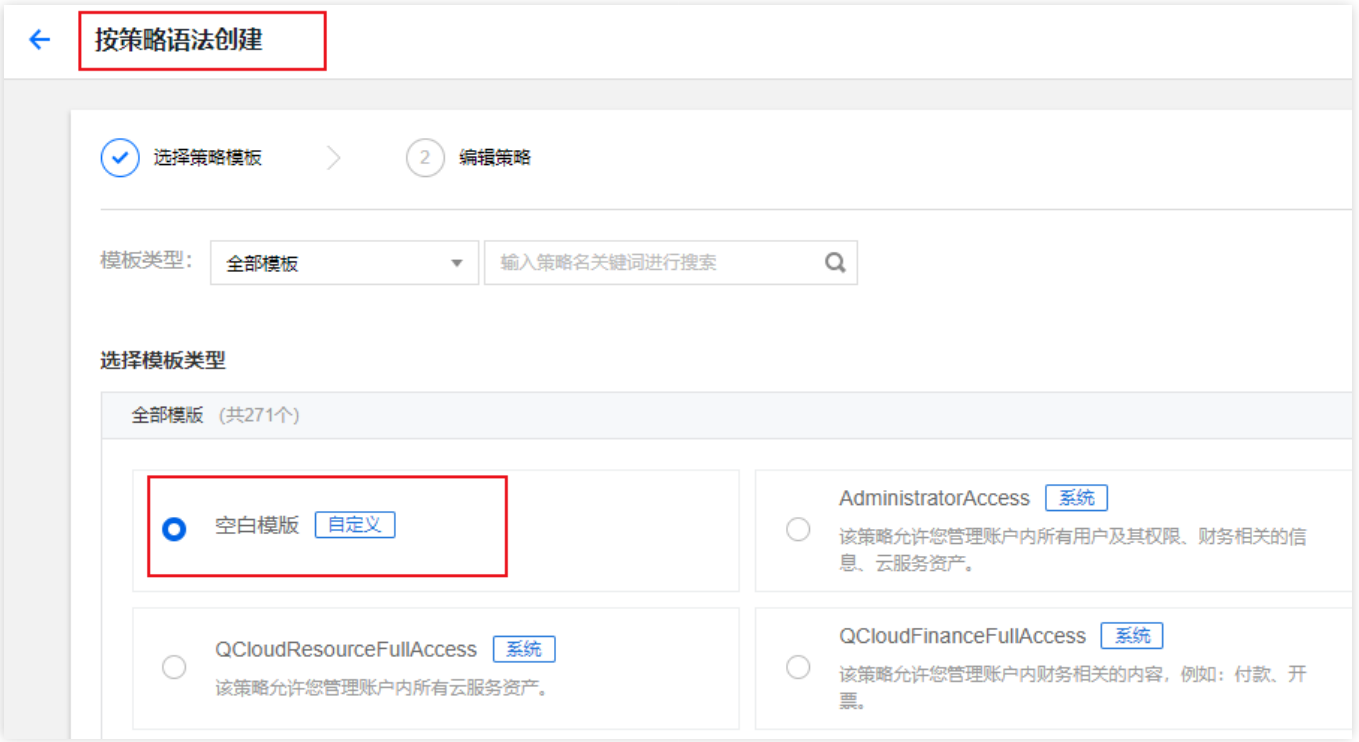
1. 创建自定义策略 访问管理文档。

2. 使用开发商账号登录控制台。

3. 进入 访问管理自定义策略管理页面，单击【新建自定义策略】，打开【选择创建策略方式】对话框。



4. 选择【按策略语法创建】> 【空白模板】。

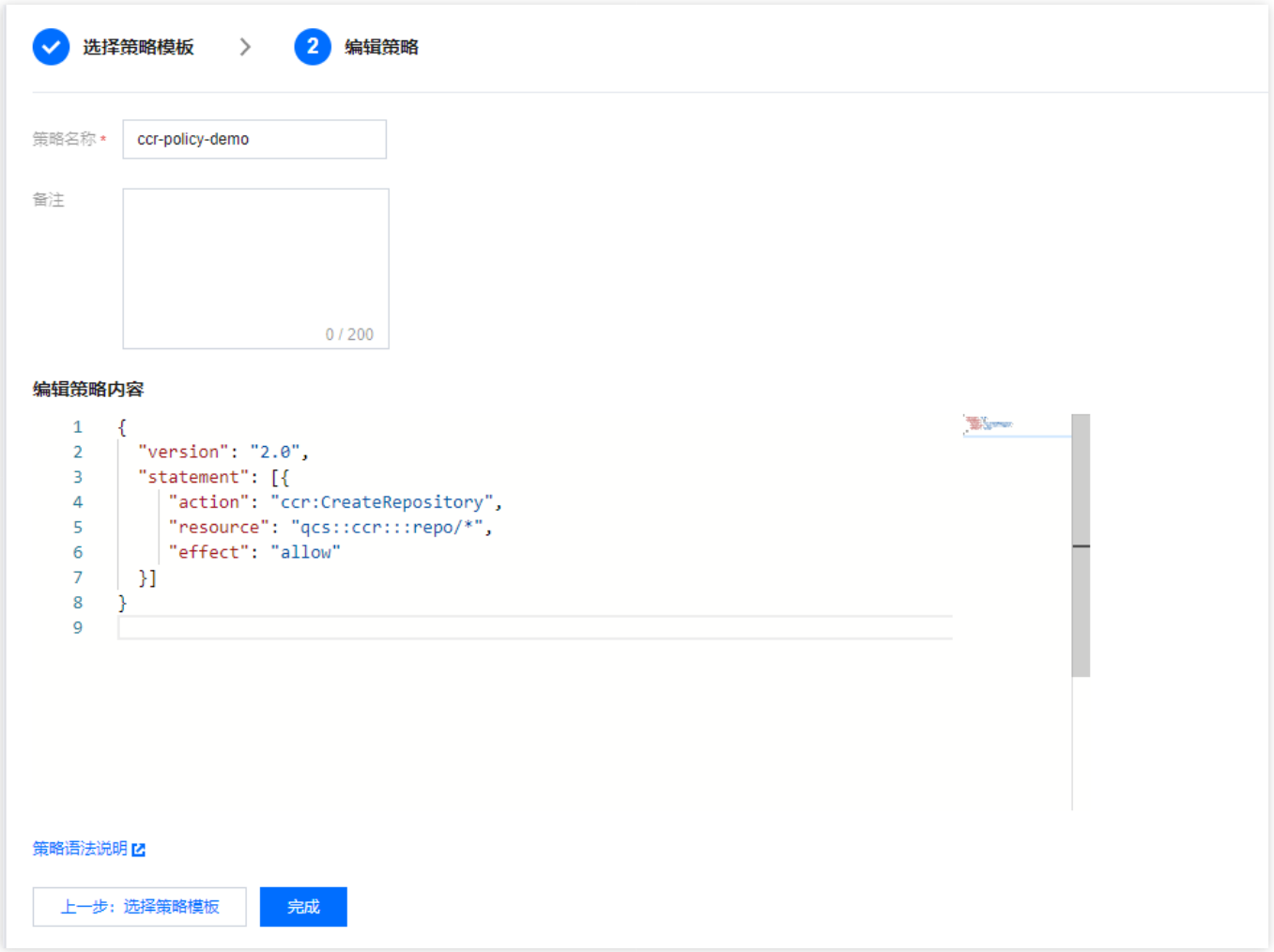


5. 单击【下一步：编辑策略模板】，进入【编辑策略】页面。

6. 设置策略名称，并将以下内容填入【编辑策略内容】编辑框中。

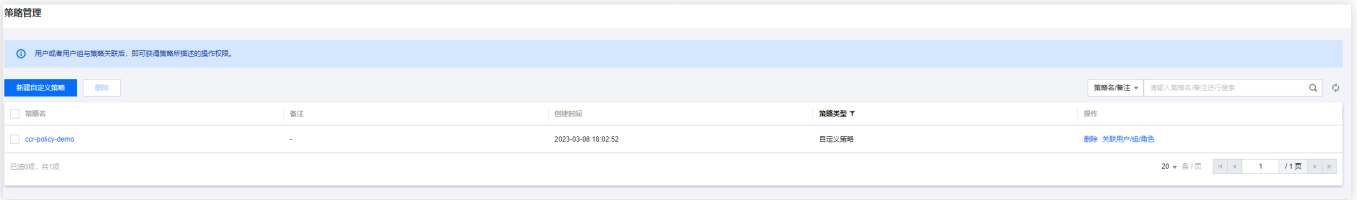
```
{
  "version": "2.0",
  "statement": [{
    "action": "ccr:CreateRepository",
    "resource": "qcs::ccr:::repo/*",
    "effect": "allow"
  }]
}
```

例如，将策略名称设置为 ccr-policy-demo



> 说明：
>
> resource 末尾使用 * 表示可以在任意命名空间下创建镜像仓库。

7. 单击【完成】，结束策略创建过程。



8. 关联自定义策略。步骤1中的策略（ccr-policy-demo）创建完成以后，您可以将其关联到任意协作者，详见访问管理文档。策略关联完成后协作者即拥有在任意命名空间下创建镜像仓库权限。
   _resource qcs::ccr:::repo/* 格式说明：
   - qcs::ccr::: 为固定格式，表示开发商的TCloudFinanceZone容器镜像仓库服务。
   - repo 为固定前缀，代表资源类型，这里是镜像仓库。
   - 斜杠(/)后面的 * 表示匹配所有镜像仓库。

关于 resource 更详细的描述，请参考 CAM 资源描述方式。

按资源进行授权

您可以同时为多个资源进行授权。例如：“允许删除命名空间 foo, bar 中的镜像仓库”，可以创建下面的自定义策略：

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "ccr:BatchDeleteRepository",
            "ccr:DeleteRepository"
        ],
        "resource": [
            "qcs::ccr:::repo/foo/*",
            "qcs::ccr:::repo/bar/*"
        ],
        "effect": "allow"
    }]
}
```

- qcs::ccr:::repo/foo/* 中 foo/* 表示镜像仓库命名空间 foo 下的所有镜像。
- qcs::ccr:::repo/bar/* 中 bar/* 表示镜像仓库命名空间 bar 下的所有镜像。

按动作(接口)进行授权

您可以对一个资源配置多个 action ，实现资源权限的统一管理。例如：“允许创建、删除、push 命名空间 foo 中的镜像仓库”，可以创建下面的自定义策略：

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "ccr:CreateRepository",
            "ccr:BatchDeleteRepository",
            "ccr:DeleteRepository",
            "ccr:push"
        ],
        "resource": "qcs::ccr:::repo/foo/*",
        "effect": "allow"
    }]
}
```

权限列表

docker client 权限

resource： qcs::ccr:::repo/${namespace}/${name} action：

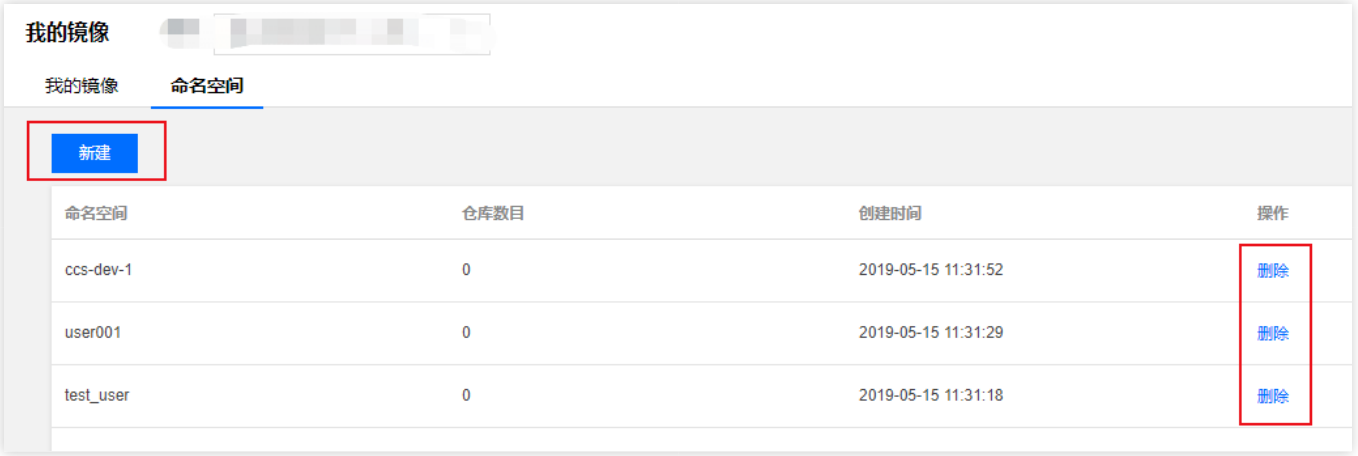
- ccr:pull 使用 docker 命令行 pull 镜像
- ccr:push 使用 docker 命令行 push 镜像

命名空间权限

resource： qcs::ccr:::repo/${namespace} action:

- ccr:CreateCCRNamespace 新建镜像仓库命名空间
- ccr:DeleteUserNamespace 删除镜像仓库命名空间

  功能指引：【容器服务】> 左侧导航栏【镜像仓库】 >【我的镜像】>【命名空间】。



镜像仓库权限

resource： qcs::ccr:::repo/${namespace}/${name} action：

- ccr:CreateRepository 创建镜像仓库
- ccr:DeleteRepository 删除镜像仓库
- ccr:BatchDeleteRepository 批量删除镜像仓库
- ccr:GetUserRepositoryList 查看镜像仓库列表

  功能指引：【容器服务】> 左侧导航栏【镜像仓库】 >【我的镜像】>【我的镜像】。

> 若要阻止协作者删除某些镜像，请配置多个 action 来实现。
>
> 例如，禁止删除任何镜像仓库。

```
{
 "version": "2.0",
 "statement": [{
 "action": [
 "ccr:BatchDeleteRepository",
 "ccr:DeleteRepository"
 ],
```

```
 "resource": "qcs::ccr:::repo/*",
 "effect": "deny"
 }]
}
```

镜像Tag权限

resource： qcs::ccr:::repo/${namespace}/${name}:${tag}
action： ccr:DeleteTag 删除镜像 Tag 权限
功能指引：【容器服务】> 左侧导航栏【镜像仓库】 >【我的镜像】>【我的镜像】> 单击某个镜像名称 > 【镜像版本】页面。

TKE集群资源级权限接口列表

资源级权限指的是能够指定允许用户对哪些资源具有执行操作的能力。TKE（原CCS）支持部分资源级权限，这意味着对于某些 TKE 操作，您可以控制何时允许用户执行操作（基于必须满足的条件）或是允许用户使用的特定资源。TKE 中可授权的资源类型：

资源类型	授权策略中的资源描述方法
集群相关	qcs::ccs:$region::cluster/*

下表将介绍当前支持资源级权限的 TKE API 操作。指定资源路径的时候，您可以在路径中使用 * 通配符。

> 注意：
> 如果某一个 TKE API 操作在下表中没有列出，则它不支持资源级权限。如果 TKE API 操作不支持资源级权限，您还是可以向用户授予使用该操作的权限，但是必须为策略语句的资源元素指定 * 。

API 操作	资源路径
DescribeClusterService	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
DescribeClusterServiceInfo	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
CreateClusterService	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId 负载均衡资源 qcs::clb:$region:$account:clb/* 云硬盘资源 qcs::cvm:$region:$account:volume/* qcs::cvm:$region:$account:volume/$diskId
ModifyClusterService	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId 负载均衡资源 qcs::clb:$region:$account:clb/* 云硬盘资源 qcs::cvm:$region:$account:volume/* qcs::cvm:$region:$account:volume/$diskId

API 操作	资源路径
DeleteClusterService	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
ModifyServiceDescription	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
DescribeServiceEvent	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
ResumeClusterService	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
PauseClusterService	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
RollBackClusterService	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
ModifyClusterServiceImage	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
RedeployClusterService	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
DescribeServiceInstance	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
ModifyServiceReplicas	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
DeleteInstances	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId

API 操作	资源路径
DescribeClusterNameSpaces	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
CreateClusterNamespace	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
DeleteClusterNamespace	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
DescribeCluster	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
CreateCluster	云服务器资源 qcs::cvm:$region:$account:instance/*
DeleteCluster	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
DescribeClusterInstances	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId
AddClusterInstances	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId 云服务器资源 qcs::cvm:$region:$account:instance/*
DeleteClusterInstances	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId 云服务器资源 qcs::cvm:$region:$account:instance/* qcs::cvm:$region:$account:instance/$instanceId
AddClusterInstancesFromExistedCvm	集群资源 qcs::ccs:region:account:cluster/* qcs::ccs:region:account:cluster/$clusterId 云服务器资源 qcs::cvm:$region:$account:instance/* qcs::cvm:$region:$account:instance/$instanceId

使用示例
配置子账号对TKE服务全读写或只读权限

操作场景

您可以通过使用访问管理（Cloud Access Management，CAM）策略让用户拥有在容器服务（TKE）控制台中查看和使用特定资源的权限。本文档中的示例指导您在控制台中配置部分权限的策略。

操作步骤

配置全读写权限

1. 登录 访问管理 控制台。
2. 在左侧导航栏中，单击 策略管理，进入策略管理页面。
3. 在策略管理页面中，单击 QcloudTKEFullAccess 策略行的【关联用户/组】。



4. 在弹出的 “关联用户/用户组” 窗口中，勾选需对 TKE 服务拥有全读写权限的账号，单击【确定】，即可完成子账号对 TKE 服务全读写权限的配置。
5. 在策略管理页面中，单击 TCRCCRFullAccess 策略行的【关联用户/组】。
6. 在弹出的 “关联用户/用户组” 窗口中，勾选需对镜像仓库拥有全读写权限的账号，单击【确定】，即可完成子账号对镜像仓库全读写权限的配置。 > 说明：
   如果您需要使用镜像仓库的触发器和自动构建功能，还需额外配置容器服务-持续集成（CCB）的相关权限。

配置只读权限

1. 登录 访问管理 控制台。
2. 在左侧导航栏中，单击 策略管理，进入策略管理页面。
3. 在策略管理页面中，单击 QcloudTKEReadOnlyAccess 策略行的【关联用户/组】。



4. 在弹出的“关联用户/用户组”窗口中，勾选需对 TKE 服务拥有只读权限的账号，单击【确定】，即可完成子账号对 TKE 服务只读权限的配置。
5. 在策略管理页面中，单击 TCRCCRReadOnlyAccess 策略行的【关联用户/组】。
6. 在弹出的“关联用户/用户组”窗口中，勾选需对镜像仓库拥有只读权限的账号，单击【确定】，即可完成子账号对镜像仓库只读权限的配置。 > 说明：
   如果您需要使用镜像仓库的触发器和自动构建功能，还需额外配置容器服务-持续集成（CCB）的相关权限。

配置子账号对单个TKE集群的管理权限

操作场景

您可以通过使用访问管理（Cloud Access Management，CAM）策略让用户拥有在容器服务（TKE）控制台中查看和使用特定资源的权限。本文档中的示例指导您在控制台中配置单个集群的策略。

操作步骤

配置对单个集群全读写权限

1. 登录 访问管理 控制台。
2. 在左侧导航栏中，单击 【策略管理】，进入策略管理页面。
3. 单击【新建自定义策略】，选择 “按策略语法创建” 方式。
4. 选择 “空白模板” 类型，单击【下一步：编辑策略】。
5. 自定义策略名称，将 “编辑策略内容” 替换为以下内容。

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "ccs:*"
      ],
      "resource": [
        "qcs::ccs:sh::cluster/cls-XXXXXXX", // 替换成您想赋予权限的指定地域下的集群
        "qcs::cvm:sh::instance/*"
      ],
      "effect": "allow"
    },
    {
      "action": [
        "cvm:*"
      ],
      "resource": "*",
      "effect": "allow"
    },
    {
      "action": [
```

```
            "vpc:*"
        ],
        "resource": "*",
        "effect": "allow"
    },
    {
        "action": [
            "clb:*"
        ],
        "resource": "*",
        "effect": "allow"
    },
    {
        "action": [
            "monitor:*",
            "cam:ListUsersForGroup",
            "cam:ListGroups",
            "cam:GetGroup",
            "cam:GetRole"
        ],
        "resource": "*",
        "effect": "allow"
    }
]
}
```

6. 在“编辑策略内容”中，将 qcs::ccs:sh::cluster/cls-XXXXXXX 修改为您想赋予权限的指定地域下的集群。

7. 例如，您需要为城市B地域的 cls-69z7ek9l 集群赋予全读写的权限，将 qcs::ccs:sh::cluster/cls-XXXXXXX 修改为 "qcs::ccs:gz::cluster/cls-69z7ek9l" 。



> 说明：
请替换成您想赋予权限的指定地域下的集群 ID。如果您需要允许子账号进行集群的扩缩容，还需要配置子账号用户支付权限。

8. 单击【创建策略】，即可完成对单个集群全读写权限的配置。

配置对单个集群只读权限

1. 登录访问管理 控制台。

2. 在左侧导航栏中，单击 【策略管理】，进入策略管理页面。

3. 单击【新建自定义策略】，选择 “按策略语法创建” 方式。

4. 选择 “空白模板” 类型，单击【下一步：编辑策略】。

5. 自定义策略名称，将 “编辑策略内容” 替换为以下内容。

```
{
    "version": "2.0",
    "statement": [
        {
            "action": [
                "ccs:Describe*",
                "ccs:Check*"
            ],
            "resource": "qcs::ccs:gz::cluster/cls-1xxxxxx", // 替换成您想赋予权限的指定地域下的集群
            "effect": "allow"
        },
        {
            "action": [
                "cvm:Describe*",
                "cvm:Inquiry*"
            ],
            "resource": "*",
            "effect": "allow"
        },
        {
            "action": [
                "vpc:Describe*",
                "vpc:Inquiry*",
                "vpc:Get*"
            ],
            "resource": "*",
            "effect": "allow"
        },
```

