腾讯专有云 TCE

Tencent Cloud Enterprise

容器服务

产品白皮书

腾讯云TCE

I 产品简介

II 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 TCE 的 CBS、CLB 等 Kubernetes 插件，同时以 TCE 私有网络为基础，实现了高可靠、高性能的网络方案。

容器服务平台 TKE 与 TCE 完全集成，在 TCE 的控制台上可以直观地创建容器集群，做集群管理、工作负载管理、网络管理、存储管理、权限管理、镜像仓库、日志监控等功能模块和服务。同时支持无状态、有状态、任务、定时任务、Kubernetes 的工作负载对象部署，并支持应用模板和配置项的管理；

名词解释

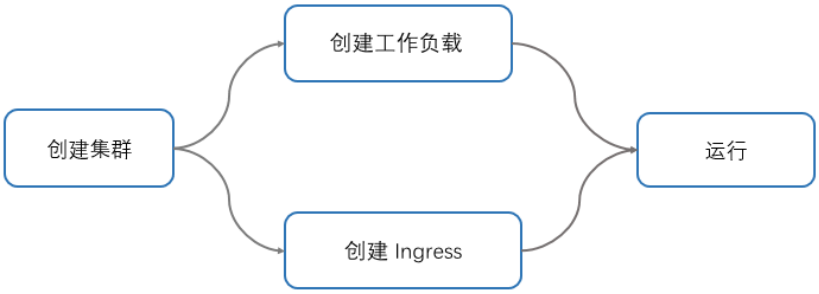
使用容器服务平台，会涉及到以下基本概念：

- **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。

II 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
安全可靠	**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。	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
	• 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度安全可靠的应用程序。 • 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。	
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

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

II 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：

（Helm Chart、事件持久化将在 3.4 中上线）

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，帮助用户快速在 TCE 上构建 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**：TCE 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。

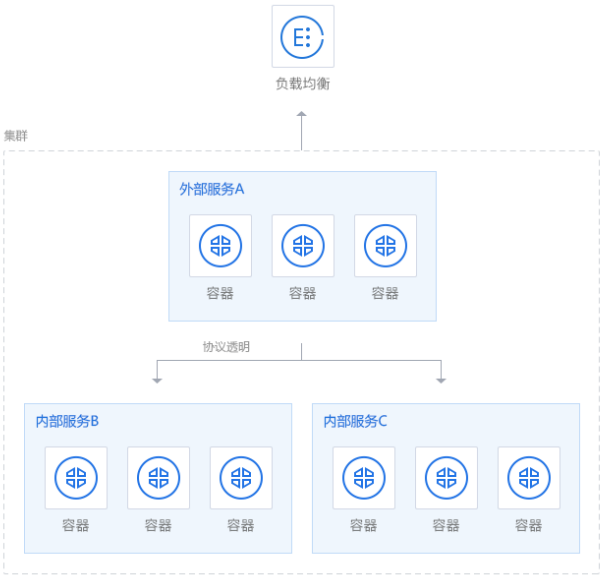
II 应用场景

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 TCE 的计算、网络、存储、监控、安全能力，直接使用 TCE 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 TCE 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。

II 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，整个过程安全可靠，并且能够无缝衔接 TCE 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 • 支持包年包月、按量计费两种计费模式 • 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
Kubernetes 管理	• 支持 Kubernetes 多版本，提供版本升级功能 • Kubernetes 证书管理，Kubectl 直接操作集群 • 控制台简单管理 Namespace

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 Kuberntes Deployment、DamentSet 等多种资源

模块	功能点
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、模板市场 • 模板支持一键复制

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 • 支持 AZ、节点、应用三层亲和、反亲和能力；
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

模块	功能点
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

TCE 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

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 TencentHub 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