```
        {
            "action": [
                "clb:Describe*"
            ],
            "resource": "*",
            "effect": "allow"
        },
        {
            "effect": "allow",
            "action": [
                "monitor:*",
                "cam:ListUsersForGroup",
                "cam:ListGroups",
                "cam:GetGroup",
                "cam:GetRole"
            ],
            "resource": "*"
        }
    ]
}
```

6. 在“编辑策略内容”中，将 qcs::ccs:gz::cluster/cls-1xxxxxx 修改为您想赋予权限的指定地域下的集群。如下图所示：

7. 例如，您需要为城市A地域的 cls-19a7dz9c 集群赋予只读的权限，将 qcs::ccs:gz::cluster/cls-1xxxxxx 修改为 qcs::ccs:bj::cluster/cls-19a7dz9c 。



> 说明：
请替换成您想赋予权限的指定地域下的集群 ID。

8. 单击【创建策略】，即可完成对单个集群只读权限的配置。

购买容器集群
集群新增资源所属项目说明

集群新增资源所属项目说明

总述

如需要通过分项目进行财务核算等，请先阅读以下内容：

1. 集群无项目属性， 集群内云服务器、负载均衡器等资源有项目属性。
2. 集群新增资源所属项目：仅将新增到该集群下的资源归属到该项目下。

建议

1. 建议集群内的所有资源在同一个项目。
2. 如若需要集群内云服务器分布在不同的项目，请自行前往云服务器控制台迁移项目。
3. 若云服务器项目不同，那么云服务器所属的 安全组实例 不同， 请尽量让同一集群下的云服务器的 安全组规则 相同。

容器服务安全组设置

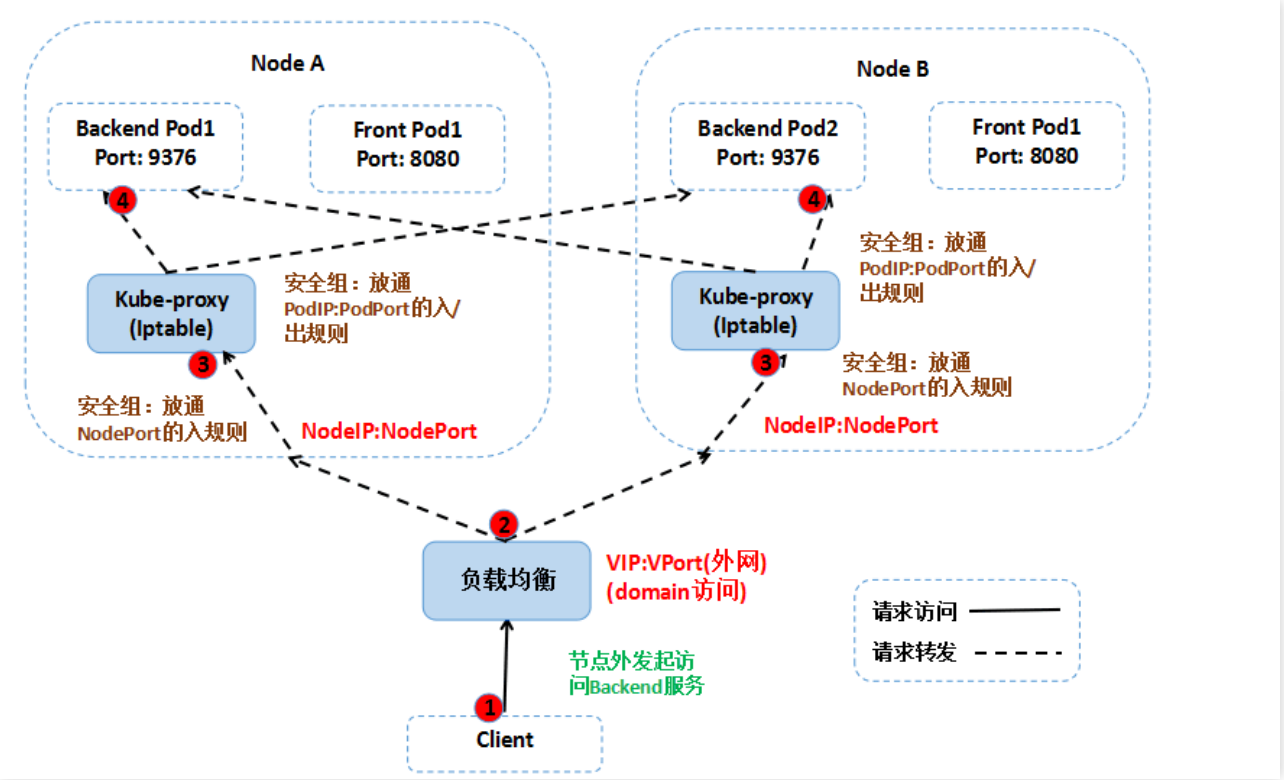
安全问题向来是一个大家非常关注的问题，TCloudFinanceZone将安全性作为产品设计中的最高原则，严格要求产品做到安全隔离，容器服务同样非常看重这一点。TCloudFinanceZone的基础网络可以提供充分的安全保障，容器服务选择了网络特性更丰富的 VPC 私有网络 来作为容器服务的底层网络，本文档主要介绍容器服务下使用安全组的最佳实践，帮助大家选择安全组策略。

安全组

安全组是一种有状态的包过滤功能的虚拟防火墙，它用于设置单台或多台云服务器的网络访问控制，是TCloudFinanceZone提供的重要的网络安全隔离手段。更多安全组的介绍可以查看 安全组。

使用容器服务选择安全组的原则

1. 由于在容器集群中，服务实例采用分布式的方式进行部署，不同的服务实例混部在集群的节点上。建议同一个集群下的主机绑定同一个安全组，集群的安全组不添加其他云服务器。
2. 安全组只对外开放最小权限。
3. 需放通以下容器服务使用规则：
    - 放通容器实例网络和集群节点网络
      当服务访问到达主机节点后，会通过 Kube-proxy 模块设置的 iptables 规则将请求进行转发到服务的任意一个实例。由于服务的实例有可能在另外的节点上，这时会出现跨节点访问。访问的目的 IP 为服务的实例的 IP ，访问的目的 IP 为节点 IP (或者节点上集群 cbr0 网桥的 IP)。这就需要在对端节点上放通容器实例网络和集群节点网络访问。
    - 同一 VPC 不同集群互访的情况，需要放通对应集群的容器网络和节点网络。
    - 需要 SSH 登录节点的放通 22 端口
    - 放通节点 30000 ~ 32768 端口
      在访问路径中，需要通过负载均衡器将数据包转发到容器集群的 NodeIP：NodePort 上。其中 NodeIP 为集群中任意一节点的主机 IP ，而 NodePort 是在创建服务时容器集群为服务默认分配的，NodePort 的范围为 30000 ~ 32768 。
      下图以外网访问服务为例：



建议

建议通过容器服务提供的安全组模板来配置集群的安全组。安全组的具体配置规则如下：

入站规则：

协议	端口	网段	是否允许	说明
TCP	30000-32768	0.0.0.0/0	允许	放通所有 IP 对 30000-32768 端口 TCP 访问
UDP	30000-32768	0.0.0.0/0	允许	放通所有 IP 对 30000-32768 端口 UDP 访问
All	traffic ALL	10.0.0.0/8	允许	放通 10.0.0.0/8 内网网段的访问
All	traffic ALL	172.16.0.0/12	允许	放通 172.16.0.0/12 内网网段的访问
All	traffic ALL	192.168.0.0/16	允许	放通 192.168.0.0/16 内网网段的访问
TCP	22	0.0.0.0/0	允许	放通所有 IP 对 22 端口的访问
All	traffic ALL	0.0.0.0/0	拒绝	未匹配已有规则，则拒绝

出站规则：

协议	端口	网段	是否允许	说明
All	traffic ALL	0.0.0.0/0	允许	放通所有规则

容器节点配置该规则，能够满足不同的访问方式访问集群中服务。
集群中服务的访问方式，可以参考 服务访问方式设置 。

容器服务节点公网IP说明

如果对业务安全有要求不希望业务直接暴露到公网，同时又希望访问公网，您可以使用TCloudFinanceZone NAT网关。下文将介绍如何使用 NAT 网关来访问公网。

公网 IP

分配的公网 IP 将提供以下作用：

- 通过公网 IP 登录到集群的节点机器。
- 通过公网 IP 访问外网服务。

外网带宽

创建外网服务时，外网负载均衡使用的是节点的带宽和流量，若需提供外网服务，节点需要有外网带宽。如果业务不需要外网服务，可以选择不购买外网带宽。

NAT 网关

云服务器不绑定弹性公网 IP ，所有访问 Internet 流量通过 NAT 网关转发。此种方案中，云服务器访问 Internet 的流量会通过内网转发至 NAT 网关，因而不会受云服务器购买时公网带宽的带宽上限限制，NAT 网关产生的网络流量费用也不会占用云服务器的公网带宽出口。通过 NAT 网关访问 Internet，您需要完成以下两个步骤：

第一步：创建 NAT 网关

1. 登录 私有网络控制台，单击左侧导航栏中的【NAT 网关】。
2. 在 “NAT网关” 管理页面，单击【新建】。
3. 在弹出的”新建NAT网关”窗口中，填写以下参数。
    - 网关名称：自定义。
    - 所属网络：选择 NAT 网关服务的私有网络。
    - 网关类型：根据实际需求进行选择，网关类型创建后可更改。
    - 出带宽上限：根据实际需求进行设置。
    - 弹性 IP：为 NAT 网关分配弹性 IP，您可以选择已有的弹性 IP，或者重新购买并分配弹性 IP。
4. 单击【创建】，即可完成 NAT 网关的创建。

    > NAT 网关创建时将会冻结1小时的租用费用。

第二步：配置相关子网所关联的路由表

> 完成创建 NAT 网关后，您需要在私有网络控制台路由表页配置路由规则，以将子网流量指向 NAT 网关。

1. 单击左侧导航栏中的【路由表】。
2. 在路由表列表中，单击需要访问 Internet 的子网所关联的路由表 ID/名称，进入路由表详情页。
3. 在“路由策略”栏中，单击【新增路由策略】。
4. 在弹出的“新增路由”窗口中，填写【目的端】，将【下一跳类型】选择为【NAT 网关】，并将【下一跳】选择为已创建的 NAT 网关 ID。
5. 单击【确定】。
   完成以上配置后，关联此路由表的子网内的云服务器访问 Intenet 的流量将指向 NAT 网关。

其他方案

方案一：使用弹性公网 IP

云服务器只绑定弹性公网 IP ，不使用 NAT 网关。此种方案，云服务器所有访问 Internet 流量通过弹性公网 IP 出，会受到云服务器购买时公网带宽的带宽上限限制。访问公网产生的相关费用，根据云服务器网络计费模式而定。
使用方法：请参见 弹性公网IP操作指南。

方案二：同时使用 NAT 网关和弹性公网 IP

如果您同时使用 NAT 网关和弹性公网 IP，此种方案中，所有云服务器主动访问 Internet 的流量只通过内网转发至 NAT 网关，回包也经过 NAT 网关返回至云服务器。此部分流量不会受云服务器购买时公网带宽的带宽上限限制，NAT 网关产生的网络流量费用不会占用云服务器的公网带宽出口。如果来自 Internet 的流量主动访问云服务器的弹性公网 IP，则云服务器回包统一通过弹性公网 IP 返回，这样产生的公网出流量受到云服务器购买时公网带宽的带宽上限限制。访问公网产生的相关费用，根据云服务器网络计费模式而定。

容器及节点网络设置

设置集群和节点网络

集群网络与容器网络是集群的基本属性。通过设置集群网络和容器网络可以规划集群的网络划分。

- 集群网络：将为集群内主机分配在节点网络地址范围内的 IP 地址，您可以选择私有网络中的子网用于集群的节点网络，更多私有网络介绍可看 私有网络和子网 。
- 容器网络：将为集群内容器分配在容器网络地址范围内的 IP 地址，您可以自定义三大私有网段作为容器网络，根据您选择的集群内服务数量的上限，自动分配适当大小的 CIDR 段用于 kubernetes service；根据您选择 Pod 数量上限/节点，自动为集群内每台云服务器分配一个适当大小的网段用于该主机分配 Pod 的 IP 地址。

集群网络与容器网络的关系

- 集群网络和容器网络网段不能冲突
- 同一 VPC 内，不同集群的容器网络网段不能冲突
- 容器网络和 VPC 路由冲突时，优先在容器网络内转发

集群网络与TCloudFinanceZone其他资源通信

- 集群内容器与容器之间互通
- 集群内容器与节点直接互通
- 集群内容器与 云数据库 CDB、云存储 redis、云数据库 Memcached 等资源同一 VPC 下内网互通

容器网络说明

1. 容器 CIDR：集群内 Sevice、 Pod 等资源所在网段
2. Services 数量上限/集群：决定分配给 Sevice 的 CIDR 大小
   容器服务集群默认创建3个 Sevice：kubernetes、hpa-metrics-service、kube-dns，同时还会有2个广播地址和网络号，因此用户可以使用的是 serviceMax-5。
3. Pod 数量上限/Node：决定分配给每个 Node 的 CIDR 的大小
   容器服务集群默认创建三个3个 Pod：kube-dns-xxxx、kube-dns-xxxx、l7-lb-controller-xxxx。
   对于一个 Node 上的 Pod，有三个地址不能分配分别是网络号，广播地址和网关地址。故 Node 其最大的 Pod 数目=podMax-3

容器节点硬盘设置

说明

容器服务创建集群和扩展集群时可设置容器节点的系统盘的类型和大小、数据盘的类型和大小，可选择不同类型的硬盘来满足您不同业务的要求。

建议

1.容器的目录存储在系统盘中，建议您创建50G的系统盘。
2.如果您对系统盘有要求，可以在集群初始化时，将docker的目录自行调整到数据盘上。

常见问题
集群类
集群相关

创建集群常见问题

创建集群时，云服务器可以不选取公网 IP 么？

云服务器可以不选取公网 IP，无公网IP的云服务器只能拉取镜像仓库下我的镜像，不能拉取 dockerhub 以及第三方镜像。
无公网 IP，但有外网带宽的云服务器可以通过绑定弹性 IP 来访问 Internet。

创建集群时，选择所属网络的作用是什么？

选择的所属网络和子网，是集群内云服务器的所在子网，用户可以通过添加不同的云服务器到不同可用区的子网下进行跨可用区容灾。

创建集群支持什么类型的机型？

支持所有按量计费的系统盘是云盘的机型。

当前容器服务宿主机支持什么操作系统？

当前支持 Ubuntu 16.04。如有其他操作系统需求可提工单联系我们。

扩展云服务器常见问题

扩展云服务器有什么限制？

只能选择当前集群所在的地域，但可以选择不同的可用区，允许集群跨可用区部署。

云服务器的数量有限制么？

有，用户所有按量计费云服务器不能超过用户配额。

销毁云服务器常见问题

销毁云服务器后，该主机下部署的容器怎么办？

销毁云服务器时，该主机下的容器等资源也会随之销毁。若某服务的容器数量小于期望运行的容器数量时，集群将会在其他主机上启动容器，直到运行的容器数量等于期望运行容器数量为止。

扩容缩容相关

Cluster Autoscaler 与基于监控指标的弹性伸缩的节点扩缩容有什么不同？

Cluster Autoscaler 确保集群中的所有 Pod 都可调度，不管具体的负载。而基于监控指标的节点弹性伸缩在自动扩缩时不关心 Pod，可能会添加一个没有任何 Pod 的节点，或者删除一个有一些系统关键 Pod 的节点，例如 kube-dns。Kubernetes 不鼓励这种自动缩容机制，故 Cluster Autoscaler 与基于监控指标的弹性伸缩的节点互相冲突，请不要同时启用。

CA 和伸缩组的对应关系是什么？

启用 CA 的集群会根据选择的节点配置，创建一个启动配置和绑定此启动配置的伸缩组。绑定后，将会在此伸缩组内进行扩缩容，扩容后的 CVM 自动加入集群。自动扩缩容的节点都是按量计费的。

容器服务控制台手动添加的节点是否会 CA 缩容？

不会，CA 缩容的节点只限于伸缩组内的节点。在【容器服务控制台】添加的节点不会加入到伸缩组中。

弹性伸缩控制台是否可以添加或者移出云主机？

不可以，不建议您在【弹性伸缩控制台】进行任何修改操作。

扩缩容会继承所选节点的哪些配置？

创建伸缩组时，需要选择集群内的一个节点作为参考来创建启动配置，参考的节点配置包括：

- vCPU
- 内存
- 系统盘大小
- 数据盘大小
- 磁盘类型
- 带宽
- 带宽计费模式
- 是否分配公网IP
- 安全组
- 私有网络
- 子网

如何使用多个伸缩组？

根据服务的重要级别、类型等特点，您可以通过创建多个伸缩组，为伸缩组设置不同的 label，从而指定伸缩组扩容出节点的 label，来对服务进行分类。

扩缩容最大值可以设置为多少？

目前用户每个可用区均有30个按量计费类型 CVM 配额，如果希望伸缩组有超过30台按量计费的 CVM，请提交工单申请。
具体配额请参见您当前可用区的云服务器实例数及配额。另外弹性伸缩也有最大值的限制，其最大值为200。如果弹性伸缩超过最大值，请提交工单申请。

集群启用缩容是否安全？

由于在缩容节点时会发生 Pod 重新调度的情况，所以服务必须可以容忍重新调度和短时的中断时再启用缩容。建议您为您的服务设置 PDB。PDB 可以在任何时候指定一个处于运行状态的 Pod 集合副本的最小数量或者最小百分比。有了 PodDisruptionBudget，应用部署者可以保证同一时间内主动移除 Pod 的集群操作不会销毁过多 Pod，避免了因销毁过多 Pod 导致数据丢失、服务中断或者无法接受的服务降级等影响。

节点上有哪些类型的 Pod 时不会被缩容？

- 当您设置了严格的 PodDisruptionBudget 的 Pod 不满足 PDB 时，不会缩容。
- Kube-system 下的 Pod。
- 节点上有非 deployment，replica set，job，stateful set 等控制器创建的 Pod。
- Pod 有本地存储。
- Pod 不能被调度到其他节点上。

节点满足缩容条件后多长时间会触发缩容？

10分钟。

节点 Not Ready 后多长时间会触发缩容？

20分钟。

多长时间扫描一次是否需要扩缩容？

10秒。

需要多长时间才可以扩容出 CVM？

一般在10分钟内，相关弹性伸缩的说明文档请参见 弹性伸缩。

为什么有 Unschedulable 的 Pod，却未进行扩容？

请确认以下原因：

- Pod 的请求资源是否过大。
- 是否设置了 node selector。
- 伸缩组的最大值是否已经达到。
- 帐号余额是否充足（帐号余额不足，弹性伸缩无法扩容），以及配额不足等 其他原因。

如何防止 Cluster Autoscaler 缩容特定节点？

```
# 可以在节点的annotations中设置如下信息
kubectl annotate node <nodename> cluster-autoscaler.kubernetes.io/scale-down-disabled=true
```

扩缩容事件如何反馈给用户？

用户可在弹性伸缩控制台查询伸缩组的伸缩活动，也可查看 k8s 的事件。在以下三种资源上都会有对应的事件：

- kube-system/cluster-autoscaler-status config map
  - ScaledUpGroup - CA 触发扩容。
  - ScaleDownEmpty - CA 删除了一个没有运行 Pod 的节点。
  - ScaleDown - CA 缩容。
- node
  - ScaleDown - CA 缩容。
  - ScaleDownFailed - CA 缩容失败。
- pod
  - TriggeredScaleUp - CA 由于此 Pod 触发扩容。
  - NotTriggerScaleUp - CA 无法找到可扩容的伸缩组使得此 Pod 可调度。
  - ScaleDown - CA 尝试驱逐此 Pod 来缩容节点。

服务常见问题

创建服务常见问题

服务的名称为什么不能重复？

服务名称是当前集群下的服务的唯一标识，服务之间可以通过服务名称+访问端口的形式互相访问。

创建服务能否使用非TCloudFinanceZone或 dockerhub 镜像的第三方镜像？

您可以通过登录到主机执行 docker login 命令登录到第三方镜像仓库拉取。

使用外网服务的有什么前置条件？

确保集群内的云服务器拥有外网带宽，否则外网服务将创建失败。

内存限制，CPU 限制如何填写？

详情参考 容器服务资源限制说明。

创建服务时的特权级是什么意思？

开启该选项会使得容器内程序具有真正的 root 权限。在容器内程序需要进行高级系统操作时建议开启，如搭建 nfs 服务器。

更新服务容器数量常见问题

更新容器数量需注意哪些问题？

需确认 CPU、内存资源充足，否则容器将创建失败。

能否将容器数量设为0？

可以，通过将容器数量设置为0，保存服务配置并且释放资源占用。

更新服务配置常见问题

更新服务是否支持滚动更新？

更新服务支持滚动更新和快速更新两种方式。

删除服务常见问题

删除服务后服务创建的负载均衡会自动销毁么？

删除服务会将该服务下的所有容器和外网负载均衡一并销毁，请提前备份好数据。

服务运行常见问题

如何设置容器系统时间为某时区时间？

容器默认使用 UTC 时间，使用容器时经常碰到容器系统时间和当地时间有时差的问题，解决方法是在 dockerfile 中创建时区文件（以下为修改示例）。

```
RUN echo "Australia/Sydney" > /etc/timezone;
```

Dockerhub 部分镜像如 ubuntu、php 和 busybox 等在容器服务里运行异常怎么办？

运行异常是因为没有设置启动命令或者默认的启动命令为 bash，导致容器执行完启动程序就退出了。要使容器一直运行，容器里 PID 为1的进程必须是常驻进程，否则 PID 为1的进程一结束容器就退出了。对于部分镜像如 centos 等，可以使用 /bin/bash 作为运行命令， -c sleep 800000 作为运行参数来创建服务，在控制台填运行参数时 -c 和 sleep 800000 必须放在两行。
目前已知的使用默认参数启动不了服务的镜像包括这些：clearlinux、ros、mageia、amazonlinux、ubuntu、clojure、crux、gcc、photon、java、debian、oraclelinux、mono、bash、buildpack-deps、golang、sourcemage、swift、openjdk、centos、busybox、docker、alpine、ibmjava、php和python。

镜像仓库常见问题

开通镜像仓库常见问题

命名空间有什么作用？

命名空间是标识用户私人镜像的地址前缀。

镜像仓库的账户是什么？

默认是用户的TCloudFinanceZone账号。

开通时创建的密码忘记了怎么办？

可以通过控制台重置密码。

创建镜像常见问题

创建镜像有配额限制么？

默认配额是100个镜像，可以通过工单申请配额。

创建的镜像可以分享给其他用户么？

暂不提供该功能。

创建的镜像如何使用？

需要先上传可用的镜像版本，通过具体的镜像版本来创建服务。

删除镜像常见问题

如何删除镜像某一个版本？

直接在控制台指定删除某一个具体的版本。

在镜像列表删除镜像会删除该镜像的所有版本么？

是的，删除镜像时会删除该镜像的所有镜像版本，请提前备份好数据。

远程终端常见问题

容器里面没有 bash，怎么办？

如果发现没有 bash，您可以在命令行中输入您想执行的命令，屏幕会显示该命令的返回结果。您可以将命令行看做一个缺少自动补全等其他功能的精简版 bash。建议您先执行安装 bash 的命令，再执行后续操作。

为什么运行 apt-get 安装软件如此之慢？

如果您觉得安装太慢，可能因为机器访问国外软件源速度过慢的原因。

Ubuntu 16.04系统

对于系统为 Ubuntu 16.04的容器，您可以运行以下命令，将 apt 的源设置为TCloudFinanceZone的源服务器。即复制以下命令粘贴到终端执行。

```
cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial main restricted universe multiverse
deb http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-security main restricted universe multiver
se
deb http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-updates main restricted universe multiver
se
deb-src http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial main restricted universe multiverse
deb-src http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-security main restricted universe multi
verse
deb-src http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-updates main restricted universe mult
iverse
ENDOF
```

CentOS 7系统

对于系统为 CentOS 7的容器，您可以执行以下操作，直接修改源地址提高安装速度。

1. 将以下代码复制并粘贴至容器内运行：

```
cat << ENDOF > /etc/yum.repos.d/CentOS-Base.repo
[os]
name=Qcloud centos os - \$basearch
baseurl=http://mirrors.finance.cloud.tencent.com/centos1/\$releasever/os/\$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
[updates]
name=Qcloud centos updates - \$basearch
baseurl=http://mirrors.finance.cloud.tencent.com/centos1/\$releasever/updates/\$basearch/
enabled=1
```

```
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[centosplus]
#name=Qcloud centosplus - \$basearch
#baseurl=http://mirrors.finance.cloud.tencent.com/centos1/\$releasever/centosplus/\$basearch/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[cloud]
#name=Qcloud centos contrib - \$basearch
#baseurl=http://mirrors.finance.cloud.tencent.com/centos1/\$releasever/cloud/$basearch/opens
tack-kilo/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[cr]
#name=Qcloud centos cr - \$basearch
#baseurl=http://mirrors.finance.cloud.tencent.com/centos1/\$releasever/cr/\$basearch/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
[extras]
name=Qcloud centos extras - \$basearch
baseurl=http://mirrors.finance.cloud.tencent.com/centos1/\$releasever/extras/\$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[fasttrack]
#name=Qcloud centos fasttrack - \basearch
#baseurl=http://mirrors.finance.cloud.tencent.com/centos1/\$releasever/fasttrack/\$basearch/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
ENDOF
```

2. 执行以下命令，清空并重建 YUM 缓存。

```
yum clean all && yum clean metadata && yum clean dbcache && yum makecache
```

直接修改源地址为临时解决方案，当容器被重新调度后，您所作的修改将会失效，所以建议您在创建镜像时解决该问题。具体的操作方法如下：

修改创建容器镜像的 Dockerfile。

在 Dockerfile 的 RUN 字段中，根据系统的不同添加直接修改源地址。例如，在一个基于 Ubuntu 系统的镜像中，加入以下内容：

```
RUN cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial main restricted universe multiverse
deb http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-security main restricted universe multiverse
deb http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-updates main restricted universe multiverse
#deb http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-proposed main restricted universe multiverse
#deb http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-backports main restricted universe multiverse
deb-src http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial main restricted universe multiverse
deb-src http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-security main restricted universe multiverse
deb-src http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-updates main restricted universe multiverse
#deb-src http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-proposed main restricted universe multiverse
#deb-src http://mirrors.finance.cloud.tencent.com/ubuntu/ xenial-backports main restricted universe multiverse
ENDOF
```

对于 CentOS 系统的镜像类似。

当登录容器后，发现没有 vim，netstat 等工具，怎么办？

您可以通过 apt-get install vim， net-tools 等命令下载您所需要的工具（CentOS 下执行 yum install vim）。

为什么运行 apt-get install 命令，提示找不到工具？

您可以通过以下操作，安装软件。

1. 执行以下命令，升级软件列表。

   ```
   apt-get update
   ```

2. 执行以下命令， 安装软件（CentOS 下执行 yum updateinfo）。

   ```
   apt-get install
   ```

如果想在容器中使用自制的工具，怎么办？

您可以在进入远程终端页面后，单击右下方的文件助手，即可进行上传和下载操作。

如何拷贝现场文件，例如 dump 或者日志到本地分析？

您可以在进入远程终端页面后，单击右下方的文件助手，即可进行上传和下载操作。

为什么用不了文件上传到容器或者下载到本地功能？

可能因为您的容器镜像里没有安装 tar 程序，您可以通过 apt-get install vim， net-tools 等命令（CentOS 下执行 yum install vim）先安装 tar 程序再重试。

为什么之前安装的工具不见了？

可能因为 kubernetes 重新调度您的容器，调度过程中会拉取镜像生成新的容器，如果镜像里面没有您之前安装的工具，新的容器是不会包含这些工具的。建议您在制作镜像时，安装一些常用的排错工具。

怎么复制控制台里的文字？

只要选中您想复制的内容，即可复制被选中的文字。

怎么粘贴复制好的文字？

同时按下 Shift + Insert 即可。

为什么会断开链接？

可能因为您在TCloudFinanceZone其他页面对容器、云服务器进行操作更改了容器的状况，也有可能是长时间（3分钟）不进行任何操作，服务器断开了这个链接。

运行 top 命令等出现 TERM environment variable not set 的错误，怎么办？

执行命令 export TERM linux 即可。

为何进入绝对路径较长的目录后，bash 提示符只显示“<”和部分路径？

因为默认的 bash 提示符被设置为显示“用户名@主机名 当前目录”。 如果当前路径长于一定长度，bash 默认会显示“<”与路径的最后一部分。

事件常见问题

Back-off restarting failed docker container

说明：正在重启异常的 Docker 容器。
解决方法：检查镜像中执行的 Docker 进程是否异常退出，若镜像内并无一持续运行的进程，可在创建服务的页面中添加执行脚本。

fit failure on node: Insufficient cpu

说明：集群 CPU 不足。
解决方法：原因是节点无法提供足够的计算核心，请在服务页面修改 CPU 限制或者对集群进行扩容。

no nodes available to schedule pods

说明：集群资源不足。
解决方法：原因是没有足够的节点用于承载实例，请在服务页面修改服务的实例数量，修改实例数量或者 CPU 限制。

pod failed to fit in any node

说明：没有合适的节点可供实例使用。
解决方法：原因是服务配置了不合适的资源限制，导致没有合适的节点用于承载实例，请在服务页面修改服务的实例数量或者 CPU 限制。

Liveness probe failed

说明：容器健康检查失败
解决方法：检查镜像内容器进程是否正常，检查检测端口是否配置正确。

Error syncing pod, skipping

Error syncing pod, skipping failed to "StartContainer" for with CrashLoopBackOff: "Back-off 5m0s restarting failed container
说明：容器进程崩溃或退出。
解决方法：检查容器内是否有持续运行的前台进程，若有检查其是否有异常行为。
如果以上解决方法未能解决您的问题，请联系客服。

词汇表

服务

由多个相同配置的实例和访问这些实例的规则组成的微服务。

Ingress

Ingress 是用于将外部 HTTP(S) 流量路由到服务（Service）的规则集合。

Image Store

用于存放 Docker 镜像，Docker 镜像用于部署容器服务。

节点

一台已注册到集群内的云服务器。

集群

指容器运行所需云资源的集合，包含了若干台云服务器、负载均衡器等资源。

配置项

配置项是多个配置的集合，帮您管理不同环境和不同业务。

实例（Pod）

由相关的一个或多个容器构成一个实例，这些容器共享相同的存储和网络空间。

应用

由多个服务组成一个完整的应用程序，可以通过模板快速部署。

API文档
镜像仓库服务（tcr）
版本（2019-09-24）
API 概览

API版本

V3

个人版相关接口

接口名称	接口功能
BatchDeleteImagePersonal	个人版镜像仓库批量删除Tag
BatchDeleteRepositoryPersonal	批量删除个人版仓库
CreateApplicationTriggerPersonal	创建应用更新触发器(功能已下线)
CreateImageLifecyclePersonal	创建个人版镜像版本清理策略
CreateNamespacePersonal	创建个人版命名空间
CreateRepositoryPersonal	创建个人版镜像仓库
CreateUserPersonal	创建个人用户
DeleteImageLifecycleGlobalPersonal	删除个人版全局镜像版本自动清理策略
DeleteImageLifecyclePersonal	删除个人版镜像仓库Tag自动清理策略
DeleteImagePersonal	删除个人版仓库tag
DeleteNamespacePersonal	删除个人版命名空间
DeleteRepositoryPersonal	删除个人版镜像仓库
DescribeFavorRepositoryPersonal	查询个人收藏仓库(功能下线中)
DescribeImageFilterPersonal	查询个人版中与指定tag镜像内容相同的tag列表

接口名称	接口功能
DescribeImageLifecycleGlobalPersonal	获取个人版全局镜像版本自动清理策略
DescribeImageLifecyclePersonal	获取个人版仓库自动清理策略
DescribeImagePersonal	获取个人版镜像仓库tag列表
DescribeNamespacePersonal	查询个人版命名空间信息
DescribeRepositoryFilterPersonal	获取满足输入搜索条件的个人版镜像仓库
DescribeRepositoryOwnerPersonal	查询个人版所有仓库
DescribeRepositoryPersonal	查询个人版仓库信息
DescribeUserQuotaPersonal	查询个人用户配额
DuplicateImagePersonal	复制个人版仓库镜像版本
ManageImageLifecycleGlobalPersonal	设置个人版全局镜像版本自动清理策略
ModifyRepositoryAccessPersonal	更新个人版仓库访问属性
ModifyRepositoryInfoPersonal	更新个人版镜像仓库描述
ModifyUserPasswordPersonal	修改个人用户登录密码
ValidateNamespaceExistPersonal	验证个人版命名空间是否存在
ValidateRepositoryExistPersonal	验证个人版仓库是否存在

调用方式
接口签名v1

TCloudFinanceZone 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TCloudFinanceZone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，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐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。以下是生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

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

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥Id	AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	shjr
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

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' : 'shjr',
    'SecretId' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE',
    '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=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&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.finance.cloud.tencent.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.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&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 = 'Gu5t9xGARNpq86cd98joQYCN3EXAMPLE';
$srcStr = 'GETcvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&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:

```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

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 EliP9YW3pW28FpsEdkXt/+WcGeI= ，最终得到的签名串请求参数（Signature）为：EliP9YW3pW28FpsEdkXt%2f%2bWcGeI%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 http 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TCloudFinanceZone 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
- Node

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.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Signature=EliP9YW3pW28FpsEdkXt%2F%2BWcGeI%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，即使是旧版的 API ，由于存在细节差异也会导致签名计算错误，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 CloudAPIDemo {
    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.finance.cloud.tencent.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.finance.cloud.tencent.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)).app
end("&");
        }
        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.Intege
r.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
        params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"); // 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", "shjr"); // 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
        params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE
", "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 time

import requests

secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

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.finance.cloud.tencent.com"
    data = {
        'Action' : 'DescribeInstances',
        'InstanceIds.0' : 'ins-09dx96dg',
        'Limit' : 20,
        'Nonce' : 11886,
        'Offset' : 0,
        'Region' : 'shjr',
        '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)
```

接口签名v3

TCloudFinanceZone 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TCloudFinanceZone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，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

2. TC3-HMAC-SHA256 签名方法

注意：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 格式默认所有业务接口均支持，multipart 格式只有特定业务接口支持，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。

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。我们仅用到了查询实例列表的两个参数：Limit 和 Offset，使用 GET 方法调用。

假设用户的 SecretId 和 SecretKey 分别是：AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE 和 Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

2.1. 拼接规范请求串

按如下格式拼接规范请求串（CanonicalRequest）：

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

- HTTPRequestMethod：HTTP 请求方法（GET、POST ），本示例中为 GET；
- 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，本示例取值为：Limit=10&Offset=0。注意：CanonicalQueryString 需要经过 URL 编码。
- CanonicalHeaders：参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入自定义的头部参与签名以提高自身请求的唯一性和安全性。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）的字典排序进行拼接。此例中为： content-type:application/x-www-form-urlencoded\nhost:cvm.finance.cloud.tencent.com\n
- 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）按照字典排序进行拼接，并且以分号（;）分隔。此例中为： content-type;host
- HashedRequestPayload：请求正文的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，对 HTTP 请求整个正文 payload 做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。注意：对于 GET 请求，RequestPayload 固定为空字符串，对于 POST 请求，RequestPayload 即为 HTTP 请求正文 payload。

根据以上规则，示例中得到的规范请求串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
GET
/
Limit=10&Offset=0
content-type:application/x-www-form-urlencoded
host:cvm.finance.cloud.tencent.com

content-type;host
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

2.2. 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
```

- 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 X-TC-Timestamp 取值，如上示例请求为 1539084154；
- 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，例如 cvm。如上示例请求，取值为 2018-10-09/cvm/tc3_request；
- HashedCanonicalRequest：前述步骤拼接所得规范请求串的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。

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，可能导致运行一段时间后，请求必定失败（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（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
TC3-HMAC-SHA256
1539084154
2018-10-09/cvm/tc3_request
91c9c192c14460df6c1ffc69e34e6c5e90708de2a6d282cccf957dbf1aa7f3a7
```

2.3. 计算签名

1）计算派生签名密钥，伪代码如下

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

- SecretKey：原始的 SecretKey；
- Date：即 Credential 中的 Date 字段信息，如上示例，为2018-10-09；
- Service：即 Credential 中的 Service 字段信息，如上示例，为 cvm；

2）计算签名，伪代码如下

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

- SecretSigning：即以上计算得到的派生签名密钥；
- StringToSign：即步骤2计算得到的待签名字符串；

2.4. 拼接 Authorization

按如下格式拼接 Authorization：

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

- Algorithm：签名方法，固定为 TC3-HMAC-SHA256；
- SecretId：密钥对中的 SecretId；
- CredentialScope：见上文，凭证范围；
- SignedHeaders：见上文，参与签名的头部信息；
- Signature：签名值

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
```

最终完整的调用信息如下：

```
https://cvm.finance.cloud.tencent.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.finance.cloud.tencent.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: shjr
```

3. 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）

4. 签名演示

Java

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.bind.DatatypeConverter;

import org.apache.commons.codec.digest.DigestUtils;

public class CloudAPITC3Demo {
    private final static String CHARSET = "UTF-8";
    private final static String ENDPOINT = "cvm.finance.cloud.tencent.com";
    private final static String PATH = "/";
    private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE";
    private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE";
    private final static String CT_X_WWW_FORM_URLENCODED = "application/x-www-form-urlencode
d";
    private final static String CT_JSON = "application/json";
```

```
    private final static String CT_FORM_DATA = "multipart/form-data";

    public static byte[] sign256(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(CHARSET));
    }

    public static void main(String[] args) throws Exception {
        String service = "cvm";
        String host = "cvm.finance.cloud.tencent.com";
        String region = "shjr";
        String action = "DescribeInstances";
        String version = "2017-03-12";
        String algorithm = "TC3-HMAC-SHA256";
        String timestamp = "1539084154";
        //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 = "GET";
        String canonicalUri = "/";
        String canonicalQueryString = "Limit=10&Offset=0";
        String canonicalHeaders = "content-type:application/x-www-form-urlencoded\n" + "host:" + host
+ "\n";
        String signedHeaders = "content-type;host";
        String hashedRequestPayload = DigestUtils.sha256Hex("");
        String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryStri
ng + "\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 = DigestUtils.sha256Hex(canonicalRequest.getBytes(CHARSET));
        String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCan
onicalRequest;
        System.out.println(stringToSign);

        // ************* 步骤 3：计算签名 *************
        byte[] secretDate = sign256(("TC3" + SECRET_KEY).getBytes(CHARSET), date);
        byte[] secretService = sign256(secretDate, service);
        byte[] secretSigning = sign256(secretService, "tc3_request");
```

```
        String signature = DatatypeConverter.printHexBinary(sign256(secretSigning, stringToSign)).toLow
erCase();
        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("Host", host);
        headers.put("Content-Type", CT_X_WWW_FORM_URLENCODED);
        headers.put("X-TC-Action", action);
        headers.put("X-TC-Timestamp", timestamp);
        headers.put("X-TC-Version", version);
        headers.put("X-TC-Region", region);
    }
}
```

Python

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

# 密钥参数
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

service = "cvm"
host = "cvm.finance.cloud.tencent.com"
endpoint = "https://" + host
region = "shjr"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
timestamp = 1539084154
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 10, "Offset": 0}

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = "GET"
canonical_uri = "/"
canonical_querystring = "Limit=10&Offset=0"
ct = "x-www-form-urlencoded"
```

```
payload = ""
if http_request_method == "POST":
    canonical_querystring = ""
    ct = "json"
    payload = json.dumps(params)
canonical_headers = "content-type:application/%s\nhost:%s\n" % (ct, host)
signed_headers = "content-type;host"
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)

# 公共参数添加到请求头部
headers = {
    "Authorization": authorization,
    "Host": host,
    "Content-Type": "application/%s" % ct,
```

```
    "X-TC-Action": action,
    "X-TC-Timestamp": str(timestamp),
    "X-TC-Version": version,
    "X-TC-Region": region,
}
```

请求结构

1. 服务地址

地域（Region）是指物理的数据中心的地理区域。TCloudFinanceZone交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 查询地域列表 查看完成的地域列表。

2. 通信协议

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

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

4. 字符编码

均使用UTF-8编码。

返回结果

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，可以联系我们，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 cor
rect."
        },
        "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，可以联系我们，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，下表列出了公共错误码。

错误码	错误描述
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。
AuthFailure.SignatureExpire	签名过期。
AuthFailure.SignatureFailure	签名错误。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未 CAM 授权。
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。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。

错误码	错误描述
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
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。

公共参数

公共参数是用于标识用户和接口鉴权目的的参数，如非必要，在每个接口单独的接口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。

签名方法 v3

使用 TC3-HMAC-SHA256 签名方法时，公共参数需要统一放到 HTTP Header 请求头部中，如下：

参数名称	类型	必选	描述
X-TC-Action	String	是	操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
X-TC-Region	String	是	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。
X-TC-Timestamp	Integer	是	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。
X-TC-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。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/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，AKIDEXAMPLE 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为产品名，必须与调用的产品域名一致，例如cvm； - 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。
X-TC-Token	String	否	临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token。

签名方法 v1

使用 HmacSHA1 和 HmacSHA256 签名方法时，公共参数需要统一放到请求串中，如下

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	是	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。

参数名称	类型	必选	描述
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。具体计算方法参见接口鉴权文档。
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	临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token 。

地域列表

地域（Region）是指物理的数据中心的地理区域。TCloudFinanceZone交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 查询地域列表 查看完成的地域列表。

个人版相关接口
个人版镜像仓库批量删除Tag

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版镜像仓库中批量删除Tag

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 13:48:26。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：BatchDeleteImagePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：
Tags	是	否	Array of String	Tag列表 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrNoRepo	仓库不存在
MissingParameter	缺少参数错误。
InternalError	内部错误。

批量删除个人版仓库

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于个人版镜像仓库中批量删除镜像仓库

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:36:21。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：BatchDeleteRepositoryPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoNames	是	否	Array of String	仓库名称数组 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrUnauthorized	鉴权失败。
MissingParameter	缺少参数错误。
InternalError	内部错误。

创建应用更新触发器(功能已下线)

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于创建应用更新触发器(功能已下线)

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 13:54:25。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateApplicationTriggerPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	触发器关联的镜像仓库，library/test格式 示例值：
TriggerName	是	否	String	触发器名称 示例值：
InvokeMethod	是	否	String	触发方式，"all"全部触发，"taglist"指定tag触发，"regex"正则触发 示例值：
InvokeExpr	否	否	String	触发方式对应的表达式 示例值：
ClusterId	是	否	String	应用所在TKE集群ID 示例值：
Namespace	是	否	String	应用所在TKE集群命名空间 示例值：

参数名称	必选	允许NULL	类型	描述
WorkloadType	是	否	String	应用所在TKE集群工作负载类型,支持Deployment、StatefulSet、DaemonSet、CronJob、Job。 示例值：
WorkloadName	是	否	String	应用所在TKE集群工作负载名称 示例值：
ContainerName	是	否	String	应用所在TKE集群工作负载下容器名称 示例值：
ClusterRegion	是	否	Int64	应用所在TKE集群地域 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrNoRepo	仓库不存在
ResourceNotFound.ErrNoUser	用户不存在（未注册）
MissingParameter	缺少参数错误。
InternalError	内部错误。
LimitExceeded.ErrTriggerMaxLimit	触发器达到配额
InvalidParameter.ErrTriggerExist	触发器名称已存在

创建个人版镜像版本清理策略

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版中创建清理策略

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 13:50:39。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateImageLifecyclePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：
Type	是	否	String	keep_last_days:保留最近几天的数据;keep_last_nums:保留最近多少个 示例值：
Val	是	否	Int64	策略值 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrUnauthorized	鉴权失败。
ResourceNotFound.ErrNoRepo	仓库不存在
MissingParameter	缺少参数错误。
InternalError	内部错误。

创建个人版命名空间

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

创建个人版镜像仓库命名空间，此命名空间全局唯一

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:30:55。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateNamespacePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
Namespace	是	否	String	命名空间名称 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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

错误码	描述
LimitExceeded.ErrNamespaceMaxLimit	用户命名空间达到配额
ResourceNotFound.ErrNoUser	用户不存在（未注册）
InvalidParameter.ErrNamespaceExist	命名空间名称已经存在
InvalidParameter.ErrNamespaceReserved	命名空间已被占用
MissingParameter	缺少参数错误。
InternalError	内部错误。

创建个人版镜像仓库

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版仓库中创建镜像仓库

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:33:42。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateRepositoryPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：
Public	否	否	Uint64	是否公共,1:公共,0:私有 示例值：
Description	否	否	String	仓库描述 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrNSMisMatch	用户请求中的信息与其namespace不匹配
MissingParameter	缺少参数错误。
InternalError	内部错误。
InvalidParameter.ErrRepoExist	无效的参数，仓库已存在。

创建个人用户

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

创建个人用户

默认接口请求频率限制：20次/秒。

接口更新时间：2020-02-20 16:21:46。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateUserPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
Password	是	否	String	用户密码 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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

错误码	描述
MissingParameter.MissingParameter	缺少参数
InvalidParameter.ErrUserExist	用户已经存在
InternalError	内部错误。

删除个人版全局镜像版本自动清理策略

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于删除个人版全局镜像版本自动清理策略

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:02:43。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteImageLifecycleGlobalPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误。

删除个人版镜像仓库Tag自动清理策略

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版镜像仓库中删除仓库Tag自动清理策略

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 13:51:23。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteImageLifecyclePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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

错误码	描述
ResourceNotFound.ErrNoRepo	仓库不存在
MissingParameter	缺少参数错误。
InternalError	内部错误。

删除个人版仓库tag

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版中删除tag

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 13:48:38。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteImagePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：
Tag	是	否	String	Tag名 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrUnauthorized	鉴权失败。
ResourceNotFound.ErrNoRepo	仓库不存在
MissingParameter	缺少参数错误。
InternalError	内部错误。

删除个人版命名空间

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

删除共享版命名空间

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:32:45。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteNamespacePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
Namespace	是	否	String	命名空间名称 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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

错误码	描述
InternalError.ErrUnauthorized	鉴权失败。
ResourceNotFound.ErrNoNamespace	用户没有创建命名空间
ResourceNotFound.ErrNoUser	用户不存在（未注册）
MissingParameter	缺少参数错误。
InternalError	内部错误。

删除个人版镜像仓库

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于个人版镜像仓库中删除

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:35:35。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteRepositoryPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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

错误码	描述
InternalError.ErrUnauthorized	鉴权失败。
ResourceNotFound.ErrNoRepo	仓库不存在
MissingParameter	缺少参数错误。
InternalError	内部错误。

查询个人收藏仓库(功能下线中)

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

查询个人收藏仓库(功能下线中)

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 14:09:21。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeFavorRepositoryPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：
Limit	是	否	Int64	分页Limit 示例值：
Offset	是	否	Int64	Offset用于分页 示例值：

3. 输出参数

参数名称	类型	描述
Data	FavorResp	个人收藏仓库列表返回信息 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
MissingParameter	缺少参数错误。
InternalError	内部错误。

查询个人版中与指定tag镜像内容相同的tag列表

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版中查询与指定tag镜像内容相同的tag列表

默认接口请求频率限制：20次/秒。

接口更新时间：2020-02-20 16:47:49。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeImageFilterPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：
Tag	是	否	String	Tag名 示例值：

3. 输出参数

参数名称	类型	描述
Data	SameImagesResp	payload 示例值：查看

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrUnauthorized	鉴权失败。
ResourceNotFound.ErrNoRepo	仓库不存在
MissingParameter	缺少参数错误。
InternalError	内部错误。

获取个人版全局镜像版本自动清理策略

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于获取个人版全局镜像版本自动清理策略

默认接口请求频率限制：20次/秒。

接口更新时间：2020-03-11 14:26:22。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeImageLifecycleGlobalPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。

3. 输出参数

参数名称	类型	描述
Data	AutoDelStrategyInfoResp	全局自动删除策略信息 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误。

获取个人版仓库自动清理策略

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于获取个人版仓库中自动清理策略

默认接口请求频率限制：20次/秒。

接口更新时间：2020-02-27 11:58:01。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeImageLifecyclePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：

3. 输出参数

参数名称	类型	描述
Data	AutoDelStrategyInfoResp	自动删除策略信息 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
MissingParameter	缺少参数错误。
InternalError	内部错误。

获取个人版镜像仓库tag列表

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于获取个人版镜像仓库tag列表

默认接口请求频率限制：20次/秒。

接口更新时间：2020-02-22 22:31:42。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeImagePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：
Offset	否	否	Int64	偏移量，默认为0 示例值：
Limit	否	否	Int64	返回最大数量，默认 20, 最大值 100 示例值：
Tag	否	否	String	tag名称，可根据输入搜索 示例值：

3. 输出参数

参数名称	类型	描述
Data	TagInfoResp	镜像tag信息

参数名称	类型	描述
		示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrUnauthorized	鉴权失败。
ResourceNotFound.ErrNoRepo	仓库不存在
MissingParameter	缺少参数错误。
InternalError	内部错误。

查询个人版命名空间信息

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

查询个人版命名空间信息

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 13:52:54。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeNamespacePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
Namespace	是	否	String	命名空间，支持模糊查询 示例值：
Limit	是	否	Int64	单页数量 示例值：
Offset	是	否	Int64	偏移量 示例值：

3. 输出参数

参数名称	类型	描述
Data	NamespaceInfoResp	用户命名空间返回信息 示例值：查看

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrNoUser	用户不存在（未注册）
MissingParameter	缺少参数错误。
InternalError	内部错误。

获取满足输入搜索条件的个人版镜像仓库

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版镜像仓库中，获取满足输入搜索条件的用户镜像仓库

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:06:45。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeRepositoryFilterPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	否	否	String	搜索镜像名 示例值：
Offset	否	否	Int64	偏移量，默认为0 示例值：
Limit	否	否	Int64	返回最大数量，默认 20，最大100 示例值：
Public	否	否	Int64	筛选条件：1表示public，0表示private 示例值：
Namespace	否	否	String	命名空间 示例值：

3. 输出参数

参数名称	类型	描述
Data	SearchUserRepositoryResp	仓库信息 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
MissingParameter	缺少参数错误。
InternalError	内部错误。

查询个人版所有仓库

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版中获取用户全部的镜像仓库列表

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:14:46。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeRepositoryOwnerPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
Offset	否	否	Int64	偏移量，默认为0 示例值：
Limit	否	否	Int64	返回最大数量，默认 20, 最大值 100 示例值：
RepoName	否	否	String	仓库名称 示例值：

3. 输出参数

参数名称	类型	描述
Data	RepoInfoResp	仓库信息 示例值：查看

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
MissingParameter	缺少参数错误。
InternalError	内部错误。

查询个人版仓库信息

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

查询个人版仓库信息

默认接口请求频率限制：20次/秒。

接口更新时间：2020-02-20 16:57:40。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeRepositoryPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名字 示例值：

3. 输出参数

参数名称	类型	描述
Data	RepositoryInfoResp	仓库信息 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrNoRepo	仓库不存在
MissingParameter	缺少参数错误。
InternalError	内部错误。

查询个人用户配额

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

查询个人用户配额

默认接口请求频率限制：20次/秒。

接口更新时间：2020-02-20 16:24:18。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeUserQuotaPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。

3. 输出参数

参数名称	类型	描述
Data	RespLimit	配额返回信息 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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

错误码	描述
InternalError	内部错误。

复制个人版仓库镜像版本

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版镜像仓库中复制镜像版本

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:16:31。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DuplicateImagePersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
SrcImage	是	否	String	源镜像名称，不包含domain。例如： cloudyun/foo:v1 示例值：
DestImage	是	否	String	目的镜像名称，不包含domain。例如： cloudyun/foo:latest 示例值：

3. 输出参数

参数名称	类型	描述
Data	DupImageTagResp	复制镜像返回值 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrNoTag	tag不存在
InvalidParameter.ErrNSMisMatch	用户请求中的信息与其namespace不匹配
MissingParameter	缺少参数错误。
InternalError	内部错误。

设置个人版全局镜像版本自动清理策略

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于设置个人版全局镜像版本自动清理策略

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:01:04。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ManageImageLifecycleGlobalPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
Type	是	否	String	global_keep_last_days:全局保留最近几天的数据;global_keep_last_nums:全局保留最近多少个 示例值：
Val	是	否	Int64	策略值 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
MissingParameter	缺少参数错误。

更新个人版仓库访问属性

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于更新个人版镜像仓库的访问属性

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:37:49。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyRepositoryAccessPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：
Public	是	否	Int64	默认值为0 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrNoRepo	仓库不存在
InvalidParameter.ErrNSMisMatch	用户请求中的信息与其namespace不匹配
MissingParameter	缺少参数错误。
InternalError	内部错误。

更新个人版镜像仓库描述

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于在个人版镜像仓库中更新容器镜像描述

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 11:44:54。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyRepositoryInfoPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：
Description	是	否	String	仓库描述 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrNoRepo	仓库不存在
InvalidParameter.ErrNSMisMatch	用户请求中的信息与其namespace不匹配
MissingParameter	缺少参数错误。
InternalError	内部错误。

修改个人用户登录密码

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

修改个人用户登录密码

默认接口请求频率限制：20次/秒。

接口更新时间：2023-10-28 18:28:18。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyUserPasswordPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
Password	是	否	String	更新后的密码 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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

错误码	描述
ResourceNotFound.ErrNoUser	用户不存在（未注册）
MissingParameter	缺少参数错误。
InternalError	内部错误。

验证个人版命名空间是否存在

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

查询个人版用户命名空间是否存在

默认接口请求频率限制：20次/秒。

接口更新时间：2020-02-20 16:59:44。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ValidateNamespaceExistPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
Namespace	是	否	String	命名空间名称 示例值：

3. 输出参数

参数名称	类型	描述
Data	NamespaceIsExistsResp	命名空间是否存在 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
MissingParameter	缺少参数错误。
InternalError	内部错误。

验证个人版仓库是否存在

1. 接口描述

接口请求域名： tcr.api3.finance.cloud.tencent.com。

用于判断个人版仓库是否存在

默认接口请求频率限制：20次/秒。

接口更新时间：2020-02-20 16:55:22。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ValidateRepositoryExistPersonal
Version	是	否	String	公共参数，本接口取值：2019-09-24
Region	是	否	String	公共参数，地域信息本接口不需要传递此参数。
RepoName	是	否	String	仓库名称 示例值：

3. 输出参数

参数名称	类型	描述
Data	RepoIsExistResp	仓库是否存在 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ErrNSMisMatch	用户请求中的信息与其namespace不匹配
MissingParameter	缺少参数错误。
InternalError	内部错误。

数据结构

AuthUser

构建用户信息

被如下接口引用：DescribeSourceCodeAuthUserInfoPersonal

名称	必选	允许NULL	类型	描述
Name	是	否	String	用户名 示例值：
Email	是	否	String	用户邮箱 示例值：

RepoIsExistResp

仓库是否存在的返回值

被如下接口引用：ValidateRepositoryExistPersonal

名称	必选	允许NULL	类型	描述
IsExist	是	否	Bool	仓库是否存在 示例值：

RespDockerHubRepoList

用于DokcerHub 仓库列表信息的返回信息

被如下接口引用：DescribeDockerHubRepositoryPersonal

名称	必选	允许NULL	类型	描述
TotalCount	是	否	Int64	仓库总数 示例值：
RepoInfo	是	否	Array of HubSimpleInfo	仓库信息列表 示例值：查看

TriggerInvokeResult

触发器触发结果

被如下接口引用：DescribeApplicationTriggerLogPersonal

名称	必选	允许NULL	类型	描述
ReturnCode	是	是	Int64	请求TKE返回值 示例值：
ReturnMsg	是	是	String	请求TKE返回信息 示例值：

HubSimpleInfo

Hub的信息描述

被如下接口引用：DescribeDockerHubRepositoryPersonal

名称	必选	允许NULL	类型	描述
Reponame	是	否	String	仓库名称 示例值：
Repotype	是	否	String	仓库类型 示例值：
Logo	是	否	String	仓库Logo 示例值：
SimpleDesc	是	否	String	仓库简述 示例值：
IsUserFavor	是	否	Bool	是否为收藏 示例值：
FavorCount	是	否	Int64	收藏数量 示例值：

RespLimit

用户配额返回值

被如下接口引用：DescribeUserQuotaPersonal

名称	必选	允许NULL	类型	描述
LimitInfo	是	否	Array of Limit	配额信息 示例值：查看

FavorResp

用于获取收藏仓库的响应

被如下接口引用：DescribeFavorRepositoryPersonal

名称	必选	允许NULL	类型	描述
TotalCount	是	否	Int64	收藏仓库的总数 示例值：
RepoInfo	是	是	Array of Favors	仓库信息数组 示例值：查看

DescribeApplicationTokenPersonalResp

镜像仓库凭证

被如下接口引用：DescribeApplicationTokenPersonal

名称	必选	允许NULL	类型	描述
ApplicationToken	是	否	String	镜像仓库凭证 示例值：

BuildInfo

构建信息

被如下接口引用：DescribeImageBuildTaskLogInfoPersonal、DescribeImageBuildTaskLogPersonal

名称	必选	允许NULL	类型	描述
Id	是	否	Int64	Id 示例值：

名称	必选	允许NULL	类型	描述
AppId	是	否	Uint64	AppId 示例值：
BuildType	是	否	String	构建类型 示例值：
BuildManually	是	否	Int64	是否手动构建 示例值：
BuildWorkDir	是	否	String	构建工作目录 示例值：
Args	是	是	String	构建参数 示例值：
Status	是	否	String	构建状态 示例值：
StartTime	是	否	Datetime_iso	构建开始时间 示例值：
EndTime	是	否	Datetime_iso	构建结束时间 示例值：
GitServer	是	否	String	构建仓库地址 示例值：
Group	是	否	String	repo所在的group 示例值：
Repo	是	否	String	构建所在的repo 示例值：
RepoUrl	是	是	String	Repo url地址 示例值：
Owner	是	否	String	用户在git服务器上的用户名 示例值：
Branch	是	否	String	构建的分支 示例值：
DockerfilePath	是	否	String	dockerfile在仓库中的路径 示例值：
RegistryNamespace	是	否	String	registry中的namespace 示例值：

名称	必选	允许NULL	类型	描述
RegistryUsername	是	否	String	用户在registry中的用户名 示例值：
Image	是	否	String	镜像名称，不包含tag 示例值：
ForceImage	是	否	String	镜像名 示例值：
CommitSHA	是	否	String	提交的SHA 示例值：
CommitAuthor	是	否	String	提交的作者 示例值：
CommitMessage	是	否	String	提交的信息 示例值：
CommitTime	是	否	Datetime_iso	提交的时间 示例值：
BuildLog	是	否	String	构建日志 示例值：

DesGitAuthRsp

查询GitAuth返回值

被如下接口引用：DescribeSourceCodeAuthPersonal

名称	必选	允许NULL	类型	描述
AuthMap	是	否	String	用户GitAuth信息 示例值：

UserIsExistsResp

用户是否存在

被如下接口引用：ValidateUserPersonal

名称	必选	允许NULL	类型	描述

名称	必选	允许NULL	类型	描述
IsExist	是	否	Bool	用户是否存在 示例值：
MainIsExist	是	否	Bool	主账号是否存在 示例值：

AutoDelStrategyInfoResp

获取自动删除策略

被如下接口引用：DescribeImageLifecycleGlobalPersonal、DescribeImageLifecyclePersonal

名称	必选	允许NULL	类型	描述
TotalCount	是	否	Int64	总数目 示例值：
StrategyInfo	是	是	Array of AutoDelStrategyInfo	自动删除策略列表 示例值：查看

DupImageTagResp

复制镜像tag返回值

被如下接口引用：DuplicateImagePersonal

名称	必选	允许NULL	类型	描述
Digest	是	否	String	镜像Digest值 示例值：

SameImagesResp

指定tag镜像内容相同的tag列表

被如下接口引用：DescribeImageFilterPersonal

名称	必选	允许NULL	类型	描述

名称	必选	允许NULL	类型	描述
SameImages	是	是	Array of String	tag列表 示例值：

DockerHubTagList

用于返回dockerhub的tag列表

被如下接口引用：DescribeDockerHubImagePersonal

名称	必选	允许NULL	类型	描述
Reponame	是	是	String	DockerHub的仓库名称 示例值：
TagList	是	是	Array of String	Tag的列表 示例值：