I 快速入门

II 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。

II 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

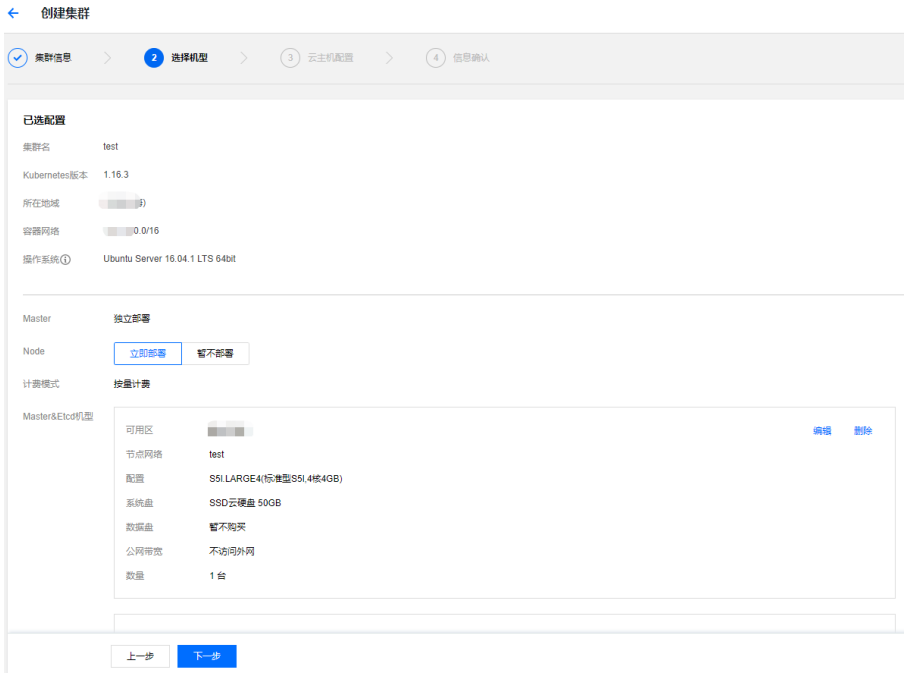
首先您需要创建集群。集群是指容器运行所需云资源的集合，包含了若干台云服务器、负载均衡器等资源。

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。
    - **集群名称**：您要创建的集群的名称。不超过 60 个字符。
    - **Kubernetes 版本**：选择 Kubernetes 版本。
    - **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。
    - **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“腾讯云标签”：为 TKE 集群配置腾讯云标签，集群内创建的云服务的资源自动继承集群标签。

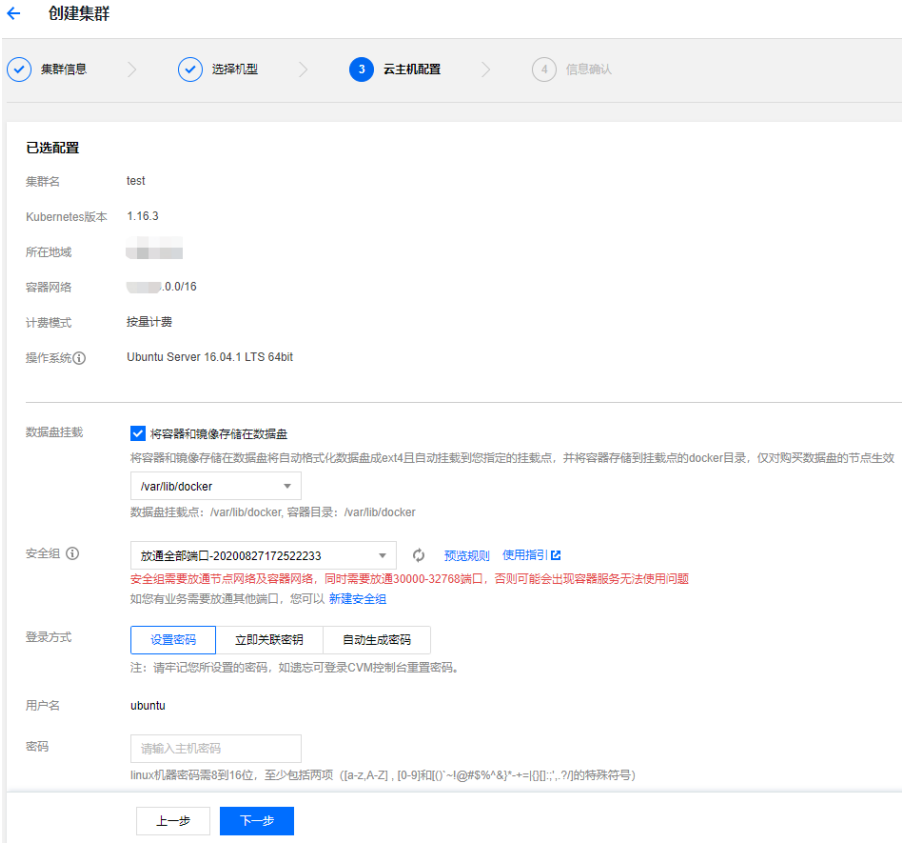
4. 选择机型（支持系统盘为云盘的所有机型）。
    - **Node**:
        - **立即部署**: 直接在新建集群过程中新增 Node 节点
        - **暂不部署**: 再集群新建完成之后再新增 Node 节点
    - **Master&Etcd 机型**:
        - **可用区**：选择当前服务可用的区域。
        - **节点网络**：为集群内服务器分配在节点网络地址范围内的 IP 地址。参阅 容器及节点网络设置。
        - **机型**：选择节点机型
        - **系统盘**：SSD 云硬盘固定为 50GB，系统盘不支持更换介质，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. 填写云主机配置并单击 【完成】。
   - **数据卷挂载**：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