TagInfoResp

Tag列表的返回值

被如下接口引用：DescribeImagePersonal

名称	必选	允许NULL	类型	描述
TagCount	是	否	Int64	Tag的总数 示例值：
TagInfo	是	否	Array of TagInfo	TagInfo列表 示例值：查看
Server	是	否	String	Server 示例值：
RepoName	是	否	String	仓库名称 示例值：

TagInfo

镜像tag信息

被如下接口引用：DescribeImagePersonal

名称	必选	允许NULL	类型	描述
TagName	是	否	String	Tag名称 示例值：
TagId	是	否	String	镜像Id 示例值：
ImageId	是	否	String	docker image 可以看到的id 示例值：
Size	是	否	String	大小 示例值：
CreationTime	是	否	String	镜像的创建时间 示例值：
DurationDays	是	是	String	镜像创建至今时间长度 示例值：
Author	是	否	String	镜像的作者 示例值：
Architecture	是	否	String	次镜像建议运行的系统架构 示例值：
DockerVersion	是	否	String	创建此镜像的docker版本 示例值：
OS	是	否	String	此镜像建议运行系统 示例值：
SizeByte	是	否	Int64	SizeByte 示例值：
Id	是	否	Int64	Id 示例值：
UpdateTime	是	否	String	数据更新时间 示例值：
PushTime	是	否	String	镜像更新时间 示例值：

BuildRule

构建规则

被如下接口引用：DescribeImageBuildPersonal

名称	必选	允许NULL	类型	描述
RegistryNamespace	是	否	String	registry中的namespace 示例值：
RegistryUsername	是	否	String	用户在registry中的用户名 示例值：
ImageName	是	否	String	镜像名称，不包含tag 示例值：
ImageTagFormat	是	否	String	镜像tag的格式 示例值：
GitServer	是	否	String	源码所在的git服务 示例值：
Group	是	否	String	repo所在的group 示例值：
Repo	是	否	String	源码在git服务器上的仓库名 示例值：
Owner	是	否	String	用户在git服务器上的用户名 示例值：
Branches	是	是	Array of String	分支 示例值：
Tag	是	是	Int64	Tag 示例值：
DockerfilePath	是	是	String	dockerfile在仓库中的路径 示例值：
BuildWorkDir	是	是	String	工作目录 示例值：
ForceTag	是	是	String	构建出的镜像覆盖该Tag 示例值：
BuildArgs	是	是	String	Args 示例值：

Favors

仓库收藏

被如下接口引用：DescribeFavorRepositoryPersonal

名称	必选	允许NULL	类型	描述
RepoName	是	否	String	仓库名字 示例值：
RepoType	是	否	String	仓库类型 示例值：
PullCount	是	是	Int64	Pull总共的次数 示例值：
FavorCount	是	是	Int64	仓库收藏次数 示例值：
Public	是	是	Int64	仓库是否公开 示例值：
IsQcloudOfficial	是	是	Bool	是否为官方所有 示例值：
TagCount	是	是	Int64	仓库Tag的数量 示例值：
Logo	是	是	String	Logo 示例值：
Region	是	否	String	地域 示例值：
RegionId	是	否	Int64	地域的Id 示例值：

RequestFavor

请求的入参，用于收藏镜像仓库

被如下接口引用：BatchDeleteFavorRepositoryPersonal

名称	必选	允许NULL	类型	描述
RepoName	是	否	String	收藏的镜像仓库名称

名称	必选	允许NULL	类型	描述
				示例值：
RepoType	是	否	String	收藏的镜像仓库类型 示例值：
RegionId	是	否	String	地域Id 示例值：

RepoInfoResp

仓库信息的返回信息

被如下接口引用：DescribeRepositoryAllPersonal、DescribeRepositoryOwnerPersonal

名称	必选	允许NULL	类型	描述
TotalCount	是	否	Int64	仓库总数 示例值：
RepoInfo	是	否	Array of RepoInfo	仓库信息列表 示例值：查看
Server	是	否	String	Server信息 示例值：

TriggerResp

触发器返回值

被如下接口引用：DescribeApplicationTriggerPersonal

名称	必选	允许NULL	类型	描述
TriggerName	是	是	String	触发器名称 示例值：
InvokeSource	是	是	String	触发来源 示例值：
InvokeAction	是	是	String	触发动作 示例值：

名称	必选	允许NULL	类型	描述
CreateTime	是	是	String	创建时间 示例值：
UpdateTime	是	是	String	更新时间 示例值：
InvokeCondition	是	是	TriggerInvokeCondition	触发条件 示例值：查看
InvokePara	是	是	TriggerInvokePara	触发器参数 示例值：查看

AutoDelStrategyInfo

自动删除策略信息

被如下接口引用：DescribeImageLifecycleGlobalPersonal、DescribeImageLifecyclePersonal

名称	必选	允许NULL	类型	描述
Username	是	否	String	用户名 示例值：
RepoName	是	否	String	仓库名 示例值：
Type	是	否	String	类型 示例值：
Value	是	否	Int64	策略值 示例值：
Valid	是	否	Int64	Valid 示例值：
CreationTime	是	否	String	创建时间 示例值：

BuildRepo

源代码仓库信息

被如下接口引用：DescribeSourceCodeRepositoryPersonal

名称	必选	允许NULL	类型	描述
GitServer	是	否	String	源码所在的git服务 示例值：
Group	是	否	String	repo所在的group 示例值：
RepoName	是	否	String	源码在git服务器上的仓库名 示例值：
RepoId	是	否	Int64	仓库Id 示例值：
Desc	是	是	String	仓库描述 示例值：
Private	是	是	Bool	是否为私有 示例值：
WebUrl	是	是	String	WebUrl 示例值：

TriggerLogResp

触发器日志

被如下接口引用：DescribeApplicationTriggerLogPersonal

名称	必选	允许NULL	类型	描述
RepoName	是	是	String	仓库名称 示例值：
TagName	是	是	String	Tag名称 示例值：
TriggerName	是	是	String	触发器名称 示例值：
InvokeSource	是	是	String	触发方式 示例值：
InvokeAction	是	是	String	触发动作 示例值：

名称	必选	允许NULL	类型	描述
InvokeTime	是	是	String	触发时间 示例值：
InvokeCondition	是	是	TriggerInvokeCondition	触发条件 示例值：查看
InvokePara	是	是	TriggerInvokePara	触发参数 示例值：查看
InvokeResult	是	是	TriggerInvokeResult	触发结果 示例值：查看

TriggerInvokeCondition

触发器触发条件

被如下接口引用：DescribeApplicationTriggerLogPersonal、DescribeApplicationTriggerPersonal

名称	必选	允许NULL	类型	描述
InvokeMethod	是	否	String	触发方式 示例值：
InvokeExpr	是	是	String	触发表达式 示例值：

UserInfo

共享版用户信息

被如下接口引用：DescribeUserPersonal

名称	必选	允许NULL	类型	描述
Username	是	否	String	用户名 示例值：
AppId	是	否	Int64	AppId 示例值：
ApplicationToken	是	否	String	密码 示例值：

名称	必选	允许NULL	类型	描述
CreationTime	是	否	String	创建时间 示例值：

DockerHubRepoinfo

DockerHub仓库信息

被如下接口引用：DescribeDockerHubRepositoryInfoPersonal

名称	必选	允许NULL	类型	描述
Reponame	是	否	String	仓库名称 示例值：
Repotype	是	否	String	仓库类型 示例值：
Logo	是	否	String	仓库Logo 示例值：
SimpleDesc	是	否	String	简述 示例值：
DetailDesc	是	否	String	详述 示例值：
FavorCount	是	否	Int64	收藏次数 示例值：
IsUserFavor	是	否	Bool	是否用户的收藏 示例值：

DescribeApplicationTriggerLogPersonalResp

查询应用更新触发器触发日志返回值

被如下接口引用：DescribeApplicationTriggerLogPersonal

名称	必选	允许NULL	类型	描述
TotalCount	是	否	Int64	返回总数 示例值：

名称	必选	允许NULL	类型	描述
LogInfo	是	是	Array of TriggerLogResp	触发日志列表 示例值：查看

Region

地域信息

被如下接口引用：DescribeRegions

名称	必选	允许NULL	类型	描述
Alias	是	否	String	gz 示例值：
RegionId	是	否	Uint64	1 示例值：
RegionName	是	否	String	ap-xx 示例值：
Status	是	否	String	alluser 示例值：
Remark	是	否	String	remark 示例值：
CreatedAt	是	否	Datetime_iso	创建时间 示例值：
UpdatedAt	是	否	Datetime_iso	更新时间 示例值：
Id	是	否	Int64	id 示例值：

BuildBranchResp

构建分支返回信息

被如下接口引用：DescribeSourceCodeRepositoryBranchPersonal

名称	必选	允许NULL	类型	描述

名称	必选	允许NULL	类型	描述
Branches	是	否	Array of String	构建的分支信息 示例值：

NamespaceInfoResp

获取命名空间信息返回

被如下接口引用：DescribeNamespacePersonal

名称	必选	允许NULL	类型	描述
NamespaceCount	是	否	Int64	命名空间数量 示例值：
NamespaceInfo	是	否	Array of NamespaceInfo	命名空间信息 示例值：查看

NamespaceInfo

命名空间信息

被如下接口引用：DescribeNamespacePersonal

名称	必选	允许NULL	类型	描述
Namespace	是	否	String	命名空间 示例值：
CreationTime	是	否	String	创建时间 示例值：
RepoCount	是	否	Int64	命名空间下仓库数量 示例值：

Trigger

Trigger，用于ccb

被如下接口引用：CreateImageBuildPersonal、ModifyImageBuildPersonal

名称	必选	允许NULL	类型	描述
Branches	是	否	Array of String	分支 示例值：
Tag	是	否	Int64	Tag 示例值：

BuildHistoryResp

构建历史信息

被如下接口引用：DescribeImageBuildTaskLogInfoPersonal

名称	必选	允许NULL	类型	描述
BuildHistory	是	否	BuildInfo	构建信息 示例值：查看

BuildInfoResp

构建信息的返回信息

被如下接口引用：DescribeImageBuildTaskLogPersonal

名称	必选	允许NULL	类型	描述
TotalCount	是	否	Int64	总数 示例值：
BuildList	是	否	Array of BuildInfo	构建信息列表 示例值：查看

AuthUserInfoResp

构建用户的信息返回值

被如下接口引用：DescribeSourceCodeAuthUserInfoPersonal

名称	必选	允许NULL	类型	描述

名称	必选	允许NULL	类型	描述
User	是	否	AuthUser	构建用户信息 示例值：查看

Limit

共享镜像仓库用户配额

被如下接口引用：DescribeUserQuotaPersonal

名称	必选	允许NULL	类型	描述
Username	是	否	String	用户名 示例值：
Type	是	否	String	配额的类型 示例值：
Value	是	否	Int64	配置的值 示例值：

RepoInfo

仓库的信息

被如下接口引用：DescribeRepositoryAllPersonal、DescribeRepositoryFilterPersonal、DescribeRepositoryOwnerPersonal

名称	必选	允许NULL	类型	描述
RepoName	是	否	String	仓库名称 示例值：
RepoType	是	否	String	仓库类型 示例值：
TagCount	是	否	Int64	Tag数量 示例值：
Public	是	否	Int64	是否为公开 示例值：

名称	必选	允许NULL	类型	描述
IsUserFavor	是	否	Bool	是否为用户收藏 示例值：
IsQcloudOfficial	是	否	Bool	是否为云平台官方仓库 示例值：
FavorCount	是	否	Int64	被收藏的个数 示例值：
PullCount	是	否	Int64	拉取的数量 示例值：
Description	是	否	String	描述 示例值：
CreationTime	是	否	String	仓库创建时间 示例值：
UpdateTime	是	否	String	仓库更新时间 示例值：

SearchUserRepositoryResp

获取满足输入搜索条件的用户镜像仓库

被如下接口引用：DescribeRepositoryFilterPersonal

名称	必选	允许NULL	类型	描述
TotalCount	是	否	Int64	总个数 示例值：
RepoInfo	是	否	Array of RepoInfo	仓库列表 示例值：查看
Server	是	否	String	Server 示例值：
PrivilegeFiltered	是	否	Bool	PrivilegeFiltered 示例值：

BuildRuleResp

Build的信息返回值

被如下接口引用：DescribeImageBuildPersonal

名称	必选	允许NULL	类型	描述
BuildRule	是	否	BuildRule	BuildRule信息 示例值：查看

TriggerInvokePara

触发器触发参数

被如下接口引用：DescribeApplicationTriggerLogPersonal、DescribeApplicationTriggerPersonal

名称	必选	允许NULL	类型	描述
AppId	是	是	String	AppId 示例值：
ClusterId	是	是	String	TKE集群ID 示例值：
Namespace	是	是	String	TKE集群命名空间 示例值：
ServiceName	是	是	String	TKE集群工作负载名称 示例值：
ContainerName	是	是	String	TKE集群工作负载中容器名称 示例值：
ClusterRegion	是	是	Int64	TKE集群地域数字ID 示例值：

DescribeApplicationTriggerPersonalResp

拉取触发器列表返回值

被如下接口引用：DescribeApplicationTriggerPersonal

名称	必选	允许NULL	类型	描述
TotalCount	是	否	Int64	返回条目总数 示例值：

名称	必选	允许NULL	类型	描述
TriggerInfo	是	否	Array of TriggerResp	触发器列表 示例值：查看

NamespaceIsExistsResp

NamespaceIsExists返回类型

被如下接口引用：ValidateNamespaceExistPersonal

名称	必选	允许NULL	类型	描述
IsExist	是	否	Bool	命名空间是否存在 示例值：
IsPreserved	是	否	Bool	是否为保留命名空间 示例值：

BuildReposList

构建仓库列表信息

被如下接口引用：DescribeSourceCodeRepositoryPersonal

名称	必选	允许NULL	类型	描述
Repos	是	否	Array of BuildRepo	仓库信息列表 示例值：查看

RepositoryInfoResp

查询共享版仓库信息返回

被如下接口引用：DescribeRepositoryPersonal

名称	必选	允许NULL	类型	描述
RepoName	是	否	String	镜像仓库名字 示例值：

名称	必选	允许NULL	类型	描述
RepoType	是	否	String	镜像仓库类型 示例值：
Server	是	否	String	镜像仓库服务地址 示例值：
CreationTime	是	否	String	创建时间 示例值：
Description	是	是	String	镜像仓库描述 示例值：
Public	是	否	Int64	是否为公有镜像 示例值：
PullCount	是	否	Int64	下载次数 示例值：
FavorCount	是	否	Int64	收藏次数 示例值：
IsUserFavor	是	否	Bool	是否为用户收藏 示例值：
IsQcloudOfficial	是	否	Bool	是否为云平台官方镜像 示例值：

错误码

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 cor
rect."
    },
    "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
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接口鉴权文档检查签名计算过程。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未 CAM 授权。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。

错误码	说明
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
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.ErrRegistry	内部服务错误(Registry)
LimitExceeded.ErrNamespaceMaxLimit	用户命名空间达到配额
InternalError.ErrGitApi	内部服务错误(GitApi)
UnsupportedOperation	操作不支持。

错误码	说明
InvalidParameter.ErrUserExist	用户已经存在
ResourceNotFound.ErrNoCommit	资源未找到(Commit)
InvalidParameter.ErrRepoExist	无效的参数，仓库已存在。
ResourceNotFound.ErrNoRepo	仓库不存在
InvalidParameter.ErrTriggerExist	触发器名称已存在
UnknownParameter	未知参数错误。
InternalError	内部错误。
LimitExceeded.ErrTriggerMaxLimit	触发器达到配额
InternalError.ErrUnauthorized	鉴权失败。
MissingParameter.MissingParameter	缺少参数
InternalError.DbError	数据库错误
InvalidParameter.ErrNSMisMatch	用户请求中的信息与其namespace不匹配
InvalidParameter.ErrNamespaceReserved	命名空间已被占用
InternalError.ErrMustAdmin	内部服务错误，必须为Admin
ResourceNotFound.ErrNoNamespace	用户没有创建命名空间
ResourceNotFound.ErrNoTag	tag不存在
InvalidParameter	参数错误。
MissingParameter	缺少参数错误。
ResourceNotFound.ErrNoTrigger	触发器不存在
InternalError.ErrAuthAssigned	已授权
ResourceNotFound.ErrNoAuth	未找到授权信息
ResourceNotFound.ErrNoUser	用户不存在（未注册）
InvalidParameter.ErrNamespaceExist	命名空间名称已经存在

容器服务（tke）
版本（2018-05-25）
API 概览

API版本

V3

伸缩组相关接口

接口名称	接口功能
CreateClusterAsGroup	创建集群的伸缩组
DeleteClusterAsGroups	删除集群伸缩组
DescribeClusterAsGroupOption	查询集群层面的弹性伸缩配置
DescribeClusterAsGroups	查询集群关联的伸缩组列表
ModifyClusterAsGroupAttribute	修改集群伸缩组属性
ModifyClusterAsGroupOptionAttribute	修改集群弹性伸缩组的属性

其他接口

接口名称	接口功能
DescribeImages	获取镜像信息
DescribeQuota	获取集群配额
DescribeRegions	查询地域列表
DescribeVersions	获取支持的集群版本信息
ForwardRequest	TKE APIServer代理转发接口

监控相关接口

接口名称	接口功能
DisableClusterAudit	关闭集群审计
DisableEventPersistence	关闭集群事件存储
EnableClusterAudit	启动集群审计
EnableEventPersistence	启动集群审计
GetDashboardID	获取集群的仪表盘ID

网络相关接口

接口名称	接口功能
AddVpcCniSubnets	添加容器子网（VPC-CNI网络类型的集群）

节点相关接口

接口名称	接口功能
AddExistedInstances	添加已经存在的CVM实例到集群
CreateClusterInstances	扩展集群节点
DeleteClusterInstances	删除集群中的节点
DescribeClusterInstanceIds	获取集群节点列表
DescribeClusterInstances	查询集群节点信息
DescribeExistedInstances	查询已经存在的节点
DrainClusterNode	驱逐集群中的指定节点

集群相关接口

接口名称	接口功能

接口名称	接口功能
CheckClusterCIDR	检查集群的CIDR是否冲突
CheckIfLogCollectorExists	检查集群指定日志采集规则（旧版）是否已经存在
CollectAllCore	采集当前TKE管控集群所使用的cpu总核数
CreateCluster	创建集群
CreateClusterEndpoint	开启集群内网/外网访问
CreateClusterVirtualNode	添加虚拟节点至指定集群的指定虚拟节点池
CreateClusterVirtualNodePool	在指定集群内创建虚拟节点池
CreateLogCollector	创建集群日志收集规则(旧版)
DeleteCluster	删除集群
DeleteClusterVirtualNode	删除集群指定虚拟节点
DeleteClusterVirtualNodePool	删除集群指定虚拟节点池
DeleteHelmChart	删除Chart
DeleteHelmChartVersion	删除Chart指定版本
DeleteLogCollector	删除日志收集规则(旧版)
DescribeChartDownloadInfo	获取Chart的下载信息
DescribeClusterEndpoint	查询集群endpoints的信息
DescribeClusterEndpointStatus	查询集群内网/外网访问端口的状态
DescribeClusterKubeconfig	获取集群的kubeconfig
DescribeClusterLevelAttribute	查询支持的集群等级
DescribeClusterSecurity	查询集群的密钥信息
DescribeClusterServices	获取集群Service列表
DescribeClusterVirtualNode	获取集群指定虚拟节点池的节点列表
DescribeClusterVirtualNodePools	获取集群的虚拟节点池列表
DescribeClusters	查询集群列表
DescribeCosInfo	获取chart上传cos信息

接口名称	接口功能
DescribeHelmChart	获取chart列表
DescribeHelmChartDetail	获取chart详情
DescribeHelmChartVersion	获取chart版本列表
DrainClusterVirtualNode	驱逐集群指定虚拟节点
EnableLogCollector	启用指定集群的日志收集服务（旧版）
GetLogCollector	获取集群指定日志收集规则的信息(旧版)
GetLogCollectorStatus	获取集群日志采集（旧版）启用状态
GetPrices	查询价格
ListLogCollector	查询集群日志收集列表(旧)
ModifyClusterAttribute	修改集群属性
ModifyClusterVirtualNodePool	修改虚拟节点池
QueryOverage	查询当前TKE管控集群的cpu总核数是否超过限制
UpdateLogCollector	更新集群指定日志收集规则(旧版)
UploadHelmChart	上传Chart

调用方式
接口签名v1

TCloudFinanceZone 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TCloudFinanceZone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，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐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。以下是生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

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

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥Id	AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	shjr
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

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' : 'shjr',
    'SecretId' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE',
    '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=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&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.finance.cloud.tencent.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.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&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 = 'Gu5t9xGARNpq86cd98joQYCN3EXAMPLE';
$srcStr = 'GETcvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&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:

```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

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 EliP9YW3pW28FpsEdkXt/+WcGeI= ，最终得到的签名串请求参数（Signature）为：EliP9YW3pW28FpsEdkXt%2f%2bWcGeI%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 http 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TCloudFinanceZone 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
- Node

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.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Signature=EliP9YW3pW28FpsEdkXt%2F%2BWcGeI%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，即使是旧版的 API ，由于存在细节差异也会导致签名计算错误，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 CloudAPIDemo {
    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.finance.cloud.tencent.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.finance.cloud.tencent.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)).app
end("&");
        }
        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.Intege
r.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
        params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"); // 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", "shjr"); // 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
        params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE
", "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 time

import requests

secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

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.finance.cloud.tencent.com"
    data = {
        'Action' : 'DescribeInstances',
        'InstanceIds.0' : 'ins-09dx96dg',
        'Limit' : 20,
        'Nonce' : 11886,
        'Offset' : 0,
        'Region' : 'shjr',
        '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)
```

接口签名v3

TCloudFinanceZone 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TCloudFinanceZone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，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

2. TC3-HMAC-SHA256 签名方法

注意：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 格式默认所有业务接口均支持，multipart 格式只有特定业务接口支持，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。

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。我们仅用到了查询实例列表的两个参数：Limit 和 Offset，使用 GET 方法调用。

假设用户的 SecretId 和 SecretKey 分别是：AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE 和 Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

2.1. 拼接规范请求串

按如下格式拼接规范请求串（CanonicalRequest）：

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

- HTTPRequestMethod：HTTP 请求方法（GET、POST ），本示例中为 GET；
- 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，本示例取值为：Limit=10&Offset=0。注意：CanonicalQueryString 需要经过 URL 编码。
- CanonicalHeaders：参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入自定义的头部参与签名以提高自身请求的唯一性和安全性。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）的字典排序进行拼接。此例中为： content-type:application/x-www-form-urlencoded\nhost:cvm.finance.cloud.tencent.com\n
- 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）按照字典排序进行拼接，并且以分号（;）分隔。此例中为： content-type;host
- HashedRequestPayload：请求正文的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，对 HTTP 请求整个正文 payload 做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。注意：对于 GET 请求，RequestPayload 固定为空字符串，对于 POST 请求，RequestPayload 即为 HTTP 请求正文 payload。

根据以上规则，示例中得到的规范请求串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
GET
/
Limit=10&Offset=0
content-type:application/x-www-form-urlencoded
host:cvm.finance.cloud.tencent.com

content-type;host
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

2.2. 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
```

- 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 X-TC-Timestamp 取值，如上示例请求为 1539084154；
- 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，例如 cvm。如上示例请求，取值为 2018-10-09/cvm/tc3_request；
- HashedCanonicalRequest：前述步骤拼接所得规范请求串的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。

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，可能导致运行一段时间后，请求必定失败（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（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
TC3-HMAC-SHA256
1539084154
2018-10-09/cvm/tc3_request
91c9c192c14460df6c1ffc69e34e6c5e90708de2a6d282cccf957dbf1aa7f3a7
```

2.3. 计算签名

1）计算派生签名密钥，伪代码如下

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

- SecretKey：原始的 SecretKey；
- Date：即 Credential 中的 Date 字段信息，如上示例，为2018-10-09；
- Service：即 Credential 中的 Service 字段信息，如上示例，为 cvm；

2）计算签名，伪代码如下

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

- SecretSigning：即以上计算得到的派生签名密钥；
- StringToSign：即步骤2计算得到的待签名字符串；

2.4. 拼接 Authorization

按如下格式拼接 Authorization：

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

- Algorithm：签名方法，固定为 TC3-HMAC-SHA256；
- SecretId：密钥对中的 SecretId；
- CredentialScope：见上文，凭证范围；
- SignedHeaders：见上文，参与签名的头部信息；
- Signature：签名值

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
```

最终完整的调用信息如下：

```
https://cvm.finance.cloud.tencent.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.finance.cloud.tencent.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: shjr
```

3. 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）

4. 签名演示

Java

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.bind.DatatypeConverter;

import org.apache.commons.codec.digest.DigestUtils;

public class CloudAPITC3Demo {
    private final static String CHARSET = "UTF-8";
    private final static String ENDPOINT = "cvm.finance.cloud.tencent.com";
    private final static String PATH = "/";
    private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE";
    private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE";
    private final static String CT_X_WWW_FORM_URLENCODED = "application/x-www-form-urlencode
d";
    private final static String CT_JSON = "application/json";
```

```
    private final static String CT_FORM_DATA = "multipart/form-data";

    public static byte[] sign256(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(CHARSET));
    }

    public static void main(String[] args) throws Exception {
        String service = "cvm";
        String host = "cvm.finance.cloud.tencent.com";
        String region = "shjr";
        String action = "DescribeInstances";
        String version = "2017-03-12";
        String algorithm = "TC3-HMAC-SHA256";
        String timestamp = "1539084154";
        //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 = "GET";
        String canonicalUri = "/";
        String canonicalQueryString = "Limit=10&Offset=0";
        String canonicalHeaders = "content-type:application/x-www-form-urlencoded\n" + "host:" + host
+ "\n";
        String signedHeaders = "content-type;host";
        String hashedRequestPayload = DigestUtils.sha256Hex("");
        String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryStri
ng + "\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 = DigestUtils.sha256Hex(canonicalRequest.getBytes(CHARSET));
        String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCan
onicalRequest;
        System.out.println(stringToSign);

        // ************ 步骤 3：计算签名 ************
        byte[] secretDate = sign256(("TC3" + SECRET_KEY).getBytes(CHARSET), date);
        byte[] secretService = sign256(secretDate, service);
        byte[] secretSigning = sign256(secretService, "tc3_request");
```

```
        String signature = DatatypeConverter.printHexBinary(sign256(secretSigning, stringToSign)).toLow
erCase();
        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("Host", host);
        headers.put("Content-Type", CT_X_WWW_FORM_URLENCODED);
        headers.put("X-TC-Action", action);
        headers.put("X-TC-Timestamp", timestamp);
        headers.put("X-TC-Version", version);
        headers.put("X-TC-Region", region);
    }
}
```

Python

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

# 密钥参数
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

service = "cvm"
host = "cvm.finance.cloud.tencent.com"
endpoint = "https://" + host
region = "shjr"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
timestamp = 1539084154
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 10, "Offset": 0}

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = "GET"
canonical_uri = "/"
canonical_querystring = "Limit=10&Offset=0"
ct = "x-www-form-urlencoded"
```

```
payload = ""
if http_request_method == "POST":
    canonical_querystring = ""
    ct = "json"
    payload = json.dumps(params)
canonical_headers = "content-type:application/%s\nhost:%s\n" % (ct, host)
signed_headers = "content-type;host"
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)