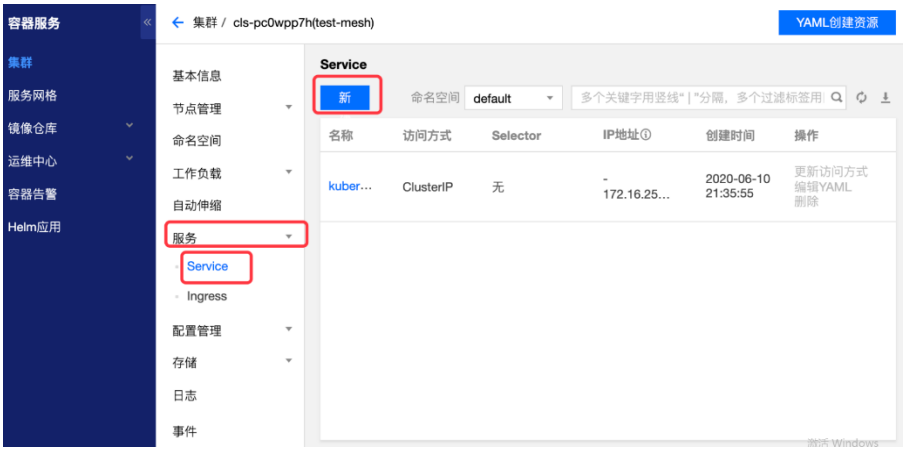


6. 单击【下一步】并确认配置信息。
7. 单击【完成】。 创建完成的集群将出现在集群列表中。

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. 单击上图中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。
        - **服务描述**：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**：
        - **服务访问方式**：
            - **提供公网访问**：自动创建传统型公网 CLB（0.02 元/小时）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. 部署设置。

   - **设置数据卷（选填）**
     要指定容器挂载至指定路径时，单击【添加数据卷】。详情查看 使用指引。

     注意： 源路径不指定时将默认分配临时路径。

     - 类型：支持使用本地硬盘、云硬盘、NFS 盘、配置项四种类型的数据卷。相关详细介绍请参阅 容器服务数据卷使用说明。
     - 名称：数据卷的名称。
     - 路径：指定容器要挂载的路径。