# 公共参数添加到请求头部
headers = {
    "Authorization": authorization,
    "Host": host,
    "Content-Type": "application/%s" % ct,
```

```
    "X-TC-Action": action,
    "X-TC-Timestamp": str(timestamp),
    "X-TC-Version": version,
    "X-TC-Region": region,
}
```

请求结构

1. 服务地址

地域（Region）是指物理的数据中心的地理区域。TCloudFinanceZone交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 查询地域列表 查看完成的地域列表。

2. 通信协议

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

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

4. 字符编码

均使用UTF-8编码。

返回结果

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，可以联系我们，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 cor
rect."
        },
        "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，可以联系我们，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，下表列出了公共错误码。

错误码	错误描述
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。
AuthFailure.SignatureExpire	签名过期。
AuthFailure.SignatureFailure	签名错误。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未 CAM 授权。
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。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。

错误码	错误描述
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
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。

公共参数

公共参数是用于标识用户和接口鉴权目的的参数，如非必要，在每个接口单独的接口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。

签名方法 v3

使用 TC3-HMAC-SHA256 签名方法时，公共参数需要统一放到 HTTP Header 请求头部中，如下：

参数名称	类型	必选	描述
X-TC-Action	String	是	操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
X-TC-Region	String	是	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。
X-TC-Timestamp	Integer	是	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。
X-TC-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。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/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，AKIDEXAMPLE 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为产品名，必须与调用的产品域名一致，例如cvm； - 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。
X-TC-Token	String	否	临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token。

签名方法 v1

使用 HmacSHA1 和 HmacSHA256 签名方法时，公共参数需要统一放到请求串中，如下

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	是	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。

参数名称	类型	必选	描述
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。具体计算方法参见接口鉴权文档。
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	临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token 。

地域列表

地域（Region）是指物理的数据中心的地理区域。TCloudFinanceZone交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 查询地域列表 查看完成的地域列表。

伸缩组相关接口
创建集群的伸缩组

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

创建伸缩组

默认接口请求频率限制：20次/秒。

接口更新时间：2020-09-03 19:31:58。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateClusterAsGroup
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
AutoScalingGroupPara	是	否	String	伸缩组创建透传参数，json化字符串格式，详见接口CreateAutoScalingGroup。LaunchConfigurationId由LaunchConfigurePara参数创建，不支持填写 示例值：
LaunchConfigurePara	是	否	String	启动配置创建透传参数，json化字符串格式，详见CreateLaunchConfiguration接口。另外ImageId参数由于集群维度已经有的ImageId信

参数名称	必选	允许NULL	类型	描述
				息，这个字段不需要填写。UserData字段设置通过UserScript设置，这个字段不需要填写。 示例值：
InstanceAdvancedSettings	否	否	InstanceAdvancedSettings	节点高级配置信息 示例值：查看
Labels	否	否	Array of Label	节点Label数组 示例值：查看

3. 输出参数

参数名称	类型	描述
LaunchConfigurationId	String	启动配置ID 示例值：
AutoScalingGroupId	String	伸缩组ID 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的RequestId。

4. 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.OsNotSupport	镜像OS不支持。
MissingParameter	缺少参数错误
InternalError.AsCommon	伸缩组资源创建报错。
InternalError.CvmNotFound	cvm不存在。
InternalError.Db	db错误。
InvalidParameter	参数错误
InternalError.AccountUserNotAuthenticated	账户未通过认证。
InternalError.CvmCommon	cvm创建节点报错。

错误码	描述
LimitExceeded	超过配额限制
InternalError.DbAffectivedRows	DB错误
InvalidParameter.AsCommonError	弹性伸缩组创建参数错误。
UnsupportedOperation	操作不支持
InternalError.Param	Param。
InternalError.ImageIdNotFound	镜像未找到。
FailedOperation	操作失败
InternalError	内部错误

删除集群伸缩组

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

删除集群伸缩组

默认接口请求频率限制：20次/秒。

接口更新时间：2020-06-18 19:41:47。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteClusterAsGroups
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID，通过DescribeClusters接口获取。 示例值：
AutoScalingGroupIds	是	否	Array of String	集群伸缩组ID的列表 示例值：
KeepInstance	否	否	Bool	是否保留伸缩组中的节点(默认值： false(不保留)) 示例值： false

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.QuotaMaxClsLimit	超过配额限制。
LimitExceeded	超过配额限制
InternalError.QuotaMaxNodLimit	超过配额限制。
UnknownParameter	未知参数错误
InvalidParameter	参数错误
InvalidParameter.Param	参数错误。
ResourceNotFound	资源不存在
InternalError.Param	Param。
InvalidParameter.AsCommonError	弹性伸缩组创建参数错误。
InternalError.QuotaMaxRtLimit	超过配额限制。
InternalError.AccountUserNotAuthenticated	账户未通过认证。
ResourceInUse	资源被占用
InternalError	内部错误
FailedOperation	操作失败
InternalError.AsCommon	伸缩组资源创建报错。
InternalError.PublicClusterOpNotSupport	集群不支持当前操作。

查询集群层面的弹性伸缩配置

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询集群层面的弹性伸缩配置

默认接口请求频率限制：20次/秒。

接口更新时间：2020-06-18 19:39:31。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterAsGroupOption
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：

3. 输出参数

参数名称	类型	描述
ClusterAsGroupOption	ClusterAsGroupOption	集群弹性伸缩属性 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
UnsupportedOperation	操作不支持
InternalError.Param	Param。
InternalError.CamNoAuth	没有权限。
InternalError.AccountUserNotAuthenticated	账户未通过认证。
InternalError	内部错误
FailedOperation	操作失败
InternalError.AsCommon	伸缩组资源创建报错。

查询集群关联的伸缩组列表

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询集群关联的伸缩组列表

默认接口请求频率限制：20次/秒。

接口更新时间：2020-06-18 19:42:36。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterAsGroups
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
AutoScalingGroupIds	否	否	Array of String	伸缩组ID列表，如果为空，表示拉取集群关联的所有伸缩组。 示例值：
Offset	否	否	Int64	偏移量，默认为0。用来控制分页的参数；Limit 为单次返回的最多条目数量，Offset 为偏移量。当相应结果是列表形式时，如果数量超过了 Limit 所限定的值，那么只返回 Limit 个值。 示例值：
Limit	否	否	Int64	返回数量，默认为20，最大值为100。用来控制分页的参数；Limit 为单次返回的最多条目数量，Offset 为偏移量。当相应结果是列表形式时，如果数量超过了 Limit 所限定的值，那么只返

参数名称	必选	允许NULL	类型	描述
				回 Limit 个值。 示例值：

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	集群关联的伸缩组总数 示例值：
ClusterAsGroupSet	Array of ClusterAsGroup	集群关联的伸缩组列表 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.VpcCommon	VPC报错。
InternalError.VpcPeerNotFound	对等连接不存在。
InternalError.VpcRecodrNotFound	未发现vpc记录。
InternalError.Db	db错误。
InternalError.Param	Param。
InternalError.AccountUserNotAuthenticated	账户未通过认证。
InternalError	内部错误
FailedOperation	操作失败
InternalError.AsCommon	伸缩组资源创建报错。
InternalError.PodNotFound	Pod未找到。

修改集群伸缩组属性

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

修改集群伸缩组属性

默认接口请求频率限制：20次/秒。

接口更新时间：2020-06-18 19:43:07。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyClusterAsGroupAttribute
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
ClusterAsGroupAttribute	否	否	ClusterAsGroupAttribute	集群关联的伸缩组属性 示例值：查看

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.RouteTableNotEmpty	路由表非空。
LimitExceeded	超过配额限制
UnknownParameter	未知参数错误
InvalidParameter.GatewayAlreadyAssociatedCidr	下一跳地址已关联CIDR。
InvalidParameter	参数错误
InvalidParameter.Param	参数错误。
ResourceNotFound	资源不存在
UnsupportedOperation	操作不支持
InvalidParameter.AsCommonError	弹性伸缩组创建参数错误。
MissingParameter	缺少参数错误
InternalError.CamNoAuth	没有权限。
InternalError.AccountUserNotAuthenticated	账户未通过认证。
InvalidParameter.CidrOutOfRouteTable	CIDR不在路由表之内。
ResourceInUse	资源被占用
InternalError	内部错误
ResourceUnavailable	资源不可用
UnauthorizedOperation	未授权操作
FailedOperation	操作失败
InternalError.AsCommon	伸缩组资源创建报错。

修改集群弹性伸缩组的属性

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

修改集群弹性伸缩组的属性

默认接口请求频率限制：20次/秒。

接口更新时间：2020-08-22 22:08:42。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyClusterAsGroupOptionAttribute
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
ClusterAsGroupOption	是	否	ClusterAsGroupOption	集群弹性伸缩属性 示例值：查看

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
ResourceUnavailable	资源不可用
InternalError	内部错误
ResourceNotFound	资源不存在
MissingParameter	缺少参数错误
InternalError.CvmCommon	cvm创建节点报错。
InternalError.CvmNotFound	cvm不存在。
InternalError.Db	db错误。
ResourceInUse	资源被占用
InternalError.ClusterNotFound	集群未找到。
UnknownParameter	未知参数错误
UnauthorizedOperation	未授权操作
InternalError.DbAffectivedRows	DB错误
InternalError.AsCommon	伸缩组资源创建报错。
UnsupportedOperation	操作不支持

其他接口
获取镜像信息

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取镜像信息

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-09 10:35:32。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeImages
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	镜像数量 示例值：
ImageInstanceSet	Array of ImageInstance	镜像信息列表 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.RouteTableNotEmpty	路由表非空。
InternalError	内部错误
InternalError.DbAffectivedRows	DB错误
InternalError.DbRecordNotFound	记录未找到。
InternalError.CamNoAuth	没有权限。
LimitExceeded	超过配额限制
InternalError.ImageIdNotFound	镜像未找到。
InternalError.Db	db错误。
UnauthorizedOperation	未授权操作
FailedOperation	操作失败
InvalidParameter.Param	参数错误。
MissingParameter	缺少参数错误
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

获取集群配额

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取集群配额

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-09 10:36:43。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeQuota
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表

3. 输出参数

参数名称	类型	描述
MaxClustersNum	Uint64	该账号当前地域支持的最大集群数量 示例值：
MaxNodesNum	Uint64	该账号当前地域单集群支持的最大节点数量 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.RouteTableNotEmpty	路由表非空。
InternalError.DbAffectivedRows	DB错误
InternalError.DbRecordNotFound	记录未找到。
LimitExceeded	超过配额限制
InternalError.Db	db错误。
UnauthorizedOperation	未授权操作
InvalidParameter.Param	参数错误。
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

查询地域列表

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取容器服务支持的所有地域

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-09 10:39:30。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeRegions
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	地域的数量 示例值：
RegionInstanceSet	Array of RegionInstance	地域列表 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.DbAffectivedRows	DB错误
InternalError.DbRecordNotFound	记录未找到。
InternalError.CamNoAuth	没有权限。
LimitExceeded	超过配额限制
InternalError.Db	db错误。
UnauthorizedOperation	未授权操作
FailedOperation	操作失败
InternalError.AccountUserNotAuthenticated	账户未通过认证。
MissingParameter	缺少参数错误
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

获取支持的集群版本信息

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取支持的集群版本

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-09 10:30:27。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeVersions
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	版本数量 示例值：
VersionInstanceSet	Array of VersionInstance	版本列表 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.RouteTableNotEmpty	路由表非空。
InternalError	内部错误
InternalError.DbAffectivedRows	DB错误
InternalError.DbRecordNotFound	记录未找到。
InternalError.CamNoAuth	没有权限。
LimitExceeded	超过配额限制
InternalError.Db	db错误。
UnauthorizedOperation	未授权操作
InvalidParameter.Param	参数错误。
MissingParameter	缺少参数错误
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
InternalError.Param	Param。
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

TKE APIServer代理转发接口

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

YUNAPI 转发请求给TKE APIServer接口

默认接口请求频率限制：200次/秒。

接口更新时间：2021-04-09 10:36:02。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ForwardRequest
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
Method	是	否	String	请求tke-apiserver http请求对应的方法 示例值：
Path	是	否	String	请求tke-apiserver http请求访问路径 示例值：
Accept	否	否	String	请求tke-apiserver http头中Accept参数 示例值：
ContentType	否	否	String	请求tke-apiserver http头中ContentType参数 示例值：
RequestBody	否	否	String	请求tke-apiserver http请求body信息 示例值：
ClusterName	否	否	String	请求tke-apiserver http头中X-TKE-ClusterName参数 示例值：

参数名称	必选	允许NULL	类型	描述
EncodedBody	否	否	Bool	是否编码请求body信息 示例值：
ClusterId	否	否	String	鉴权使用 示例值：

3. 输出参数

参数名称	类型	描述
ResponseBody	String	请求tke-apiserver http请求返回的body信息 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误
LimitExceeded	超过配额限制
UnauthorizedOperation	未授权操作
FailedOperation	操作失败
MissingParameter	缺少参数错误
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

监控相关接口
关闭集群审计

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

关闭集群审计

默认接口请求频率限制：20次/秒。

接口更新时间：2022-01-21 11:24:59。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DisableClusterAudit
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InvalidParameter	参数错误
InternalError	内部错误

关闭集群事件存储

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

关闭集群事件存储

默认接口请求频率限制：20次/秒。

接口更新时间：2022-01-21 11:24:59。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DisableEventPersistence
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InvalidParameter	参数错误
InternalError	内部错误

启动集群审计

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

启动集群审计
1）使用用户指定的日志集和日志主题。
2）使用默认的日志集和日志主题。

默认接口请求频率限制：20次/秒。

接口更新时间：2022-01-21 11:29:23。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：EnableClusterAudit
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
LogsetId	否	否	String	日志集ID 示例值：
TopicId	否	否	String	日志主题ID 示例值：
RegionId	是	否	String	区域ID 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误
UnknownParameter	未知参数错误

启动集群审计

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

启动集群事件存储
1）使用用户指定的日志集和日志主题。
2）使用默认的日志集和日志主题。

默认接口请求频率限制：20次/秒。

接口更新时间：2022-01-21 11:29:23。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：EnableEventPersistence
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
LogsetId	否	否	String	日志集ID 示例值：
TopicId	否	否	String	日志主题ID 示例值：
RegionId	是	否	String	区域ID 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误
UnknownParameter	未知参数错误

获取集群的仪表盘ID

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取集群的仪表盘ID
1.DashboardType可选值:
针对审计：
audit-overview,audit-search,audit-node-overview,audit-k8s-resource-overview

默认接口请求频率限制：20次/秒。

接口更新时间：2022-01-21 11:26:42。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：GetDashboardID
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
DashboardType	是	否	String	仪表盘的类型。传audit-overview表示审计总览，传audit-search表示聚合检索，传audit-node-overview表示节点操作概览，传audit-k8s-resource-overview表示K8S对象操作概览，传event-overview表示事件总览，传event-search表示异常事件聚合检索。 示例值： audit-overview

3. 输出参数

参数名称	类型	描述
DashboardID	String	仪表盘ID 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InvalidParameter	参数错误
InternalError	内部错误

网络相关接口
添加容器子网（VPC-CNI网络类型的集群）

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

为VPC-CNI网络类型的集群添加容器子网

默认接口请求频率限制：20次/秒。

接口更新时间：2021-11-15 20:35:21。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：AddVpcCniSubnets
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
VpcId	是	否	String	VPC ID 示例值：
SubnetIds	是	否	Array of String	要添加的子网ID 示例值：

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	参数错误
FailedOperation	操作失败
UnknownParameter	未知参数错误

节点相关接口
添加已经存在的CVM实例到集群

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

添加已经存在的cvm实例到集群

默认接口请求频率限制：20次/秒。

接口更新时间：2021-05-10 11:20:07。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：AddExistedInstances
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
InstanceIds	是	否	Array of String	实例列表 示例值：
InstanceAdvancedSettings	否	否	InstanceAdvancedSettings	实例额外需要设置参数信息 示例值：查看
EnhancedService	否	否	EnhancedService	增强服务。通过该参数可以指定是否开启云安全、云监控等服务。若不指定该参数，则默认开启云监控、云安全服务。 示例值：查看
LoginSettings	否	否	LoginSettings	节点登录信息（目前仅支持使用Password或者单个KeyIds）

参数名称	必选	允许NULL	类型	描述
				示例值：查看
SecurityGroupIds	否	否	Array of String	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。（目前仅支持设置单个sgId） 示例值：
HostName	否	否	String	重装系统时，可以指定修改实例的HostName(集群为HostName模式时，此参数必传，规则名称除不支持大写字符外与RunInstances接口HostName一致) 示例值：
ResetCvmFlag	否	否	String	节点加入集群时，是否重装系统。可取值 true/false, 默认为 true, 表示需要重装系统 示例值：
NodePool	否	否	NodePoolOption	节点池选项 示例值：查看

3. 输出参数

参数名称	类型	描述
FailedInstanceIds	Array of String	失败的节点ID 示例值：
SuccInstanceIds	Array of String	成功的节点ID 示例值：
TimeoutInstanceIds	Array of String	超时未返回出来节点的ID(可能失败，也可能成功) 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.DbAffectivedRows	DB错误
InvalidParameter	参数错误
LimitExceeded	超过配额限制
InternalError.Db	db错误。
InternalError.Param	Param。

扩展集群节点

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

新建集群节点

默认接口请求频率限制：20次/秒。

接口更新时间：2020-06-16 15:57:00。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateClusterInstances
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群 ID，请填写 查询集群列表 接口中返回的 clusterId 字段 示例值：
InstanceAdvancedSettings	否	否	InstanceAdvancedSettings	实例额外需要设置参数信息 示例值：查看
RunInstancePara	是	否	String	CVM创建透传参数，json化字符串格式，详见RunInstances接口。 示例值：

3. 输出参数

参数名称	类型	描述
InstanceIdSet	Array of String	节点实例ID 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.QuotaMaxClsLimit	超过配额限制。
InternalError.ImageIdNotFound	镜像未找到。
InternalError.VpcCommon	VPC报错。
InternalError.VpcPeerNotFound	对等连接不存在。
InternalError.CvmCommon	cvm创建节点报错。
InternalError.QuotaMaxNodLimit	超过配额限制。
InternalError.VpcRecodrNotFound	未发现vpc记录。
UnknownParameter	未知参数错误
InternalError.Db	db错误。
InternalError.OsNotSupport	镜像OS不支持。
InternalError.CvmNotFound	cvm不存在。
InvalidParameter	参数错误
ResourceNotFound	资源不存在
UnsupportedOperation	操作不支持
InternalError.Param	Param。
MissingParameter	缺少参数错误

错误码	描述
InternalError.DbAffectivedRows	DB错误
InternalError.QuotaMaxRtLimit	超过配额限制。
InternalError.DbRecordNotFound	记录未找到。
ResourceInUse	资源被占用
InternalError	内部错误
InternalError.UnexceptedInternal	内部错误
ResourceUnavailable	资源不可用
UnauthorizedOperation	未授权操作
FailedOperation	操作失败

删除集群中的节点

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

删除集群中的节点

默认接口请求频率限制：20次/秒。

接口更新时间：2021-05-10 11:17:55。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteClusterInstances
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
InstanceIds	否	否	Array of String	主机InstanceId列表 示例值：
MachineNames	否	否	Array of String	主机MachineName列表 示例值：
InstanceDeleteMode	否	否	String	集群实例删除时的策略。terminate表示直接销毁实例[仅支持按量计费云主机实例],retain表示仅移除实例，不进行销毁。 示例值： terminate
ForceDelete	否	否	Bool	是否强制删除(当节点在初始化时，可以指定参数为true) 示例值： true

3. 输出参数

参数名称	类型	描述
SuccInstanceIds	Array of String	成功删除的cvm实例id列表 示例值：
FailedInstanceIds	Array of String	未成功删除的cvm实例id列表 示例值：
NotFoundInstanceIds	Array of String	未找到的cvm实例id列表 示例值：
SuccMachineNames	Array of String	成功删除的MachineName列表 示例值：
FailedMachineNames	Array of String	未成功删除的MachineName列表 示例值：
NotFoundMachineNames	Array of String	未找到的MachineName列表 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.DbAffectivedRows	DB错误
InvalidParameter	参数错误
LimitExceeded	超过配额限制
InternalError.Db	db错误。
InternalError.Param	Param。
InternalError.PublicClusterOpNotSupport	集群不支持当前操作。

获取集群节点列表

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取集群节点列表

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-09 10:27:23。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterInstanceIds
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
InstanceRole	否	否	String	节点角色。MASTER_ETCD表示master节点,WORKER表示工作节点,ALL表示所有节点 示例值： MASTER_ETCD
Offset	否	否	Int64	偏移量，默认为0。用来控制分页的参数；Limit 为单次返回的最多条目数量，Offset 为偏移量。当相应结果是列表形式时，如果数量超过了 Limit 所限定的值，那么只返回 Limit 个值。 示例值：
Limit	否	否	Int64	返回数量，默认为20，最大值为100。用来控制分页的参数；Limit 为单次返回的最多条目数量，Offset 为偏移量。当相应结果是列表形式时，如果数量超过了 Limit 所限定的值，那么只返

参数名称	必选	允许NULL	类型	描述
				回 Limit 个值。 示例值：
VagueIpAddress	否	否	String	根据节点的IP地址进行搜索，同时搜索内网IP和外网IP 示例值：
VagueInstanceName	否	否	String	根据节点的名称进行模糊搜索 示例值：
InstanceStates	否	否	Array of String	根据节点的状态进行筛选 示例值：
Filters	否	否	Array of Filter	过滤条件,当前支持多个过滤条件来搜索 示例值：查看
Labels	否	否	String	根据节点的标签进行搜索 示例值：

3. 输出参数

参数名称	类型	描述
InstanceIdSet	Array of String	节点ID列表 示例值：
TotalCount	Uint64	数量 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误

错误码	描述
InternalError.VpcPeerNotFound	对等连接不存在。
InternalError.VpcRecodrNotFound	未发现vpc记录。
InternalError.AccountUserNotAuthenticated	账户未通过认证。
InternalError.AsCommon	伸缩组资源创建报错。
InternalError.CvmCommon	cvm创建节点报错。
InternalError.Param	Param。
InternalError.VpcCommon	VPC报错。

查询集群节点信息

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询集群的节点列表

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-29 11:45:57。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterInstances
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Offset	否	否	Int64	偏移量，默认为0。用来控制分页的参数；Limit 为单次返回的最多条目数量，Offset 为偏移量。当相应结果是列表形式时，如果数量超过了 Limit 所限定的值，那么只返回 Limit 个值。 示例值：
Limit	否	否	Int64	返回数量，默认为20，最大值为100。用来控制分页的参数；Limit 为单次返回的最多条目数量，Offset 为偏移量。当相应结果是列表形式时，如果数量超过了 Limit 所限定的值，那么只返回 Limit 个值。 示例值：
InstanceIds	否	否	Array of String	需要获取的节点实例Id列表。如果为空，表示拉取集群下所有节点实例。 示例值：

参数名称	必选	允许NULL	类型	描述
InstanceRole	否	否	String	节点角色。 MASTER_ETCD表示只返回master节点信息，WORKER表示只返回work节点信息，ALL表示返回所有节点信息,""空字符串表示返回所有节点信息 示例值： MASTER_ETCD
Filters	否	否	Array of Filter	集群节点列表过滤条件，支持多条件过滤 示例值：查看

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	集群中实例总数 示例值：
InstanceSet	Array of Instance	集群中实例列表 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.DbAffectivedRows	DB错误
InvalidParameter.ClusterNotFound	集群ID不存在。
InternalError.Db	db错误。
FailedOperation	操作失败
InternalError.InitMasterFailed	初始化master失败。
InternalError.Param	Param。

错误码	描述
InternalError.PublicClusterOpNotSupport	集群不支持当前操作。
ResourceNotFound	资源不存在

查询已经存在的节点

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询已经存在的节点，判断是否可以加入集群

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-09 10:37:29。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeExistedInstances
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	否	否	String	集群 ID，请填写查询集群列表 接口中返回的 ClusterId 字段（仅通过ClusterId获取需要过滤条件中的VPCID。节点状态比较时会使用该地域下所有集群中的节点进行比较。参数不支持同时指定InstanceIds和ClusterId。 示例值：
InstanceIds	否	否	Array of String	按照一个或者多个实例ID查询。实例ID形如：ins-xxxxxxxx。（此参数的具体格式可参考API简介的id.N一节）。每次请求的实例的上限为100。参数不支持同时指定InstanceIds和Filters。 示例值：
Filters	否	否	Array of Filter	过滤条件,字段和详见DescribeInstances如果设置了ClusterId，会附加集群的VPCID作为查询字段，在此情况下如果在Filter中指定了"vpc-id"作为过滤字段，指定的VPCID必须与集群的VPCID

参数名称	必选	允许NULL	类型	描述
				相同。 示例值：查看
VagueIpAddress	否	否	String	实例IP进行过滤(同时支持内网IP和外网IP) 示例值：
VagueInstanceName	否	否	String	实例名称进行过滤 示例值：
Offset	否	否	Uint64	偏移量，默认为0。用来控制分页的参数；Limit 为单次返回的最多条目数量，Offset 为偏移量。当相应结果是列表形式时，如果数量超过了 Limit 所限定的值，那么只返回 Limit 个值。 示例值：
Limit	否	否	Uint64	返回数量，默认为20，最大值为100。用来控制分页的参数；Limit 为单次返回的最多条目数量，Offset 为偏移量。当相应结果是列表形式时，如果数量超过了 Limit 所限定的值，那么只返回 Limit 个值。 示例值：

3. 输出参数

参数名称	类型	描述
ExistedInstanceSet	Array of ExistedInstance	已经存在的实例信息数组。 示例值：查看
TotalCount	Uint64	符合条件的实例数量。 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误

错误码	描述
InternalError.DbAffectivedRows	DB错误
InternalError.DbRecordNotFound	记录未找到。
InvalidParameter	参数错误
InternalError.CreateMasterFailed	创建集群失败。
InternalError.OsNotSupport	镜像OS不支持。
LimitExceeded	超过配额限制
InternalError.ImageIdNotFound	镜像未找到。
InternalError.Db	db错误。
InternalError.VpcRecodrNotFound	未发现vpc记录。
UnauthorizedOperation	未授权操作
FailedOperation	操作失败
InternalError.AccountUserNotAuthenticated	账户未通过认证。
InternalError.CvmCommon	cvm创建节点报错。
MissingParameter	缺少参数错误
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
InternalError.CvmNotFound	cvm不存在。
InternalError.InitMasterFailed	初始化master失败。
InternalError.InvalidPrivateNetworkCidr	无效CIDR。
InternalError.Param	Param。
InternalError.UnexceptedInternal	内部错误
InternalError.VpcCommon	VPC报错。
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

驱逐集群中的指定节点

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

驱逐集群中的指定节点

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-30 14:38:13。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DrainClusterNode
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
InstanceId	是	否	String	实例ID 示例值：
DryRun	否	否	Bool	是否驱逐节点前的预检。true表示不进行驱逐，只返回当前节点上的pod信息，false表示直接驱逐节点 示例值： true

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
Pods	Array of SimplePodInfo	节点上pod列表 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter.AsCommonError	弹性伸缩组创建参数错误。
InvalidParameter.CidrOutOfRouteTable	CIDR不在路由表之内。
InternalError.CamNoAuth	没有权限。
InvalidParameter	参数错误
LimitExceeded	超过配额限制
UnauthorizedOperation	未授权操作
FailedOperation	操作失败
InvalidParameter.Param	参数错误。
InternalError.AccountUserNotAuthenticated	账户未通过认证。
InternalError.AsCommon	伸缩组资源创建报错。
MissingParameter	缺少参数错误
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

集群相关接口
检查集群的CIDR是否冲突

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

检查集群的CIDR是否冲突

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-09 10:38:45。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CheckClusterCIDR
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
VpcId	是	否	String	集群的vpc-id 示例值：
ClusterCIDR	是	否	String	集群的CIDR 示例值：
NetworkType	是	否	String	集群的网络类型。GR表示global route集群，VPC-CNI表示vpc-cni集群 示例值： GR

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
IsConflict	Bool	是否存在CIDR冲突。true表示存在冲突,false表示不存在冲突 示例值： true
ConflictType	String	CIDR冲突的类型("CIDR_CONFLICT_WITH_OTHER_CLUSTER" 同VPC其他集群CIDR存在冲突 "CIDR_CONFLICT_WITH_VPC_CIDR" 与VPC的CIDR存在冲突 "CIDR_CONFLICT_WITH_VPC_GLOBAL_ROUTE" 与同VPC的全局路由存在冲突)。 示例值： CIDR_CONFLICT_WITH_OTHER_CLUSTER
ConflictMsg	String	CIDR冲突描述信息。 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.DbAffectivedRows	DB错误
InternalError.DbRecordNotFound	记录未找到。
InvalidParameter.CIDRInvalid	非法的CIDR。
InternalError.CidrConflictWithOtherCluster	CIDR和其他集群CIDR冲突。
InvalidParameter.CidrOutOfRouteTable	CIDR不在路由表之内。
InvalidParameter.GatewayAlreadyAssociatedCidr	下一跳地址已关联CIDR。
InternalError.CamNoAuth	没有权限。
InternalError.CidrMaskSizeOutOfRange	CIDR掩码无效。
InternalError.VstationError	VstationError。
InvalidParameter	参数错误
InvalidParameter.InvalidPrivateNetworkCIDR	无效的私有CIDR网段。
InternalError.CidrOutOfRouteTable	CIDR不在路由表之内。

错误码	描述
LimitExceeded	超过配额限制
InternalError.VpcPeerNotFound	对等连接不存在。
InternalError.Db	db错误。
InternalError.VpcRecodrNotFound	未发现vpc记录。
UnauthorizedOperation	未授权操作
InternalError.CidrConflictWithOtherRoute	CIDR和其他路由冲突。
InternalError.CidrInvali	CIDR无效。
InvalidParameter.Param	参数错误。
MissingParameter	缺少参数错误
InvalidParameter.CIDRMaskSizeOutOfRange	CIDR掩码超出范围(集群CIDR范围 最小值: 10 最大值: 24)。
InternalError.CidrConflictWithVpcCidr	CIDR和vpc冲突。
InternalError.CidrConflictWithVpcGlobalRoute	CIDR和全局路由冲突。
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
InternalError.UnexceptedInternal	内部错误
InternalError.VpcCommon	VPC报错。
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

检查集群指定日志采集规则（旧版）是否已经存在

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

检查集群指定日志采集规则（旧版）是否已经存在

默认接口请求频率限制：20次/秒。

接口更新时间：2022-04-12 20:35:16。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CheckIfLogCollectorExists
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Name	是	否	String	日志采集规则名字 示例值：

3. 输出参数

参数名称	类型	描述
CheckResult	Bool	日志采集规则是否已经存在。true表示日志采集规则存在，false表示日志采集规则不存在。 示例值： true

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InternalError	内部错误

采集当前TKE管控集群所使用的cpu总核数

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

license平台定时采集当前TKE管控集群所使用的cpu总核数

默认接口请求频率限制：20次/秒。

接口更新时间：2022-06-13 16:16:22。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CollectAllCore
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表

3. 输出参数

参数名称	类型	描述
ProductList	Array of Product	云产品License用量列表 示例值：查看
ProductCode	String	产品代码，可以用于产品过滤 示例值： "TKE"
ProductName	String	产品名称，固定TKE 示例值： "TKE"
Version	String	版本 示例值： "2018-05-28"

参数名称	类型	描述
Data	Array of MachineCore	资源使用量 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.QuotaMaxClsLimit	超过配额限制。
InternalError.TaskNotFound	任务未找到。
InternalError.Param	Param。
MissingParameter	缺少参数错误
InternalError	内部错误
InvalidParameter.Param	参数错误。

创建集群

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

创建集群

默认接口请求频率限制：20次/秒。

接口更新时间：2020-09-03 19:31:05。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateCluster
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
RunInstancesForNode	否	否	Array of RunInstancesForNode	CVM创建透传参数，json化字符串格式，详见RunInstances接口。总机型(包括地域)数量不超过10个，相同机型(地域)购买多台机器可以通过设置参数中RunInstances中InstanceCount来实现。 示例值：查看
ClusterCIDRSettings	是	否	ClusterCIDRSettings	集群容器网络配置信息 示例值：查看
ClusterBasicSettings	否	否	ClusterBasicSettings	集群的基本配置信息 示例值：查看
ClusterAdvancedSettings	否	否	ClusterAdvancedSettings	集群高级配置信息 示例值：查看
InstanceAdvancedSettings	否	否	InstanceAdvancedSettings	节点高级配置信息 示例值：查看

参数名称	必选	允许NULL	类型	描述
ClusterType	是	否	String	集群类型，托管集群：MANAGED_CLUSTER，独立集群：INDEPENDENT_CLUSTER。 示例值：INDEPENDENT_CLUSTER
ExistedInstancesForNode	否	否	Array of ExistedInstancesForNode	已存在实例的配置信息。所有实例必须在同一个VPC中，最大数量不超过100。 示例值：查看
InstanceDataDiskMountSettings	否	否	Array of InstanceDataDiskMountSetting	CVM类型和其对应的数据盘挂载配置信息 示例值：查看
ClusterArch	否	否	String	集群的cpu架构，取值： x86/arm 示例值： x86

3. 输出参数

参数名称	类型	描述
ClusterId	String	集群ID 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.DfwGetUSGCount	获得当前安全组总数失败。
InternalError.InitMasterFailed	初始化master失败。
InternalError.QuotaMaxClsLimit	超过配额限制。
InternalError.CidrInvali	CIDR无效。
InternalError.AsCommon	伸缩组资源创建报错。
InternalError.Db	db错误。
InternalError.UnexceptedInternal	内部错误
InvalidParameter	参数错误

错误码	描述
InternalError.CidrConflictWithVpcGlobalRoute	CIDR和全局路由冲突。
InternalError.InvalidPrivateNetworkCidr	无效CIDR。
InternalError.CidrConflictWithOtherRoute	CIDR和其他路由冲突。
InternalError.QuotaMaxNodLimit	超过配额限制。
InternalError.VpcRecodrNotFound	未发现vpc记录。
InternalError.AccountUserNotAuthenticated	账户未通过认证。
InternalError.CvmCommon	cvm创建节点报错。
InternalError.VpcCommon	VPC报错。
LimitExceeded	超过配额限制
InternalError.QuotaUSGLimit	安全组配额不足。
InternalError.DbAffectivedRows	DB错误
InternalError.CidrMaskSizeOutOfRange	CIDR掩码无效。
InternalError.CidrConflictWithOtherCluster	CIDR和其他集群CIDR冲突。
InternalError.CreateMasterFailed	创建集群失败。
InternalError.Param	Param。
InternalError.CidrConflictWithVpcCidr	CIDR和vpc冲突。
InternalError.PublicClusterOpNotSupport	集群不支持当前操作。
InternalError.DfwGetUSGQuota	获得安全组配额失败。
InternalError	内部错误

开启集群内网/外网访问

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

开启集群内网/外网访问

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-29 11:37:33。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateClusterEndpoint
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
SubnetId	否	否	String	集群端口所在的子网ID（仅在开启非外网访问时需要填，必须为集群所在VPC内的子网） 示例值：
IsExtranet	否	否	Bool	是否为外网访问。true表示外网访问,false表示内网访问 示例值： true
Domain	否	否	String	用户自定义域名 示例值：
SecurityGroup	否	否	String	用户指定安全组 示例值：

参数名称	必选	允许NULL	类型	描述
ExtensiveParameters	否	否	String	运营商类型、网络计费模式、宽带上限(仅支持公网) 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.CamNoAuth	没有权限。
InvalidParameter	参数错误
LimitExceeded	超过配额限制
InternalError.Db	db错误。
UnauthorizedOperation	未授权操作
FailedOperation	操作失败
InvalidParameter.Param	参数错误。
MissingParameter	缺少参数错误
UnsupportedOperation	操作不支持
InternalError.Param	Param。
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

添加虚拟节点至指定集群的指定虚拟节点池

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

添加虚拟节点至指定集群的指定虚拟节点池

默认接口请求频率限制：20次/秒。

接口更新时间：2021-12-09 14:44:11。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateClusterVirtualNode
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
NodePoolId	是	否	String	虚拟节点池id 示例值：
SubnetIds	是	否	Array of String	子网ID数组,与SubnetId必写一个 示例值：
SubnetId	是	否	String	子网id，与SubnetIds必写一个 示例值：

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
InstanceIds	Array of String	虚拟节点数组 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InvalidParameter	参数错误
InternalError	内部错误

在指定集群内创建虚拟节点池

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

在指定集群内创建虚拟节点池

默认接口请求频率限制：20次/秒。

接口更新时间：2021-12-09 14:40:54。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateClusterVirtualNodePool
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Name	是	否	String	虚拟节点池名称 示例值：
SubnetIds	是	否	Array of String	子网ID数组 示例值：
SecurityGroupIds	是	否	Array of String	安全组数组 示例值：
Labels	是	否	Array of Label	虚拟节点池的Labels 示例值：查看
Taints	是	否	Array of Taint	虚拟节点池的Taints 示例值：查看

参数名称	必选	允许NULL	类型	描述
Unschedulable	否	否	Bool	节点池中的超级节点是否不可调度。true表示不可调度，false表示可调度 示例值： true

3. 输出参数

参数名称	类型	描述
NodePoolId	String	虚拟节点池ID 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InternalError	内部错误
InvalidParameter	参数错误

创建集群日志收集规则(旧版)

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

创建集群日志收集规则(旧版)

默认接口请求频率限制：20次/秒。

接口更新时间：2022-04-14 17:32:25。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateLogCollector
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Name	是	否	String	日志采集规则 示例值：
Description	否	否	String	日志收集器描述 示例值：
InputType	否	否	String	日志收集规则的输入类型。传container-log表示收集容器的日志，传host-log表示收集机器的日志。 示例值： container-log
OutputType	否	否	String	日志收集规则的输出类型。传ckafka表示日志收集到ckafka，传kafka表示日志收集到

参数名称	必选	允许NULL	类型	描述
				kafka，传cls表示日志收集到cls。 示例值： cls
InputOption	否	否	LogInputOption	输入选项 示例值：查看
OutputOption	否	否	LogOutputOption	输出选项 示例值：查看

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
UnknownParameter	未知参数错误
InvalidParameter	参数错误

删除集群

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

删除集群

默认接口请求频率限制：50次/秒。

接口更新时间：2020-07-29 11:01:50。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteCluster
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
InstanceDeleteMode	是	否	String	集群实例删除时的策略：terminate（销毁实例，仅支持按量计费云主机实例） retain （仅移除，保留实例） 示例值： terminate

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
ResourceUnavailable	资源不可用
InternalError.PublicClusterOpNotSupport	集群不支持当前操作。
FailedOperation	操作失败
InternalError	内部错误
InvalidParameter	参数错误
ResourceNotFound	资源不存在
InternalError.Param	Param。

删除集群指定虚拟节点

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

删除集群指定虚拟节点

默认接口请求频率限制：20次/秒。

接口更新时间：2021-12-09 15:53:27。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteClusterVirtualNode
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
NodePoolId	是	否	String	节点池id 示例值：
Force	否	否	Bool	是否强制删除虚拟节点。true表示强制删除, false表示节点上若有pod运行就不删除 示例值： true
NodeNames	否	否	Array of String	虚拟节点名 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InternalError	内部错误

删除集群指定虚拟节点池

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

删除集群指定虚拟节点池

默认接口请求频率限制：20次/秒。

接口更新时间：2021-12-09 15:52:05。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteClusterVirtualNodePool
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
NodePoolIds	是	否	Array of String	节点池id 示例值：
Force	否	否	Bool	是否强制删除虚拟节点池。true表示强制删除节点池, false表示节点池中如果有pod运行就不删除 示例值： true

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InternalError	内部错误

删除Chart

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

删除Chart

默认接口请求频率限制：20次/秒。

接口更新时间：2021-05-13 14:50:51。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteHelmChart
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ChartName	是	否	String	Chart名称 示例值：

3. 输出参数

参数名称	类型	描述
Deleted	Bool	表示chart是否删除成功。true表示删除成功,false表示删除失败 示例值： true
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误

删除Chart指定版本

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

删除Chart指定版本

默认接口请求频率限制：20次/秒。

接口更新时间：2021-05-13 14:54:00。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteHelmChartVersion
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ChartName	是	否	String	Chart名称 示例值：
ChartVersion	是	否	String	Chart版本 示例值：

3. 输出参数

参数名称	类型	描述
Deleted	Bool	表示chart是否删除成功。true表示删除成功,false表示删除失败 示例值： true
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误
UnknownParameter	未知参数错误

删除日志收集规则(旧版)

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

删除集群指定日志收集规则(旧版)

默认接口请求频率限制：20次/秒。

接口更新时间：2022-04-11 16:59:58。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteLogCollector
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Name	是	否	String	日志采集规则名字 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
UnknownParameter	未知参数错误

获取Chart的下载信息

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取Chart的下载信息

默认接口请求频率限制：20次/秒。

接口更新时间：2021-05-18 16:02:26。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeChartDownloadInfo
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ChartName	是	否	String	Chart名称 示例值：
ChartVersion	是	否	String	Chart版本 示例值：

3. 输出参数

参数名称	类型	描述
PreSignedDownloadURL	String	下载地址 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误
UnknownParameter	未知参数错误

查询集群endpoints的信息

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询集群endpoints的信息

默认接口请求频率限制：20次/秒。

接口更新时间：2022-08-24 19:32:45。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterEndpoint
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：

3. 输出参数

参数名称	类型	描述
CertificationAuthority	String	集群CA证书 示例值：
ClusterExternalEndpoint	String	集群外网访问地址（包含端口） 示例值：
ClusterIntranetEndpoint	String	集群内网访问地址（包含端口） 示例值：

参数名称	类型	描述
ClusterDomain	String	集群默认访问域名 示例值：
ClusterExternalDomain	String	集群外网访问域名 示例值：
ClusterIntranetDomain	String	集群内外访问域名 示例值：
SecurityGroup	String	集群外网访问使用的安全组 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
UnknownParameter	未知参数错误

查询集群内网/外网访问端口的状态

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询集群内网/外网访问端口的状态

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-29 11:38:31。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterEndpointStatus
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
IsExtranet	否	否	Bool	是否为外网访问。true表示外网访问,false表示内网访问 示例值： true

3. 输出参数

参数名称	类型	描述
Status	String	查询集群访问端口状态。Creating表示开启中,Created表示开启成功,NotFound表示未开启 示例值： Creating
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.CamNoAuth	没有权限。
InvalidParameter	参数错误
LimitExceeded	超过配额限制
UnauthorizedOperation	未授权操作
FailedOperation	操作失败
InvalidParameter.Param	参数错误。
MissingParameter	缺少参数错误
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
InternalError.Param	Param。
InternalError.VpcCommon	VPC报错。
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

获取集群的kubeconfig

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取集群的kubeconfig

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-08 18:16:00。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterKubeconfig
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Type	否	否	String	返回的config中sever地址类型，INNERLB表示内网地址,INTERNETLB表示外网地址 示例值： INTERNETLB

3. 输出参数

参数名称	类型	描述
Kubeconfig	String	用户的kubeconfig 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误
ResourceNotFound	资源不存在
ResourceUnavailable	资源不可用

查询支持的集群等级

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询支持的集群等级

默认接口请求频率限制：20次/秒。

接口更新时间：2024-03-04 17:16:10。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterLevelAttribute
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	否	否	String	集群ID 示例值：

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	总数 示例值：
Items	ClusterLevelAttribute	集群等级的相关属性 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误

查询集群的密钥信息

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询集群的密钥信息

默认接口请求频率限制：50次/秒。

接口更新时间：2021-04-29 11:33:50。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterSecurity
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群 ID，请填写 查询集群列表 接口中返回的 clusterId 字段 示例值：
JnsGwEndpointEnable	否	否	Bool	是否返回内网地址。true表示返回内网地址,false表示不返回内网地址 示例值： true

3. 输出参数

参数名称	类型	描述
UserName	String	集群的账号名称 示例值：

参数名称	类型	描述
Password	String	集群的访问密码 示例值：
CertificationAuthority	String	集群访问CA证书 示例值：
ClusterExternalEndpoint	String	集群访问的地址 示例值：
Domain	String	集群访问的域名 示例值：
PgwEndpoint	String	集群Endpoint地址 示例值：
SecurityPolicy	Array of String	集群访问策略组 示例值：
Kubeconfig	String	用户的kubeconfig文件信息 示例值：
JnsGwEndpoint	String	集群的jnsgw地址 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.DbAffectivedRows	DB错误
InternalError.DbRecordNotFound	记录未找到。
LimitExceeded	超过配额限制
InternalError.Db	db错误。
UnauthorizedOperation	未授权操作

错误码	描述
FailedOperation	操作失败
InternalError.AccountUserNotAuthenticated	账户未通过认证。
MissingParameter	缺少参数错误
ResourceUnavailable	资源不可用
UnsupportedOperation	操作不支持
InternalError.Param	Param。
ResourceInUse	资源被占用
UnknownParameter	未知参数错误
ResourceNotFound	资源不存在

获取集群Service列表

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

该接口获取集群内Service相关的详细描述信息，参考kubernetes API获取Service，只对内部短期使用

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-09 10:40:30。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterServices
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Namespace	否	否	String	命名空间 示例值：
AllNamespace	否	否	Uint64	是否使用所有命名空间。true表示所有命名空间,false表示不使用所有命名空间 示例值： true

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
TotalCount	Int64	返回的Service总数 示例值：
Services	Array of SummaryService	Service详细信息 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.ClusterNotFound	集群ID不存在。
InvalidParameter.Param	参数错误。
InternalError.UnexceptedInternal	内部错误

获取集群指定虚拟节点池的节点列表

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取集群指定虚拟节点池的节点列表

默认接口请求频率限制：20次/秒。

接口更新时间：2021-12-09 15:13:10。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterVirtualNode
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
NodePoolId	是	否	String	节点池id 示例值：
NodeNames	否	否	Array of String	NodeNames数组 示例值：

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	虚拟节点数量 示例值：

参数名称	类型	描述
Nodes	Array of Node	虚拟节点数组 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InternalError	内部错误

获取集群的虚拟节点池列表

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取集群的虚拟节点池列表

默认接口请求频率限制：20次/秒。

接口更新时间：2021-12-09 15:00:43。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusterVirtualNodePools
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	虚拟节点池数量 示例值：
NodePoolSet	Array of NodePoolSet	虚拟节点池数组 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InternalError	内部错误

查询集群列表

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询集群列表

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-09 10:26:16。