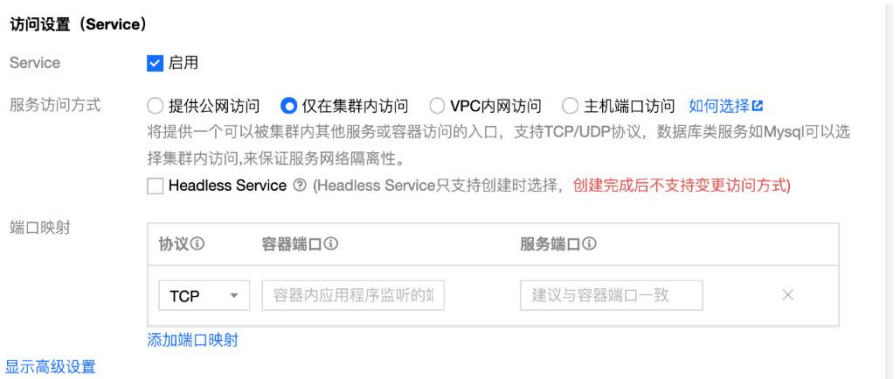
   - **设置运行容器**
     - 名称：要创建容器的名称。最长 63 个字符，只能包含小写字母、数字及分隔符(-)，且不能以分隔符开头或结尾。

- 镜像：单击【选择镜像】，可选择在我的镜像、我的收藏、TencentHub 镜像、DockerHub 镜像和其他镜像下创建服务。
  - 镜像版本（Tag）：容器服务默认选择版本。如果您需要使用镜像的其它版本，单击版本显示框选择。
  - CPU/内存限制：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：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。

- **实例数量**
  - 手动调节：直接设定实例数量。一个实例由相关的一个或多个容器构成。可单击 + 或 - 控制实例数量。
  - 自动调节：满足任一设定条件，则自动调节实例（pod）数目。

4. 访问设置。
   - **服务访问方式**：服务的访问方式决定了这个服务的网络属性，不同访问方式的服务可以提供不同网络能力。提供的四种访问方式详细介绍请参阅 服务访问方式设置。