接口只验签名不鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeClusters
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterIds	否	否	Array of String	集群ID列表(为空时， 表示获取账号下所有集群) 示例值：
Offset	否	否	Int64	偏移量,默认0 示例值：
Limit	否	否	Int64	最大输出条数，默认20，最大为100 示例值：
Filters	否	否	Array of Filter	过滤条件,当前只支持按照单个条件ClusterName进行过滤 示例值：查看

3. 输出参数

参数名称	类型	描述
TotalCount	Int64	集群总个数 示例值：
Clusters	Array of Cluster	集群信息列表 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.DbAffectivedRows	DB错误
InternalError.CamNoAuth	没有权限。
InvalidParameter	参数错误
LimitExceeded	超过配额限制
InternalError.QuotaMaxClsLimit	超过配额限制。
InternalError.Db	db错误。
InternalError.QuotaMaxNodLimit	超过配额限制。
InvalidParameter.Param	参数错误。
InternalError.Param	Param。
InternalError.PublicClusterOpNotSupport	集群不支持当前操作。
ResourceNotFound	资源不存在

获取chart上传cos信息

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取chart上传cos信息

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-20 16:48:25。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeCosInfo
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	参数错误

错误码	描述
InternalError	内部错误

获取chart列表

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取chart列表

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-20 16:45:47。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeHelmChart
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
Search	是	否	String	搜索内容 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	参数错误
InternalError	内部错误

获取chart详情

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取chart详情

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-20 16:45:11。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeHelmChartDetail
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ChartName	是	否	String	Chart名称 示例值：
ChartVersion	是	否	String	Chart版本 示例值：

3. 输出参数

参数名称	类型	描述
FileEntry	FileEntry	文件 示例值：查看
Chart	Chart	chart 示例值：查看

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	参数错误
InternalError	内部错误

获取chart版本列表

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取chart版本列表

默认接口请求频率限制：20次/秒。

接口更新时间：2021-05-12 16:46:29。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeHelmChartVersion
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ChartName	是	否	String	Chart名称 示例值：

3. 输出参数

参数名称	类型	描述
Results	Array of Result	Results 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误

驱逐集群指定虚拟节点

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

驱逐集群指定虚拟节点

默认接口请求频率限制：20次/秒。

接口更新时间：2021-12-09 15:43:13。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DrainClusterVirtualNode
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
NodeName	是	否	String	节点名称 示例值：
DryRun	否	否	Bool	在驱逐节点前是否预检查。true表示不真正驱逐，只反馈节点上运行的pod信息, false表示不进行预检查直接驱逐节点 示例值： true

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
TotalCount	Uint64	虚拟节点数量 示例值：
Pods	Array of SimplePodInfo	虚拟节点数组 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InternalError	内部错误

启用指定集群的日志收集服务（旧版）

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

启用指定集群的日志收集服务（旧版）

默认接口请求频率限制：20次/秒。

接口更新时间：2022-04-14 18:17:40。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：EnableLogCollector
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	否	否	String	集群ID 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InvalidParameter	参数错误

获取集群指定日志收集规则的信息(旧版)

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取集群指定日志收集规则的信息(旧版)

默认接口请求频率限制：20次/秒。

接口更新时间：2022-04-11 20:07:53。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：GetLogCollector
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Name	是	否	String	日志采集规则名字 示例值：

3. 输出参数

参数名称	类型	描述
LogCollector	LogCollector	日志采集规则 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
UnknownParameter	未知参数错误

获取集群日志采集（旧版）启用状态

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

获取集群日志采集（旧版）启用状态

默认接口请求频率限制：20次/秒。

接口更新时间：2022-04-15 14:54:32。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：GetLogCollectorStatus
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：

3. 输出参数

参数名称	类型	描述
Enabled	Bool	是否开启了日志采集。true表示开启了日志采集，false表示没有开启日志采集 示例值： true
CurrentNumberScheduled	Int64	当前POD运行数量 示例值：

参数名称	类型	描述
DesiredNumberScheduled	Int64	期望POD运行数量 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
UnknownParameter	未知参数错误

查询价格

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询价格

默认接口请求频率限制：20次/秒。

接口更新时间：2024-01-03 18:33:05。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：GetPrices
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ProductCode	是	否	String	产品ID 示例值：
SubProductCode	是	否	String	子产品ID 示例值：
PayMode	否	否	Uint64	付费模式,传0表示后付费模式,传1表示预付费模式 示例值： 0
Cenario	否	否	String	预付费专用(预付费会区分不同场景) 示例值：
ZoneID	否	否	String	如果不存在,默认取第一个可用区 示例值：
Parts	是	否	String	配置信息:map 示例值：

参数名称	必选	允许NULL	类型	描述
GoodsNum	是	否	Uint64	商品数量 示例值：
TimeSpan	是	否	Uint64	购买时长 单位秒 示例值：

3. 输出参数

参数名称	类型	描述
RealTotalCost	Float	实际金额(单位分) 示例值：
TotalCost	Float	原价(单位分) 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误

查询集群日志收集列表(旧)

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询集群日志收集列表(旧)

默认接口请求频率限制：20次/秒。

接口更新时间：2022-04-07 20:34:07。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ListLogCollector
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Limit	否	否	Int64	最大输出条数，默认20 示例值：
Offset	否	否	Int64	偏移量,默认0 示例值：
Search	否	否	String	过滤条件,当前只支持按照单个条件"日志收集名称"进行过滤 示例值：

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
TotalCount	Int64	日志收集总个数 示例值：
LogCollectorList	Array of LogCollector	日志收集信息表 示例值：查看
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
UnknownParameter	未知参数错误

修改集群属性

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

修改集群属性

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-27 15:39:11。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyClusterAttribute
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
ClusterName	否	否	String	集群名称 示例值：
ClusterDesc	否	否	String	集群描述 示例值：
ClusterRunTime	否	否	String	集群运行时名称。docker表示docker运行时,containerd表示containerd运行时 示例值： docker

参数名称	必选	允许NULL	类型	描述
ClusterLevel	否	否	String	集群等级 示例值：
AutoUpgradeClusterLevel	否	是	AutoUpgradeClusterLevel	自动升级集群等级 示例值：查看

3. 输出参数

参数名称	类型	描述
ProjectId	Int64	集群所属项目 示例值：
ClusterName	String	集群名称 示例值：
ClusterDesc	String	集群描述 示例值：
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
InternalError.DbAffectivedRows	DB错误
InternalError.DbRecordNotFound	记录未找到。
InternalError.CamNoAuth	没有权限。
InternalError.Db	db错误。
InvalidParameter.Param	参数错误。
MissingParameter	缺少参数错误
ResourceUnavailable	资源不可用

错误码	描述
InternalError.Param	Param。
ResourceInUse	资源被占用
ResourceNotFound	资源不存在

修改虚拟节点池

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

修改虚拟节点池

默认接口请求频率限制：20次/秒。

接口更新时间：2021-12-09 15:54:36。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyClusterVirtualNodePool
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Name	否	否	String	虚拟节点池名称 示例值：
Labels	否	否	Array of Label	虚拟节点池的Labels 示例值：查看
Taints	否	否	Array of Taint	虚拟节点池的Taints 示例值：查看
Unschedulable	否	否	Bool	虚拟节点池是否不可调度。true表示不可调度，即节点池中的超级节点默认不可调度,false表示可以调度 示例值： true

参数名称	必选	允许NULL	类型	描述
NodePoolId	是	否	String	节点池id 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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
UnknownParameter	未知参数错误
InternalError	内部错误
InvalidParameter	参数错误

查询当前TKE管控集群的cpu总核数是否超过限制

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

查询当前TKE管控集群的cpu总核数是否超过限制

默认接口请求频率限制：20次/秒。

接口更新时间：2022-06-13 16:20:04。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：QueryOverage
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表

3. 输出参数

参数名称	类型	描述
Pass	String	表示license配额是否超用,超用的话不允许新建集群/添加节点等操作。 取值"true"表示配额没有超用，取值"false"表示配额已经超用 示例值： true
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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.Param	Param。
MissingParameter	缺少参数错误
InvalidParameter	参数错误

更新集群指定日志收集规则(旧版)

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

更新集群指定日志收集规则(旧版)

默认接口请求频率限制：20次/秒。

接口更新时间：2022-04-14 17:30:05。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：UpdateLogCollector
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
ClusterId	是	否	String	集群ID 示例值：
Name	是	否	String	日志规则名字 示例值：
Description	否	否	String	日志规则描述 示例值：
InputType	否	否	String	日志收集规则的输入类型。传container-log表示收集容器的日志，传host-log表示收集机器的日志。 示例值： container-log
OutputType	否	否	String	日志收集规则的输出类型。传ckafka表示日志收集到ckafka，传kafka表示日志收集到

参数名称	必选	允许NULL	类型	描述
				kafka，传cls表示日志收集到cls。 示例值： cls
InputOption	否	否	LogInputOption	输入选项 示例值：查看
OutputOption	否	否	LogOutputOption	输出选项 示例值：查看

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	内部错误
UnknownParameter	未知参数错误

上传Chart

1. 接口描述

接口请求域名： tke.api3.finance.cloud.tencent.com。

上传Chart

默认接口请求频率限制：20次/秒。

接口更新时间：2021-04-20 16:46:31。

接口既验签名又鉴权。

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	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：UploadHelmChart
Version	是	否	String	公共参数，本接口取值：2018-05-25
Region	是	否	String	公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表
CosPath	是	否	String	cos文件地址 示例值：
NamespaceName	是	否	String	namespace 默认去UIN 示例值：
RegistryId	是	否	String	namespace 默认取APPID 示例值：

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 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	参数错误
InternalError	内部错误

数据结构

ClusterAsGroup

集群关联的伸缩组信息

被如下接口引用：DescribeClusterAsGroups

名称	必选	允许NULL	类型	描述
AutoScalingGroupId	是	否	String	伸缩组ID 示例值：
Status	是	否	String	伸缩组状态(开启 enabled 开启中 enabling 关闭 disabled 关闭中 disabling 更新中 updating 删除中 deleting 开启缩容中 scaleDownEnabling 关闭缩容中 scaleDownDisabling) 示例值： enabled
IsUnschedulable	是	是	Bool	节点是否设置成不可调度 示例值： true
Labels	是	是	Array of Label	伸缩组的label列表 示例值：查看
CreatedTime	是	是	Datetime	创建时间 示例值：

ExistedInstance

已经存在的实例信息

被如下接口引用：DescribeExistedInstances

名称	必选	允许NULL	类型	描述
Usable	是	是	Bool	实例是否支持加入集群(true 可以加入 false 不能加入) 示例值： true
UnusableReason	是	是	String	实例不支持加入的原因。 示例值：
AlreadyInCluster	是	是	String	实例已经所在的集群ID。 示例值：
InstanceId	是	否	String	实例ID形如：ins-xxxxxxxx。 示例值：
InstanceName	是	是	String	实例名称。 示例值：
PrivateIpAddresses	是	是	Array of String	实例主网卡的内网IP列表。 示例值：

名称	必选	允许NULL	类型	描述
PublicIpAddresses	是	是	Array of String	实例主网卡的公网IP列表。 注意：此字段可能返回 null，表示取不到有效值。 示例值：
CreatedTime	是	是	Datetime_iso	创建时间。按照ISO8601标准表示，并且使用UTC时间。格式为：YYYY-MM-DDThh:mm:ssZ。 示例值：
InstanceChargeType	是	是	String	实例计费模式。取值范围： PREPAID：表示预付费，即包年包月 POSTPAID_BY_HOUR：表示后付费，即按量计费 CDHPAID：CDH付费，即只对CDH计费，不对CDH上的实例计费。 示例值： PREPAID
CPU	是	是	Uint64	实例的CPU核数，单位：核。 示例值：
Memory	是	是	Uint64	实例内存容量，单位：GB。 示例值：
OsName	是	是	String	操作系统名称。 示例值：
InstanceType	是	是	String	实例机型。 示例值：
IPv6Addresses	否	是	Array of String	ipv6地址 示例值：

ExistedInstancesPara

已存在实例的重装参数

被如下接口引用：CreateCluster

名称	必选	允许NULL	类型	描述
InstanceIds	是	否	Array of String	集群ID 示例值：
InstanceAdvancedSettings	否	否	InstanceAdvancedSettings	实例额外需要设置参数信息 示例值：查看
EnhancedService	否	否	String	增强服务。通过该参数可以指定是否开启云安全、云监控等服务。若不指定该参数，则默认开启云监控、云安全服务。 示例值：
LoginSettings	否	否	String	节点登录信息（目前仅支持使用Password或者单个KeyIds） 示例值：

名称	必选	允许NULL	类型	描述
SecurityGroupIds	否	否	Array of String	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：
HostName	否	否	String	重装系统时，可以指定修改实例的 HostName(集群为HostName模式时，此参数必传，规则名称除不支持大写字符外与RunInstances接口HostName一致) 示例值：

CertificateInfo

客户证书信息

被如下接口引用：ListClusterCertificates

名称	必选	允许NULL	类型	描述
Kubeconfig	是	否	String	kubeconfig信息 示例值：
CertificateStatus	是	否	String	证书状态 示例值：
ClientCertificate	是	否	String	证书certificate 示例值：
ClientKey	是	否	String	证书key 示例值：
NickName	是	否	String	对应用户的昵称 示例值：
SubAccountUin	是	否	String	用户的uin 示例值：
ExpirationTime	是	否	Datetime_iso	证书过期时间 示例值：

NotReadyPodsItem

集群某个命名空间下NotReady的Pod集合

被如下接口引用：DescribeClusterHealthyStatus

名称	必选	允许NULL	类型	描述
Namespace	是	否	String	命名空间名称 示例值：

名称	必选	允许NULL	类型	描述
Pods	是	是	Array of String	pod列表 示例值：

LogCollector

日志收集信息

被如下接口引用：GetLogCollector、ListLogCollector

名称	必选	允许NULL	类型	描述
Name	否	否	String	日志收集名称 示例值：
Description	否	否	String	日志收集描述 示例值：
ClusterId	否	否	String	集群ID 示例值：
ClusterName	否	否	String	集群名称 示例值：
CreatedAt	否	否	String	创建时间 示例值：
InputType	否	否	String	日志收集规则的输入类型。传container-log表示收集容器的日志，传host-log表示收集机器的日志。 示例值： container-log
OutputType	否	否	String	日志收集规则的输出类型。传ckafka表示日志收集到ckafka，传kafka表示日志收集到kafka，传cls表示日志收集到cls。 示例值： cls
InputOption	否	是	LogInputOption	输入选项 示例值：查看
OutputOption	否	是	LogOutputOption	输出选项 示例值：查看

RunSecurityServiceEnabled

描述了“云安全”服务相关的信息

被如下接口引用：AddExistedInstances、UpgradeClusterInstances

名称	必选	允许NULL	类型	描述
Enabled	否	否	Bool	是否开启云安全服务。 true表示开启云安全服务,false表示不开启云安全服务 示例值： true

Tag

标签绑定的资源类型，当前支持类型："cluster"

被如下接口引用：CreateCluster、DescribeClusters

名称	必选	允许NULL	类型	描述
Key	否	否	String	标签键 示例值：
Value	否	否	String	标签值 示例值：

Flag

参数描述

被如下接口引用：DescribeClusterAvailableExtraArgs

名称	必选	允许NULL	类型	描述
Name	是	否	String	参数名 示例值：
Type	是	否	String	参数类型 示例值：
Usage	是	否	String	参数描述 示例值：
Default	是	否	String	参数默认值 示例值：
Constraint	是	否	String	参数可选范围（目前包含range和in两种，"[]"代表range，如"[1, 5]"表示参数必须>=1且 <=5, "()"代表in， 如"('aa', 'bb')"表示参数只能为字符串'aa'或者'bb'，该参数为空表示不校验） 示例值：

ClusterBasicSettings

描述集群的基本配置信息

被如下接口引用：CreateCluster

名称	必选	允许NULL	类型	描述
ClusterOs	否	否	String	集群系统。centos7.2x86_64 或者 ubuntu16.04.1 LTSx86_64，默认取值为 ubuntu16.04.1 LTSx86_64 示例值：

名称	必选	允许NULL	类型	描述
ClusterVersion	否	否	String	集群版本,默认值为1.10.5 示例值：
ClusterName	否	否	String	集群名称 示例值：
ClusterDescription	否	否	String	集群描述 示例值：
VpcId	否	否	String	私有网络ID，形如vpc-xxx。创建托管空集群时必传。 示例值：
ProjectId	否	否	Int64	集群内新增资源所属项目ID。 示例值：
TagSpecification	否	否	Array of TagSpecification	标签描述列表。通过指定该参数可以同时绑定标签到相应的资源实例，当前仅支持绑定标签到集群实例。 示例值：查看
OsCustomizeType	否	否	String	容器的镜像版本，"DOCKER_CUSTOMIZE"(容器定制版),"GENERAL"(普通版本，默认值) 示例值： GENERAL
NeedWorkSecurityGroup	否	否	Bool	是否开启节点的默认安全组(默认: 否) 示例值： false
AutoUpgradeClusterLevel	否	是	AutoUpgradeClusterLevel	集群自动变配 示例值：查看
ClusterLevel	否	是	String	托管集群的集群等级 示例值：

fileEntry

chart返回值

被如下接口引用：DescribeHelmChartDetail

名称	必选	允许NULL	类型	描述
Name	是	是	String	名称 示例值：
Value	否	是	String	chart值 示例值：
ID	否	是	Int64	id 示例值：
Children	否	是	Array of FileEntry	fileEntry 示例值：查看

Filter

过滤器

被如下接口引用：CheckInstancesUpgradeAble、DescribeClusterInstanceIds、DescribeClusterInstances、DescribeClusters、DescribeExistedInstances

名称	必选	允许NULL	类型	描述
Name	是	否	String	属性名称, 若存在多个Filter时，Filter间的关系为逻辑与（AND）关系。 当前只支持按照单个条件ClusterName进行过滤 示例值：
Values	是	否	Array of String	属性值, 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。 示例值：

ClusterInstancesVersion

集群中worker节点不同版本节点数统计信息

被如下接口引用：DescribeInstancesVersion

名称	必选	允许NULL	类型	描述
ClusterId	是	否	String	集群ID 示例值：
ClusterVersion	是	否	String	集群版本 示例值：
InstanceVersions	是	否	Array of ClusterInstancesVersionItem	节点版本统计信息 示例值：查看
Error	是	是	String	出错信息 示例值：
UpgradeAble	是	是	Bool	是否存在可升级节点。true表示存在，false表示不存在 示例值： true

Label

k8s中标签，一般以数组的方式存在

被如下接口引用：AddExistedInstances、CreateCluster、CreateClusterAsGroup、CreateClusterInstances、CreateClusterVirtualNodePool、CreateLogCollector、DescribeClusterAsGroups、DescribeClusterInstances、DescribeClusterVirtualNodePools、GetLogCollector、ListLogCollector、ModifyClusterVirtualNodePool、UpdateLogCollector、UpgradeClusterInstances

名称	必选	允许NULL	类型	描述

名称	必选	允许NULL	类型	描述
Name	是	否	String	map表中的Name 示例值：
Value	是	否	String	map表中的Value 示例值：

InspectionReportItem

健康检查结果项目

被如下接口引用：DescribeClusterInspectionReport

名称	必选	允许NULL	类型	描述
Name	是	否	String	名称 示例值：
Error	是	否	String	错误信息 示例值：

SimplePodInfo

驱逐节点返回值

被如下接口引用：DrainClusterNode、DrainClusterVirtualNode

名称	必选	允许NULL	类型	描述
Name	是	否	String	pod名称 示例值：
Namespace	是	否	String	pod所属命名空间 示例值：
NodeIp	是	否	String	所属节点IP 示例值：

LogInputOption

日志输入描述

被如下接口引用：CreateLogCollector、GetLogCollector、ListLogCollector、UpdateLogCollector

名称	必选	允许NULL	类型	描述
Path	否	否	String	路径 示例值：

名称	必选	允许NULL	类型	描述
Labels	否	否	Label	标签 类型为map 键值对 示例值：查看
AllNamespaces	否	否	Bool	是否是全部命名空间。传true表示所有的命名空间，传false表示不是所有的命名空间 示例值： true
Namespaces	否	否	LogInputOptNamespace	命名空间 示例值：查看

ClusterAdvancedSettings

集群高级配置

被如下接口引用：CreateCluster

名称	必选	允许NULL	类型	描述
IPVS	否	否	Bool	是否启用IPVS 示例值： false
AsEnabled	否	否	Bool	是否启用集群节点自动扩缩容(创建集群流程不支持开启此功能) 示例值： false
ContainerRuntime	否	否	String	集群使用的runtime类型，包括"docker"和"containerd"两种类型，默认为"docker" 示例值： docker
NodeNameType	否	否	String	集群中节点NodeName类型（包括 hostname,lan-ip两种形式，默认为lan-ip。如果开启了hostname模式，创建节点时需要设置HostName参数，并且InstanceName需要和HostName一致） 示例值： lan-ip
ExtraArgs	否	否	ClusterExtraArgs	集群自定义参数 示例值：查看
NetworkType	否	否	String	集群网络类型（包括GR(全局路由)和VPC-CNI两种模式，默认为GR。 示例值： GR
IsNonStaticIpMode	否	否	Bool	集群VPC-CNI模式是否为非固定IP，默认: FALSE 固定IP。 示例值： false
IsDualStack	否	否	String	集群是否支持双栈,true表示支持双栈，false表示不支持双栈，默认为false 示例值： false

InstanceExtraArgs

节点自定义参数

被如下接口引用：AddExistedInstances、CreateCluster、CreateClusterAsGroup、CreateClusterInstances、DescribeClusterInstances、UpgradeClusterInstances

名称	必选	允许NULL	类型	描述
Kubelet	否	是	Array of String	kubelet自定义参数 示例值：

Product

云产品License用量列表

被如下接口引用：CollectAllCore

名称	必选	允许NULL	类型	描述
ProductCode	是	否	String	产品显示名称， 用于在License管理页面进行展示，对应License 管理-用量列表中的"产品>名称" 示例值： "TKE"
LicenseItemList	是	否	Array of Product	约束项用量列表 示例值：查看

RunMonitorServiceEnabled

描述了 “云监控” 服务相关的信息

被如下接口引用：AddExistedInstances、UpgradeClusterInstances

名称	必选	允许NULL	类型	描述
Enabled	否	否	Bool	是否开启云监控服务。 true表示开启云监控服务,false表示不开启云监控服务 示例值： true

InstanceAdvancedSettings

描述了k8s集群相关配置与信息。

被如下接口引用：AddExistedInstances、CreateCluster、CreateClusterAsGroup、CreateClusterInstances、DescribeClusterInstances、UpgradeClusterInstances

名称	必选	允许NULL	类型	描述
MountTarget	否	是	String	数据盘挂载点, 默认不挂载数据盘. 已格式化的 ext3，ext4，xfs 文件系统的数据盘将直接挂载，其他文件系统或未格式化的数据盘将自动格式化为ext4 并挂载，请注意备份数据! 无数据盘或有多块数据盘的云主机此设置不生效。 示例值：

名称	必选	允许NULL	类型	描述
DockerGraphPath	否	是	String	dockerd --graph 指定值, 默认为 /var/lib/docker 示例值：
UserScript	否	是	String	base64 编码的用户脚本, 此脚本会在 k8s 组件运行后执行, 需要用户保证脚本的可重入及重试逻辑, 脚本及其生成的日志文件可在节点的 /data/ccs_userscript/ 路径查看, 如果要求节点需要在进行初始化完成后才可加入调度, 可配合 unschedulable 参数使用, 在 userScript 最后初始化完成后, 添加 kubectl uncordon nodename --kubeconfig=/root/.kube/config 命令使节点加入调度 示例值：
Unschedulable	否	是	Int64	设置加入的节点是否参与调度，默认值为0，表示参与调度；非0表示不参与调度, 待节点初始化完成之后, 可执行 kubectl uncordon nodename使node加入调度. 示例值： 0
Labels	否	是	Array of Label	节点Label数组 示例值：查看
DataDisks	否	是	Array of DataDisk	数据盘相关信息 示例值：查看
ExtraArgs	否	是	InstanceExtraArgs	节点相关的自定义参数信息 示例值：查看
PreStartUserScript	否	是	String	base64 编码的用户脚本, 此脚本会在 k8s 组件运行前执行, 需要用户保证脚本的可重入及重试逻辑, 脚本及其生成的日志文件可在节点的 /data/ccs_userscript/ 路径查看. 示例值：

NodePoolOption

加入存量节点时的节点池选项

被如下接口引用：AddExistedInstances

名称	必选	允许NULL	类型	描述
AddToNodePool	否	否	Bool	是否加入节点池 示例值：
NodePoolId	否	否	String	节点池id 示例值：

Metadata

Metadata

被如下接口引用：DescribeHelmChartDetail

名称	必选	允许NULL	类型	描述
APIVersion	否	是	String	api版本 示例值：
AppVersion	否	是	String	app版本 示例值：
Description	否	是	String	描述 示例值：
Name	否	是	String	名称 示例值：
Version	否	是	String	版本 示例值：

Cluster

集群信息结构体

被如下接口引用：DescribeClusters

名称	必选	允许NULL	类型	描述
ClusterId	是	否	String	集群ID 示例值：
ClusterName	是	否	String	集群名称 示例值：
ClusterDescription	是	否	String	集群描述 示例值：
ClusterVersion	是	否	String	集群版本（默认值为1.10.5） 示例值：
ClusterOs	是	否	String	集群系统。centos7.2x86_64 或者 ubuntu16.04.1 LTSx86_64，默认取值为ubuntu16.04.1 LTSx86_64 示例值：
ClusterType	是	否	String	集群类型，托管集群：MANAGED_CLUSTER，独立集群：INDEPENDENT_CLUSTER。 示例值： INDEPENDENT_CLUSTER
ClusterNetworkSettings	是	否	ClusterNetworkSettings	集群网络相关参数 示例值：查看
ClusterNodeNum	是	否	Uint64	集群当前node数量 示例值：
ProjectId	是	否	Uint64	集群所属的项目ID 示例值：

名称	必选	允许NULL	类型	描述
TagSpecification	是	是	Array of TagSpecification	标签描述列表。 示例值：查看
ClusterStatus	是	否	String	集群状态 (Running 运行中 Creating 创建中 Abnormal 异常) 示例值： Running
Property	是	是	String	集群属性(包括集群不同属性的MAP，属性字段包括NodeNameType (lan-ip模式和hostname 模式，默认无lan-ip模式)) 示例值：
ClusterMaterNodeNum	是	否	Uint64	集群当前master数量 示例值：
ImageId	是	是	String	集群使用镜像id 示例值：
OsCustomizeType	是	是	String	OsCustomizeType 示例值：
ContainerRuntime	是	是	String	集群运行环境docker或container 示例值： docker
CreatedTime	是	是	String	创建时间 示例值：
ClusterArch	是	是	String	创建的集群架构,取值: x86/arm 示例值： x86
DirectAccess	否	是	String	pod直通模式 示例值：
ClusterOsAlias	否	是	String	集群操作系统名称 示例值：
IsDualStack	否	是	String	是否为双栈集群 示例值：
Ipv6ServiceCidr	否	是	String	Ipv6 Service Cidr 示例值：
ClusterLevel	否	是	String	集群等级,仅支持托管集群 示例值：

RunClusterInspectionResponseItem

触发集群巡检请求返回

被如下接口引用：RunClusterInspections

名称	必选	允许NULL	类型	描述
ClusterId	是	否	String	集群实例id

名称	必选	允许NULL	类型	描述
				示例值：
Error	是	是	String	如果请求未能正常被处理，则Error中将包含错误信息 示例值：

TagSpecification

标签描述列表。通过指定该参数可以同时绑定标签到相应的资源实例，当前仅支持绑定标签到云主机实例。

被如下接口引用：CreateCluster、DescribeClusters

名称	必选	允许NULL	类型	描述
ResourceType	否	是	String	标签绑定的资源类型，当前支持类型："cluster" 示例值：
Tags	否	是	Array of Tag	标签对列表 示例值：查看

SwitchSet

运维配置

被如下接口引用：DescribeLogSwitches

名称	必选	允许NULL	类型	描述
ClusterId	是	否	String	集群ID 示例值：
Log	是	是	DescribeLogSwitchInfo	日志运维 示例值：查看
Event	是	是	DescribeLogSwitchInfo	事件运维 示例值：查看
Audit	是	是	DescribeLogSwitchInfo	审计运维 示例值：查看

LogOutputOption

日志收集输出选项

被如下接口引用：CreateLogCollector、GetLogCollector、ListLogCollector、UpdateLogCollector

名称	必选	允许NULL	类型	描述
InstanceId	否	是	String	实例id 示例值：

名称	必选	允许NULL	类型	描述
TopicId	否	是	String	topicID 示例值：
Host	否	是	String	主机hsot 示例值：
Port	否	是	String	端口 示例值：