- **端口映射**：选择 **协议**，填写 **容器端口** 和 **服务端口**。



5. 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

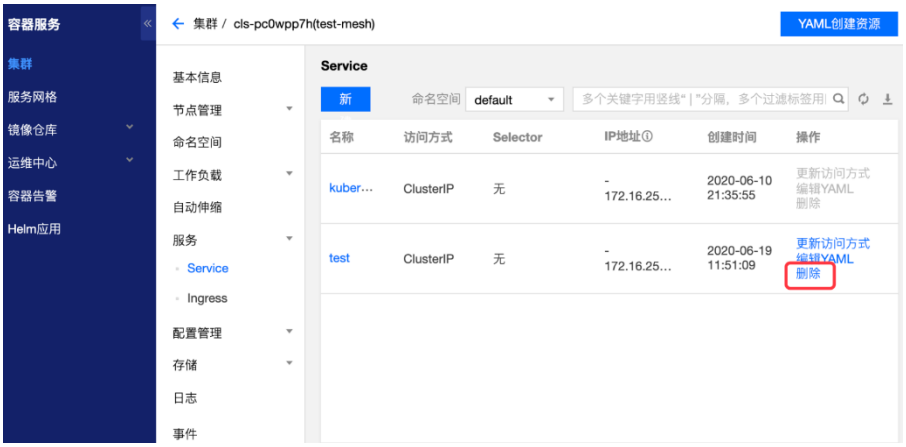
1. 单击左侧导航栏中的【集群】，在“集群管理”页单击集群列表右侧的【删除】。

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【确定】。

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。

II 入门示例

III 创建简单的 nginx 服务

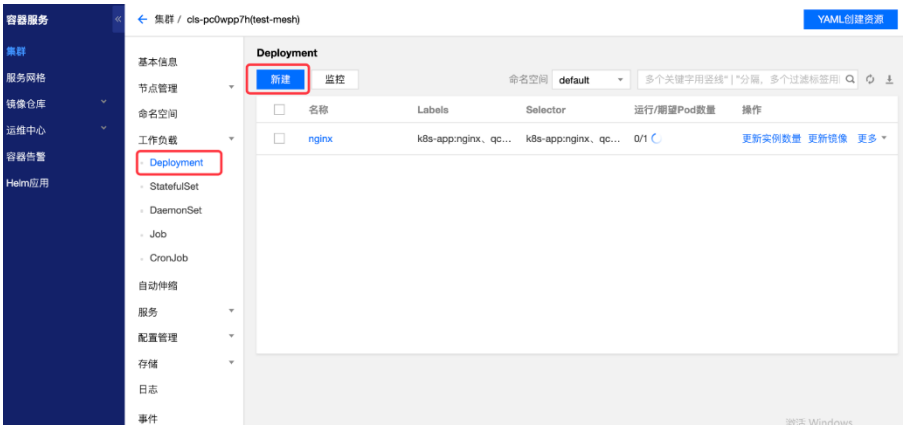
本文档旨在帮助大家了解如何快速创建一个容器集群内的 Nginx 服务。

> 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】。

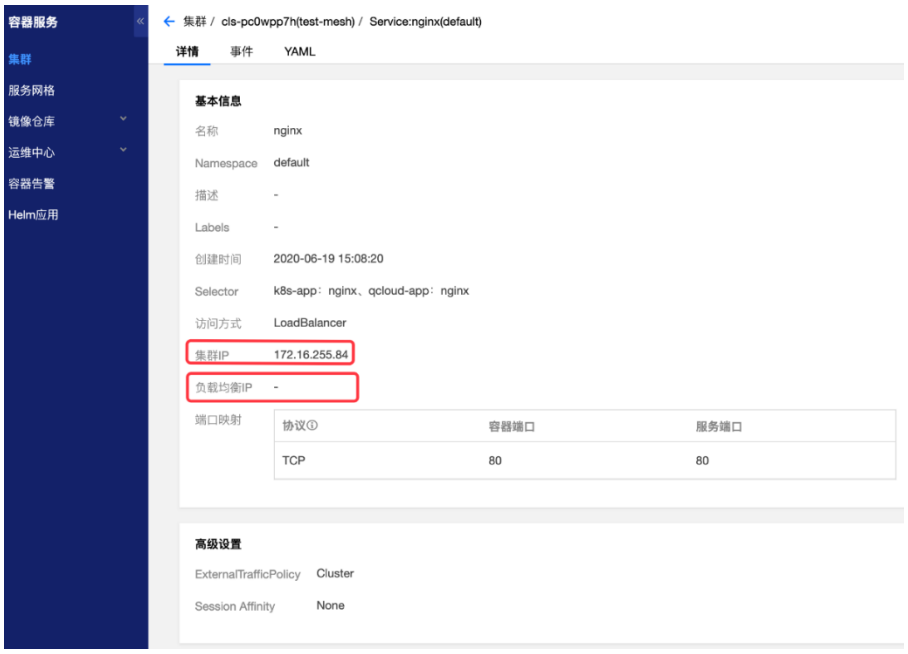


4. 在【新建 Workload】页面里输入以下参数。
    - **工作负载名**：输入自定义名称，这里以 `nginx` 为例。
    - **实例内容器**：
        - **名称**：自定义，这里以 `my-container` 为例。
        - **镜像**：根据实际需求进行选择，这里以 `nginx` 为例。
    - **访问设置（Service）**：勾选【启用】Service 按钮，会新建一个与负载同名的 Service。
    - **服务访问方式**：为方便测试，这里选择 提供公网访问。
    - **端口映射**：容器端口和服务端口都填 80。
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。

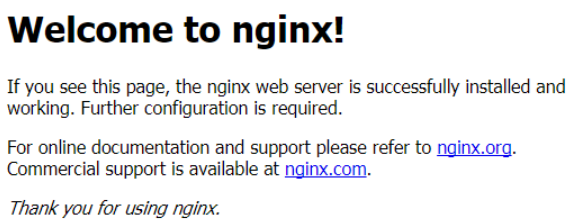


ii. 可通过浏览器输入上图中的**负载均衡 IP:服务端口**来访问 Nginx 服务。

- 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。



III 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。

```
2. [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);
```