Topic	否	是	String	topic 示例值：
LogsetId	否	是	String	LogsetId 示例值：

ClusterLevelAttribute

集群等级属性

被如下接口引用：DescribeClusterLevelAttribute

名称	必选	允许NULL	类型	描述
Name	否	否	String	名字 示例值：
Alias	否	否	String	Alias 示例值：
NodeCount	否	否	Uint64	节点数量 示例值：
PodCount	否	否	Uint64	pod数量 示例值：
ConfigMapCount	否	否	Uint64	ConfigMap数量 示例值：
RSCount	否	否	Uint64	RS数量 示例值：
CRDCount	否	否	Uint64	CRD数量 示例值：
OtherCount	否	否	Uint64	其他数量 示例值：
Enable	否	否	Bool	是否启用 示例值：

File

文件

被如下接口引用：DescribeHelmChartDetail

名称	必选	允许NULL	类型	描述
Data	否	是	String	数据 示例值：
Name	否	是	String	名称 示例值：

NotifySettings

告警通知settings

被如下接口引用：AddAlarmPolicy、DescribeAlarmPolicies、ModifyAlarmPolicy

名称	必选	允许NULL	类型	描述
ReceiverGroups	是	否	Array of Int64	告警接收组（用户组） 示例值：
NotifyWay	是	否	Array of String	告警通知方式。SMS表示短信,EMAIL表示邮件,CALL表示电话,WECHAT表示微信 示例值： SMS
PhoneNotifyOrder	否	是	Array of Int64	电话告警顺序。 注：NotifyWay选择CALL，采用该参数。 示例值：
PhoneCircleTimes	否	是	Int64	电话告警次数。 注：NotifyWay选择CALL，采用该参数。 示例值：
PhoneInnerInterval	否	是	Int64	电话告警轮内间隔。单位：秒 注：NotifyWay选择CALL，采用该参数。 示例值：
PhoneCircleInterval	否	是	Int64	电话告警轮外间隔。单位：秒 注：NotifyWay选择CALL，采用该参数。 示例值：
PhoneArriveNotice	否	是	Int64	电话告警触达通知 注：NotifyWay选择CALL，采用该参数。 示例值：

ClusterAsGroupAttribute

集群伸缩组属性

被如下接口引用：ModifyClusterAsGroupAttribute

名称	必选	允许NULL	类型	描述

名称	必选	允许NULL	类型	描述
AutoScalingGroupId	是	否	String	伸缩组ID 示例值：
AutoScalingGroupEnabled	否	否	Bool	是否开启，伸缩组启用停用的时候需要 示例值： true
AutoScalingGroupRange	否	否	AutoScalingGroupRange	伸缩组最大最小实例数，调整伸缩组配置的时候需要 示例值：查看

InspectionStatistic

InspectionStatistic

被如下接口引用：DescribeClusterInspectionOverviews、DescribeClusterInspections

名称	必选	允许NULL	类型	描述
GoodItem	是	否	Uint64	GoodItem 示例值：
WarnItem	是	否	Uint64	WarnItem 示例值：
RiskItem	是	否	Uint64	RiskItem 示例值：
SeriousItem	是	否	Uint64	SeriousItem 示例值：
FailedItem	是	否	Uint64	FailedItem 示例值：

AlarmPolicy

告警策略结构体

被如下接口引用：DescribeAlarmPolicies

名称	必选	允许NULL	类型	描述
AlarmPolicyId	是	否	String	告警策略ID 示例值：
ClusterInstanceId	是	否	String	k8s集群ID 示例值：
Namespace	是	是	String	k8s命名空间 示例值：
WorkloadType	是	是	String	k8s工作负载类型。Deployment表示deployment工作负载，DaemonSet表示DaemonSet工作负载,StatefulSet

名称	必选	允许NULL	类型	描述
				表示StatefulSet工作负载 示例值： Deployment
AlarmPolicySettings	是	否	AlarmPolicySettings	告警策略settings 示例值：查看
NotifySettings	是	否	NotifySettings	告警通知settings 示例值：查看
ShieldSettings	是	否	ShieldSettings	告警屏蔽settings 示例值：查看

CommonNames

用于返回入参列表中请求对应的子用户的CommonName

被如下接口引用：DescribeClusterCommonNames

名称	必选	允许NULL	类型	描述
CN	是	否	String	对应commonName 示例值：
SubaccountUin	是	否	String	子账户id 示例值：

AutoUpgradeClusterLevel

集群等级自动升级

被如下接口引用：CreateCluster、ModifyClusterAttribute

名称	必选	允许NULL	类型	描述
IsAutoUpgrade	否	是	Bool	是否开启自动升级。true表示开启自动升级,false表示不开启自动升级 示例值： true

DataDisk

描述了k8s节点数据盘相关配置与信息。

被如下接口引用：AddExistedInstances、CreateCluster、CreateClusterAsGroup、CreateClusterInstances、DescribeClusterInstances、UpgradeClusterInstances

名称	必选	允许NULL	类型	描述
DiskType	是	否	String	云盘类型。CLOUD_PREMIUM表示高性能云硬盘, CLOUD_BASIC表示普通云硬盘,CLOUD_SSD表示SSD云硬盘 示例值： CLOUD_SSD

名称	必选	允许NULL	类型	描述
FileSystem	是	否	String	云盘格式化的文件系统。ext3表示格式化为ext3,ext4表示格式化为ext4,xfs表示格式化为xfs 示例值： ext4
DiskSize	是	否	Int64	云盘大小(G） 示例值：
AutoFormatAndMount	是	否	Bool	是否自动化格式盘并挂载。true表示云盘会自动进行格式化并且挂载，false表示不会格式化 示例值： true
MountTarget	否	否	String	挂载目录 示例值：

RunInstancesForNode

不同角色的节点配置参数

被如下接口引用：CreateCluster

名称	必选	允许NULL	类型	描述
NodeRole	是	否	String	节点角色，取值:MASTER_ETCD 和 WORKER。MASTER_ETCD只有在创 建 INDEPENDENT_CLUSTER 独立集群时需要指定。 MASTER_ETCD节点数量为3～7，建议为奇数。MASTER_ETCD节点最小配置为4C8G。 WORKER表示工作节点角色 示例值： MASTER_ETCD
RunInstancesPara	是	否	Array of String	CVM创建透传参数，json化字符串格式，详见RunInstances接口，传入公共参数外的其他参数即可，其中ImageId会替换为TKE集群OS对应的镜像。 示例值：
InstanceAdvancedSettingsOverrides	否	否	Array of InstanceAdvancedSettings	节点高级设置，该参数会覆盖集群级别设置的InstanceAdvancedSettings，和上边的RunInstancesPara按照顺序一一对应（当前只对节点自定义参数ExtraArgs生效）。 示例值：查看

DescribeLogSwitchInfo

DescribeLogSwitchInfo

被如下接口引用：DescribeLogSwitches

名称	必选	允许NULL	类型	描述
Enable	是	是	Bool	是否开启。 true:开启，false:未开启 示例值： true
LogsetId	是	是	String	日志集ID 示例值：
TopicId	是	是	String	日志主题ID 示例值：

ClusterStatus

集群状态信息

被如下接口引用：DescribeClusterStatus

名称	必选	允许NULL	类型	描述
ClusterId	是	否	String	集群Id 示例值：
ClusterState	是	否	String	集群状态。Creating表示集群创建中,Running表示集群正常运行,Upgrading表示集群升级中,Abnormal表示集群异常 示例值： Creating
ClusterInstanceState	是	否	String	集群实例状态。Unknown表示非法状态,Creating表示正在创建中,Upgrading表示实例正在升级中,AllNormal表示所有实例正常,AllAbnormal表示所有实例异常，PartialAbnormal表示部分实例异常 示例值： AllNormal
ClusterBMonitor	是	否	Bool	集群是否开启监控。true表示开启了监控，false表示未开启监控 示例值： true
ClusterInitNodeNum	是	否	Int64	集群创建中的节点数，-1表示获取节点状态超时，-2表示获取节点状态失败 示例值：
ClusterRunningNodeNum	是	否	Int64	集群运行中的节点数，-1表示获取节点状态超时，-2表示获取节点状态失败 示例值：
ClusterFailedNodeNum	是	否	Int64	集群异常的节点数，-1表示获取节点状态超时，-2表示获取节点状态失败 示例值：
ClusterClosedNodeNum	是	是	Int64	集群已关机的节点数，-1表示获取节点状态超时，-2表示获取节点状态失败 示例值：

名称	必选	允许NULL	类型	描述
ClusterClosingNodeNum	是	是	Int64	集群关机中的节点数，-1表示获取节点状态超时，-2表示获取节点状态失败 示例值：

Chart

chart列表

被如下接口引用：DescribeHelmChartDetail

名称	必选	允许NULL	类型	描述
Files	否	是	Array of File	文件列表 示例值：查看
Lock	否	是	String	Lock 示例值：
Metadata	否	是	Metadata	metadata 示例值：查看
Schema	否	是	String	Schema 示例值：
Templates	否	是	Array of File	Templates 示例值：查看
Values	否	是	String	Values 示例值：

EnhancedService

描述了实例的增强服务启用情况与其设置，如云安全，云监控等实例 Agent

被如下接口引用：AddExistedInstances、UpgradeClusterInstances

名称	必选	允许NULL	类型	描述
SecurityService	否	否	RunSecurityServiceEnabled	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：查看
MonitorService	否	否	RunMonitorServiceEnabled	开启云安全服务。若不指定该参数，则默认开启云监控服务。 示例值：查看

Evaluator

告警指标判断逻辑结构体

被如下接口引用：AddAlarmPolicy、DescribeAlarmPolicies、ModifyAlarmPolicy

名称	必选	允许NULL	类型	描述
Type	是	否	String	告警判断类型，gt表示大于等于,lt表示小于等于 示例值： gt
Value	是	否	Float	告警设置的阈值 示例值：

AlarmPolicyFilter

告警策略过滤

被如下接口引用：DescribeAlarmPolicies

名称	必选	允许NULL	类型	描述
AlarmPolicyName	是	否	String	告警策略名称 示例值：

SummaryService

Service详细信息

被如下接口引用：DescribeClusterServices

名称	必选	允许NULL	类型	描述
Name	是	是	String	Service名称 示例值：
Status	是	是	String	Service状态 示例值：
ServiceIp	是	是	String	Service IP 示例值：
ExternalIp	是	是	String	外网IP 示例值：
LbId	是	是	String	负载均衡ID 示例值：
LbStatus	是	是	String	负载均衡状态 示例值：
AccessType	是	是	String	Service访问类型 示例值：
DesiredReplicas	是	是	Int64	期望副本数 示例值：

名称	必选	允许NULL	类型	描述
CurrentReplicas	是	是	Int64	当前副本数 示例值：
CreatedAt	是	是	String	创建时间 示例值：
Namespace	是	是	String	命名空间 示例值：
ReasonMap	是	是	String	ReasonMap 示例值：
SourceReasonMap	是	是	String	SourceReasonMap 示例值：
Labels	是	是	String	Labels 示例值：
SysLables	是	是	String	SysLables 示例值：
UserLables	是	是	String	UserLables 示例值：
ScaleType	是	是	String	ScaleType 示例值：
HpaPolicy	是	是	String	HpaPolicy 示例值：

Taint

kubernetes Taint

被如下接口引用：CreateClusterVirtualNodePool、DescribeClusterVirtualNodePools、ModifyClusterVirtualNodePool

名称	必选	允许NULL	类型	描述
Key	否	否	String	Key 示例值：
Value	否	否	String	Value 示例值：
Effect	否	否	String	Effect 示例值：

AlarmPolicySettings

告警策略settings

被如下接口引用：AddAlarmPolicy、DescribeAlarmPolicies、ModifyAlarmPolicy

名称	必选	允许NULL	类型	描述
AlarmPolicyName	是	否	String	告警策略名称 示例值：
AlarmPolicyDescription	否	是	String	告警策略描述 示例值：
AlarmPolicyType	是	否	String	告警类型，cluster表示集群类型,node表示节点类型,pod表示pod类型 示例值： cluster
AlarmObjectsType	否	是	String	告警对象绑定类型。all表示全部对象绑定。part表示只有选择的部分对象绑定。 示例值： all
AlarmMetrics	是	否	Array of AlarmMetric	告警指标列表。 示例值：查看
AlarmObjects	是	是	String	告警对象，多个用逗号分隔。 示例值：

ClusterCIDRSettings

集群容器网络相关参数

被如下接口引用：CreateCluster

名称	必选	允许NULL	类型	描述
ClusterCIDR	否	否	String	用于分配集群容器和服务 IP 的 CIDR，不得与 VPC CIDR 冲突，也不得与同 VPC 内其他集群 CIDR 冲突。且网段范围必须在内网网段内，例如:10.1.0.0/14, 192.168.0.1/18,172.16.0.0/16。 示例值：
IgnoreClusterCIDRConflict	否	否	Bool	是否忽略 ClusterCIDR 冲突错误, true表示忽略，false表示不忽略 示例值： true
MaxNodePodNum	否	否	Uint64	集群中每个Node上最大的Pod数量。取值范围4～256。不为2的幂值时会向上取最接近的2的幂值。 示例值：
MaxClusterServiceNum	否	否	Uint64	集群最大的service数量。取值范围32～32768，不为2的幂值时会向上取最接近的2的幂值。 示例值：
ServiceCIDR	否	否	String	用于分配集群服务 IP 的 CIDR，不得与 VPC CIDR 冲突，也不得与同 VPC 内其他集群 CIDR 冲突。且网段范围必须在内网网段内，例如:10.1.0.0/14, 192.168.0.1/18,172.16.0.0/16。 示例值：
EniSubnetIds	否	否	Array of String	VPC-CNI网络模式下，弹性网卡的子网Id。 示例值：

名称	必选	允许NULL	类型	描述
ClaimExpiredSeconds	否	否	Int64	VPC-CNI网络模式下，弹性网卡IP的回收时间，取值范围[300,15768000) 示例值：

Result

Result

被如下接口引用：DescribeHelmChartVersion

名称	必选	允许NULL	类型	描述
Chart	否	是	String	chart 示例值：
Name	否	是	String	名称 示例值：
Score	否	是	Int64	分数 示例值：

Instance

集群的实例信息

被如下接口引用：DescribeClusterInstances

名称	必选	允许NULL	类型	描述
InstanceId	是	否	String	实例ID 示例值：
InstanceRole	是	否	String	节点角色。 MASTER_ETCD表示master节点信息，WORKER表示work节点信息 示例值： MASTER_ETCD
FailedReason	是	否	String	实例异常(或者处于初始化中)的原因 示例值：
InstanceState	是	否	String	实例的状态。running表示运行中,initializing表示初始化中,failed表示异常 示例值： running
DrainStatus	是	是	String	实例是否封锁状态，true表示封锁状态,false表示非封锁状态 示例值： true
InstanceAdvancedSettings	是	是	InstanceAdvancedSettings	节点配置 示例值：查看

名称	必选	允许NULL	类型	描述
CreatedTime	是	否	String	添加时间 示例值：

ClusterInstancesVersionItem

集群中某个版本的worker节点数目

被如下接口引用：DescribeInstancesVersion

名称	必选	允许NULL	类型	描述
InstanceVersion	是	否	String	版本 示例值：
Total	是	否	Uint64	节点数 示例值：
UpgradeAble	是	是	Bool	是否可升级。true表示可升级,false表示不可升级 示例值： true

ClusterHealthyPodsStatus

集群健康检查中pod的健康情况

被如下接口引用：DescribeClusterHealthyStatus

名称	必选	允许NULL	类型	描述
Total	是	否	Int64	pod总数 示例值：
NotReadyTotal	是	否	Int64	NotReady的pod总数 示例值：
NotReadyPods	是	是	Array of NotReadyPodsItem	NotReady的pod详细信息 示例值：查看

Step

执行步骤信息

被如下接口引用：DescribeClusterCreateProgress

名称	必选	允许NULL	类型	描述
Type	是	否	String	名称 示例值：

名称	必选	允许NULL	类型	描述
LastProbeTime	是	否	String	最后一次执行时间 示例值：
Status	是	否	String	当前状态 示例值：
Message	是	否	String	执行信息 示例值：

AlarmMetric

告警指标结构体

被如下接口引用：AddAlarmPolicy、DescribeAlarmPolicies、ModifyAlarmPolicy

名称	必选	允许NULL	类型	描述
MetricId	否	否	String	告警指标ID 示例值：
Measurement	否	否	String	ARGUS系统Measurement 示例值：
StatisticsPeriod	否	否	Int64	统计周期 示例值：
MetricName	否	否	String	指标名 示例值：
ArgusPolicyName	是	否	String	指标描述 示例值：
Evaluator	是	否	Evaluator	告警判断设置 示例值：查看
ContinuePeriod	是	否	Int64	持续周期 示例值：
Status	是	否	Bool	状态，true表示OK 示例值： true
Unit	否	是	String	指标单位 示例值：

VersionInstance

版本信息

被如下接口引用：DescribeVersions

名称	必选	允许NULL	类型	描述

名称	必选	允许NULL	类型	描述
Name	是	是	String	版本名称 示例值：
Version	是	是	String	版本信息 示例值：
Remark	是	是	String	Remark 示例值：

ClusterInspectionProgress

集群巡检当前进度

被如下接口引用：DescribeClusterInspections

名称	必选	允许NULL	类型	描述
Step	是	否	String	当前步骤名称。init_env: 初始化环境，安装agent。init_k8s_resources：获取kubernetes资源,init_components: 获取核心组件参数,init_machines：获取节点系统参数diagnostic： 正在检查集群 示例值： init_env
Percent	是	否	Float	检查进度百分比 示例值：

ClusterNetworkSettings

集群网络相关的参数

被如下接口引用：DescribeClusters

名称	必选	允许NULL	类型	描述
ClusterCIDR	是	否	String	用于分配集群容器和服务 IP 的 CIDR，不得与 VPC CIDR 冲突，也不得与同 VPC 内其他集群 CIDR 冲突 示例值：
IgnoreClusterCIDRConflict	否	否	Bool	是否忽略 ClusterCIDR 冲突错误, 默认不忽略 示例值：
MaxNodePodNum	否	否	Uint64	集群中每个Node上最大的Pod数量(默认为256) 示例值：
MaxClusterServiceNum	否	否	Uint64	集群最大的service数量(默认为256) 示例值：
Ipvs	否	否	Bool	是否启用IPVS(默认不开启) 示例值：

名称	必选	允许NULL	类型	描述
VpcId	否	否	String	集群的VPCID（如果创建空集群，为必传值，否则自动设置为和集群的节点保持一致） 示例值：
Cni	否	否	Bool	网络插件是否启用CNI(默认开启) 示例值：
SubnetIds	是	否	Array of String	集群网络对应子网 示例值：

ExistedInstancesForNode

不同角色的已存在节点配置参数

被如下接口引用：CreateCluster

名称	必选	允许NULL	类型	描述
NodeRole	是	否	String	节点角色，取值:MASTER_ETCD, WORKER。MASTER_ETCD只有在创建 INDEPENDENT_CLUSTER 独立集群时需要指定。MASTER_ETCD节点数量为3~7，建议为奇数。MASTER_ETCD最小配置为4C8G。 示例值： MASTER_ETCD
ExistedInstancesPara	是	否	ExistedInstancesPara	已存在实例的重装参数 示例值：查看
InstanceAdvancedSettingsOverride	否	否	InstanceAdvancedSettings	节点高级设置，会覆盖集群级别设置的InstanceAdvancedSettings（当前只对节点自定义参数ExtraArgs生效） 示例值：查看

ImageInstance

镜像信息

被如下接口引用：DescribeImages

名称	必选	允许NULL	类型	描述
Alias	是	是	String	镜像别名 示例值：
OsName	是	是	String	操作系统名称 示例值：

名称	必选	允许NULL	类型	描述
ImageId	是	是	String	镜像ID 示例值：
OsCustomizeType	是	是	String	容器的镜像版本，"DOCKER_CUSTOMIZE"(容器定制版),"GENERAL"(普通版本，默认值) 示例值： GENERAL
Arch	是	是	String	镜像的cpu架构，取值: x86/arm 示例值： x86
SportIpv6	否	是	String	表示镜像是否支持ipv6,true表示支持，false表示不支持 示例值： true

AvailableExtraArgs

集群可用的自定义参数

被如下接口引用：DescribeClusterAvailableExtraArgs

名称	必选	允许NULL	类型	描述
KubeAPIServer	是	是	Array of Flag	kube-apiserver可用的自定义参数 示例值：查看
KubeControllerManager	是	是	Array of Flag	kube-controller-manager可用的自定义参数 示例值：查看
KubeScheduler	是	是	Array of Flag	kube-scheduler可用的自定义参数 示例值：查看
Kubelet	是	是	Array of Flag	kubelet可用的自定义参数 示例值：查看

UpgradeNodeResetParam

节点升级重装参数

被如下接口引用：UpgradeClusterInstances

名称	必选	允许NULL	类型	描述
InstanceAdvancedSettings	否	否	InstanceAdvancedSettings	实例额外需要设置参数信息 示例值：查看
EnhancedService	否	否	EnhancedService	增强服务。通过该参数可以指定是否开启云安全、云监控等服务。若不指定该参数，则默认开启云监控、云安全服务。 示例值：查看

版权所有：TCloudFinanceZone

名称	必选	允许NULL	类型	描述
LoginSettings	否	否	LoginSettings	节点登录信息（目前仅支持使用Password或者单个KeyIds） 示例值：查看
SecurityGroupIds	否	否	Array of String	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。（目前仅支持设置单个sgId） 示例值：

LogInputOptNamespace

日志输入命名空间描述

被如下接口引用：CreateLogCollector、GetLogCollector、ListLogCollector、UpdateLogCollector

名称	必选	允许NULL	类型	描述
Namespace	是	否	String	命名空间 示例值：
AllContainers	否	否	Bool	是否为命名空间内的所有容器 示例值：
Services	否	否	Array of String	Services名称数组 示例值：

Node

虚拟节点

被如下接口引用：DescribeClusterVirtualNode

名称	必选	允许NULL	类型	描述
Name	否	否	String	虚拟节点名 示例值：
NodePoolId	否	否	String	节点池id 示例值：
SubnetId	否	否	String	节点子网id 示例值：
Phase	否	否	String	节点状态。Pending表示节点pending，Running表示节点正常运行,Terminating表示节点删除中,Terminated表示节点已删除,Draining表示节点正在驱逐 示例值： Pending
CreatedTime	否	否	Datetime	创建时间 示例值：

MachineCore

平台machine资源核数

被如下接口引用：CollectAllCore

名称	必选	允许NULL	类型	描述
ProductId	是	否	String	请求id 示例值：
ProductIdDescribe	否	是	String	产品描述 示例值：
UsageUnit	是	否	String	使用单元 示例值：
UsageValue	是	否	String	使用量 示例值：

AutoScalingGroupRange

集群关联的伸缩组最大实例数最小值实例数

被如下接口引用：ModifyClusterAsGroupAttribute

名称	必选	允许NULL	类型	描述
MinSize	否	否	Int64	伸缩组最小实例数 示例值：
MaxSize	否	否	Int64	伸缩组最大实例数 示例值：

TaskStepInfo

任务步骤信息

被如下接口引用：DescribeUpgradeClusterProgress

名称	必选	允许NULL	类型	描述
Step	是	是	String	升级步骤名称。preUpgrade表示升级前，upgradeMaster表示升级master操作，postUpgrade表示升级后，upgradeNode表示升级node 示例值： preUpgrade
LifeState	是	否	String	任务当前的状态。processing表示任务正在处理中，waiting表示正等待任务执行，done表示任务已经处理完成，unKnow表示任务异常状态 示例值： processing
StartAt	是	是	String	步骤开始时间 示例值：

名称	必选	允许NULL	类型	描述
EndAt	是	是	String	步骤结束时间 示例值：
FailedMsg	是	是	String	若步骤生命周期为failed,则此字段显示错误信息 示例值：
InstanceID	否	否	String	实例ID(仅节点升级使用) 示例值：

ClusterExtraArgs

集群master自定义参数

被如下接口引用：CreateCluster、UpdateClusterVersion

名称	必选	允许NULL	类型	描述
KubeAPIServer	否	是	Array of String	kube-apiserver自定义参数 示例值：
KubeControllerManager	否	是	Array of String	kube-controller-manager自定义参数 示例值：
KubeScheduler	否	是	Array of String	kube-scheduler自定义参数 示例值：

ShieldSettings

告警屏蔽settings，目前仅支持按时间段屏蔽

被如下接口引用：AddAlarmPolicy、DescribeAlarmPolicies、ModifyAlarmPolicy

名称	必选	允许NULL	类型	描述
EnableShield	是	否	Bool	是否启动告警屏蔽功能。true表示启动告警屏蔽,false表示不启用告警屏蔽 示例值： true
ShieldTimeStart	否	是	Int64	告警屏蔽开始时间，单位s，如8:00:00=> 8 *60* 60=28800 示例值：
ShieldTimeEnd	否	是	Int64	告警屏蔽结束时间，单位s，如10:00:00=> 10 *60* 60=36000 示例值：

RegionInstance

地域属性信息

被如下接口引用：DescribeRegions

名称	必选	允许NULL	类型	描述
RegionName	是	是	String	地域名称 示例值：
RegionId	是	是	Int64	地域ID 示例值：
Status	是	是	String	地域状态。AVAILABLE表示地域可用,UNAVAILABLE表示地域不可用 示例值： AVAILABLE
FeatureGates	是	是	String	地域特性开关(按照JSON的形式返回所有属性) 示例值：
Alias	是	是	String	地域简称 示例值：
Remark	是	是	String	地域白名单 示例值：

InstanceDataDiskMountSetting

CVM实例数据盘挂载配置

被如下接口引用：CreateCluster

名称	必选	允许NULL	类型	描述
InstanceType	是	否	String	CVM实例类型 示例值：
DataDisks	是	否	Array of DataDisk	数据盘挂载信息 示例值：查看
Zone	是	否	String	CVM实例所属可用区 示例值：

DescribeClusterInspectionItem

描述集群巡检概览信息

被如下接口引用：DescribeClusterInspections

名称	必选	允许NULL	类型	描述
ClusterId	是	否	String	集群实例id 示例值：
Error	是	是	String	如果发生错误，则该字段包含错误信息 示例值：
Progress	是	是	ClusterInspectionProgress	如果当前正在巡检，则该字段包含检测巡检 示例值：查看

名称	必选	允许NULL	类型	描述
LastResult	是	是	ClusterInspectionOverview	最近一次巡检结果概览 示例值：查看
Cron	是	是	String	自动巡检周期，crontab格式 示例值：

UpgradeAbleInstancesItem

可升级节点信息

被如下接口引用：CheckInstancesUpgradeAble

名称	必选	允许NULL	类型	描述
InstanceId	是	否	String	节点Id 示例值：
Version	是	否	String	节点的当前版本 示例值：

LoginSettings

描述了实例登录相关配置与信息。

被如下接口引用：AddExistedInstances、UpgradeClusterInstances

名称	必选	允许NULL	类型	描述
Password	否	否	String	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16位，至少包括两项[a-z，A-Z]、[0-9] 和 [( ) ` ~ ! @ # $ % ^ & * - + =]
KeyIds	否	否	Array of String	密钥ID列表。关联密钥后，就可以通过对应的私钥来访问实例；KeyId可通过接口DescribeKeyPairs获取，密钥与密码不能同时指定，同时Windows操作系统不支持指定密钥。当前仅支持购买的时候指定一个密钥。 示例值：
KeepImageLogin	否	否	String	保持镜像的原始设置。该参数与Password或KeyIds.N不能同时指定。只有使用自定义镜像、共享镜像或外部导入镜像创建实例时才能指定该参数为TRUE。取值范围： • TRUE：表示保持镜像的登录设置 • FALSE：表示不保持镜像的登录设置 默认取值：FALSE。 示例值：

NodePoolSet

虚拟节点池信息

被如下接口引用：DescribeClusterVirtualNodePools

名称	必选	允许NULL	类型	描述
NodePoolId	否	否	String	节点池Id 示例值：
SubnetIds	否	否	Array of String	节点池子网Id 示例值：
SecurityGroupIds	否	否	Array of String	节点池安全组id 示例值：
Name	否	否	String	节点池名称 示例值：
LifeState	否	否	String	节点池状态。creating表示节点池创建中,appending表示节点池正追加新节点,normal表示节点池正常状态,updating表示节点池更新中，deleting表示节点池删除中,shrinking表示节点池缩容 示例值： creating
Unschedulable	否	否	Bool	节点池中的超级节点是否不可调度。true表示不可调度，false表示可调度 示例值： true
Labels	否	否	Array of Label	节点池标签 示例值：查看
Taints	否	否	Array of Taint	节点池污点 示例值：查看

ClusterAsGroupOption

集群弹性伸缩配置

被如下接口引用：DescribeClusterAsGroupOption、ModifyClusterAsGroupOptionAttribute

名称	必选	允许NULL	类型	描述
IsScaleDownEnabled	是	否	Bool	是否开启缩容 示例值： false
Expander	是	否	String	多伸缩组情况下扩容选择算法(random 随机选择，most-pods 最多类型的Pod least-waste 最少的资源浪费，默认为random) 示例值： random
MaxEmptyBulkDelete	否	是	Int64	最大并发缩容数 示例值：
ScaleDownDelay	否	是	Int64	集群扩容后多少分钟开始判断缩容（默认为10分钟） 示例值：
ScaleDownUnneededTime	否	是	Int64	节点连续空闲多少分钟后被缩容（默认为 10分钟） 示例值：

名称	必选	允许NULL	类型	描述
ScaleDownUtilizationThreshold	否	是	Int64	节点资源使用量低于多少(百分比)时认为空闲(默认: 50(百分比)) 示例值：
SkipNodesWithLocalStorage	否	是	Bool	含有本地存储Pod的节点是否不缩容(默认： FALSE) 示例值： true
SkipNodesWithSystemPods	否	是	Bool	含有kube-system namespace下非DaemonSet管理的Pod的节点是否不缩容 (默认： FALSE) 示例值： true
IgnoreDaemonSetsUtilization	否	是	Bool	计算资源使用量时是否默认忽略DaemonSet的实例(默认值: False，不忽略) 示例值： true
MaxTotalUnreadyPercentage	否	是	Int64	unready节点最大占比 示例值：
OkTotalUnreadyCount	否	是	Int64	unready节点累计数量 示例值：
ScaleDownUnreadyTime	否	是	Int64	unready节点缩容时间 示例值：
UnregisteredNodeRemovalTime	否	是	Int64	未注册成功节点移除耗时 示例值：

ClusterInspectionOverview

集群巡检一次检测的概览

被如下接口引用：DescribeClusterInspectionOverviews、DescribeClusterInspections

名称	必选	允许NULL	类型	描述
Id	是	否	String	本次检查的id，用于与其他检测结果区分 示例值：
StartTime	是	否	String	检查的时间 示例值：
Error	是	是	String	如果检查异常终止，则该字段包含错误信息 示例值：
Statistic	是	是	InspectionStatistic	统计信息 示例值：查看

错误码

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 cor
rect."
    },
    "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
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接口鉴权文档检查签名计算过程。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未 CAM 授权。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。

错误码	说明
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
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.Param	Param。
UnsupportedOperation	操作不支持
InternalError.TaskNotFound	任务未找到。
InvalidParameter.ClusterNotFound	集群ID不存在。

错误码	说明
InternalError.AsCommon	伸缩组资源创建报错。
InvalidParameter.CIDRInvalid	非法的CIDR。
LimitExceeded	超过配额限制
UnauthorizedOperation	未授权操作
InternalError.CidrConflictWithVpcGlobalRoute	CIDR和全局路由冲突。
InternalError.PublicClusterOpNotSupport	集群不支持当前操作。
InternalError.CamNoAuth	没有权限。
InternalError.QuotaUSGLimit	安全组配额不足。
InternalError.Db	db错误。
InvalidParameter.AsCommonError	弹性伸缩组创建参数错误。
InvalidParameter.Param	参数错误。
InternalError.VstationError	VstationError。
InternalError.ImageIdNotFound	镜像未找到。
FailedOperation	操作失败
InternalError.InitMasterFailed	初始化master失败。
InternalError.DbRecordNotFound	记录未找到。
InvalidParameter	参数错误
InternalError.QuotaMaxNodLimit	超过配额限制。
InvalidParameter.RouteTableNotEmpty	路由表非空。
InternalError.ClusterNotFound	集群未找到。
InvalidParameter.CidrOutOfRouteTable	CIDR不在路由表之内。
InvalidParameter.CIDRMaskSizeOutOfRange	CIDR掩码超出范围(集群CIDR范围 最小值: 10 最大值: 24)。
InternalError.CidrInvali	CIDR无效。
InternalError.CvmNotFound	cvm不存在。

错误码	说明
InternalError.AccountUserNotAuthenticated	账户未通过认证。
InternalError.CreateMasterFailed	创建集群失败。
MissingParameter	缺少参数错误
InternalError.CidrOutOfRouteTable	CIDR不在路由表之内。
InvalidParameter.InvalidPrivateNetworkCIDR	无效的私有CIDR网段。
InternalError.CidrConflictWithOtherRoute	CIDR和其他路由冲突。
InternalError.OsNotSupport	镜像OS不支持。
ResourceNotFound	资源不存在
InternalError.CvmCommon	cvm创建节点报错。
InternalError.CidrConflictWithOtherCluster	CIDR和其他集群CIDR冲突。
InternalError.KubeJarvisAlreadyRunning	集群正在检查。
InvalidParameter.GatewayAlreadyAssociatedCidr	下一跳地址已关联CIDR。
InternalError.QuotaMaxClsLimit	超过配额限制。
InternalError.PodNotFound	Pod未找到。
UnknownParameter	未知参数错误
InternalError.CidrMaskSizeOutOfRange	CIDR掩码无效。
InternalError.DfwGetUSGQuota	获得安全组配额失败。
ResourceUnavailable	资源不可用
InternalError.DbAffectivedRows	DB错误
InternalError.VpcCommon	VPC报错。
InternalError.QuotaMaxRtLimit	超过配额限制。
InternalError.InvalidPrivateNetworkCidr	无效CIDR。
InternalError.TaskLifeStateError	任务当前所处状态不支持此操作。
InternalError.CidrConflictWithVpcCidr	CIDR和vpc冲突。
InternalError.VpcPeerNotFound	对等连接不存在。

错误码	说明
InternalError.UnexceptedInternal	内部错误
InternalError	内部错误
ResourceInUse	资源被占用
InternalError.DfwGetUSGCount	获得当前安全组总数失败。
InternalError.VpcRecodrNotFound	未发现vpc记录。