6. 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
```

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

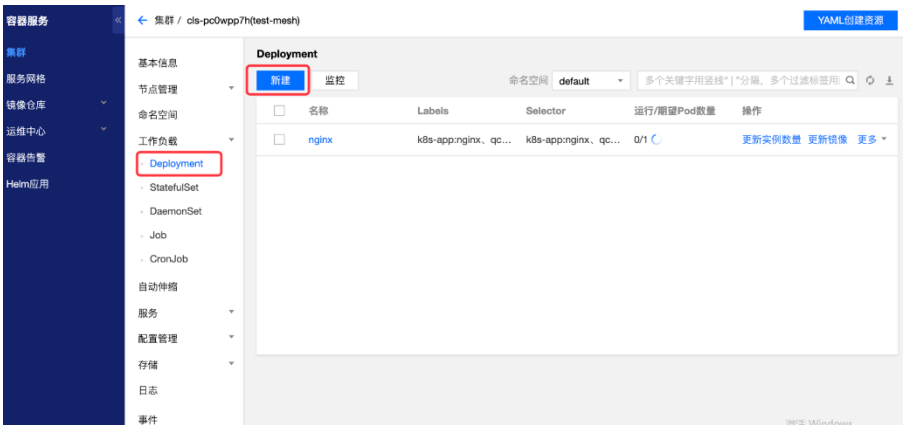
第二步：通过该镜像创建 Hello World 服务

> 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。

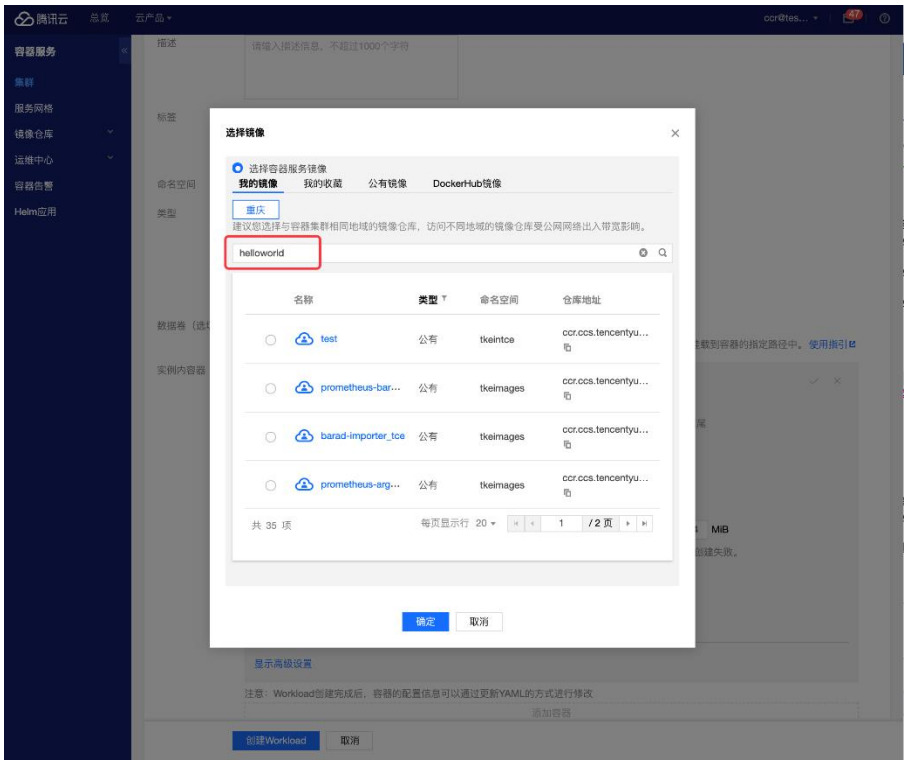
创建 helloworld 服务

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



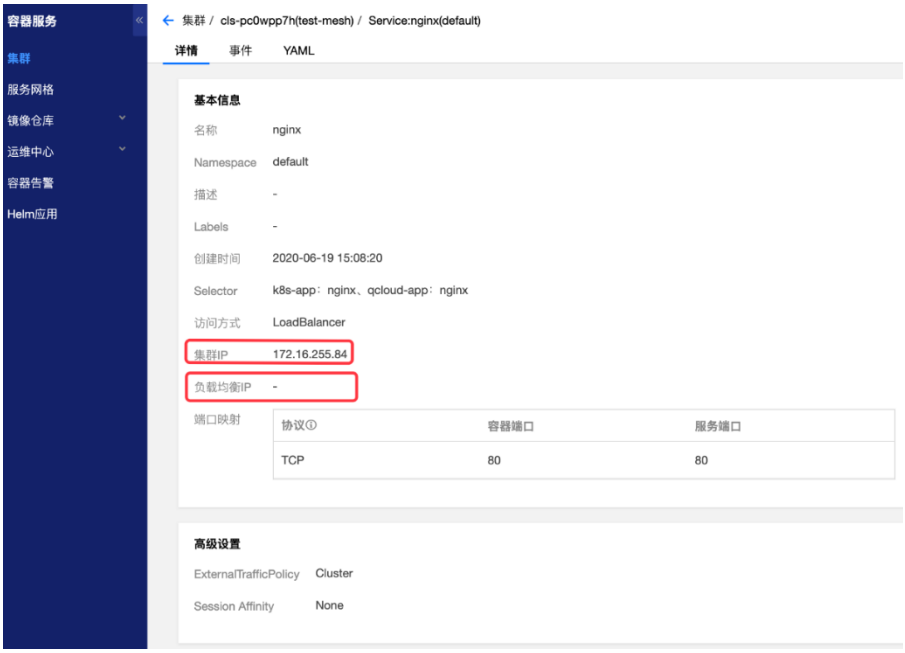
    - **访问设置（Service）**：勾选【启用】Service 按钮，会新建一个与负载同名的 Service
        - **服务访问方式**：为方便测试，这里选择 提供公网访问
        - **端口映射**：容器端口和服务端口都填 80
5. 单击**创建 Workload**。

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。

III 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。

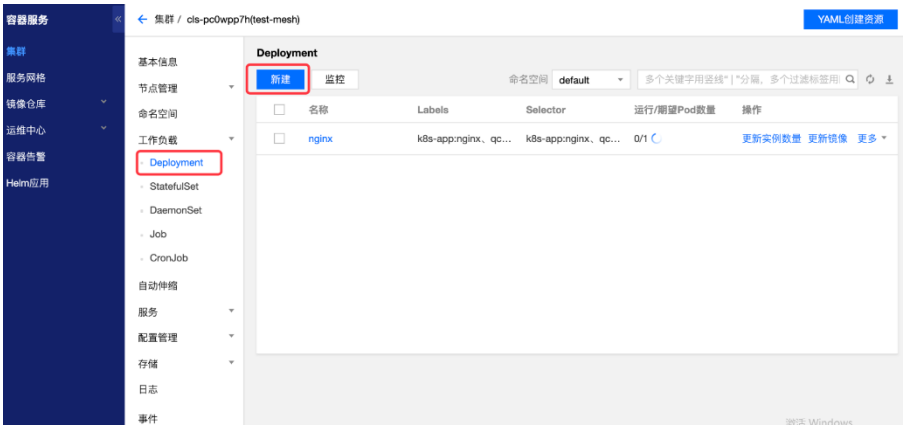
本文档旨在介绍如何使用 `tutum/wordpress` 镜像来创建一个公开访问的 WordPress 网站。

> 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 TCE 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 TCE 帐户。有关如何创建 TCE 帐户，请在 [注册页面] 填写相关信息注册 TCE 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 。

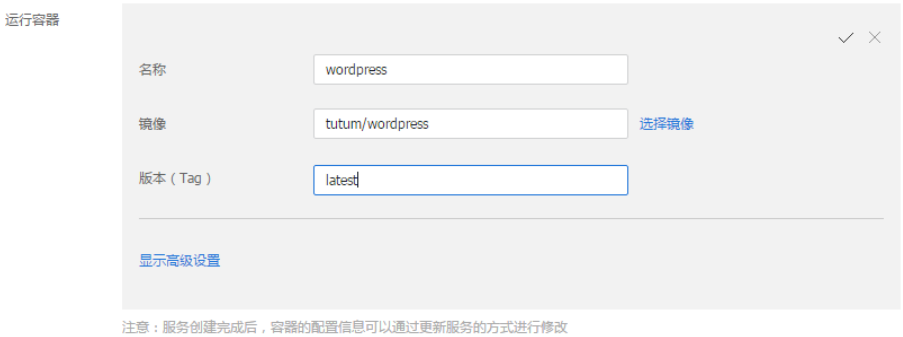
创建 WordPress 服务

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】。



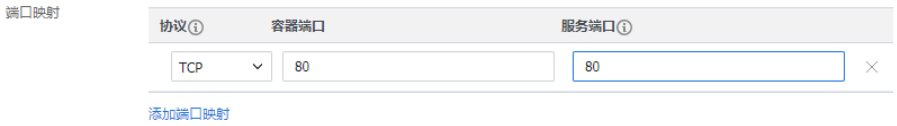
4. 在“新建 Workload”页面，根据实际情况，设置工作负载基本信息。
5. 镜像配置。
   - **名称**：输入运行容器的名称，此处以 wordpress 为例。
   - **镜像**：填写 `tutum/wordpress` 。
   - **版本（Tag）**：填写 latest。



注意：服务创建完成后，容器的配置信息可以通过更新服务的方式进行修改

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。

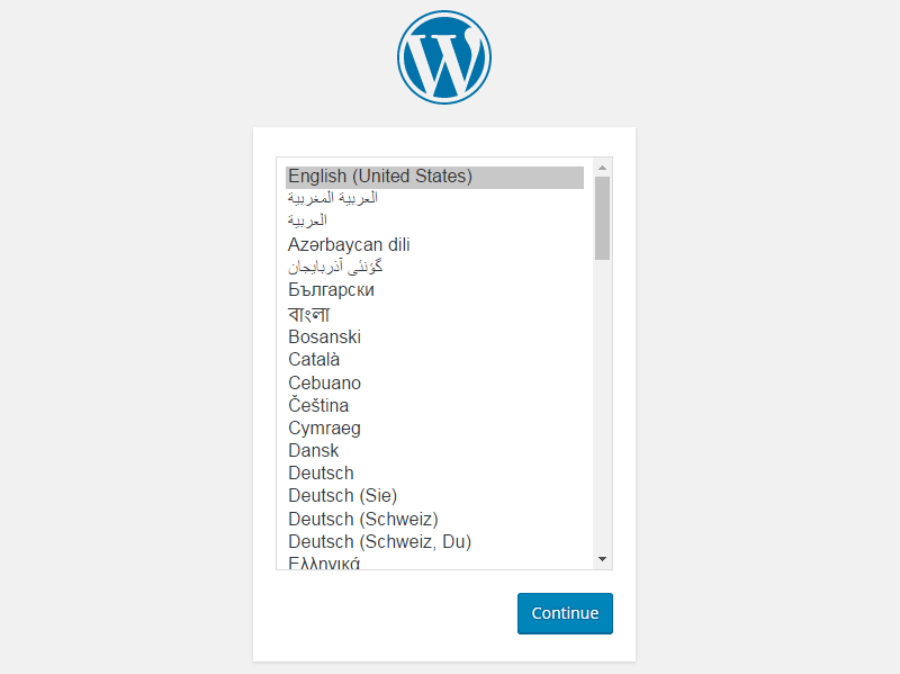
访问 WordPress 服务

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。 如下图所示:

- 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。

III 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。

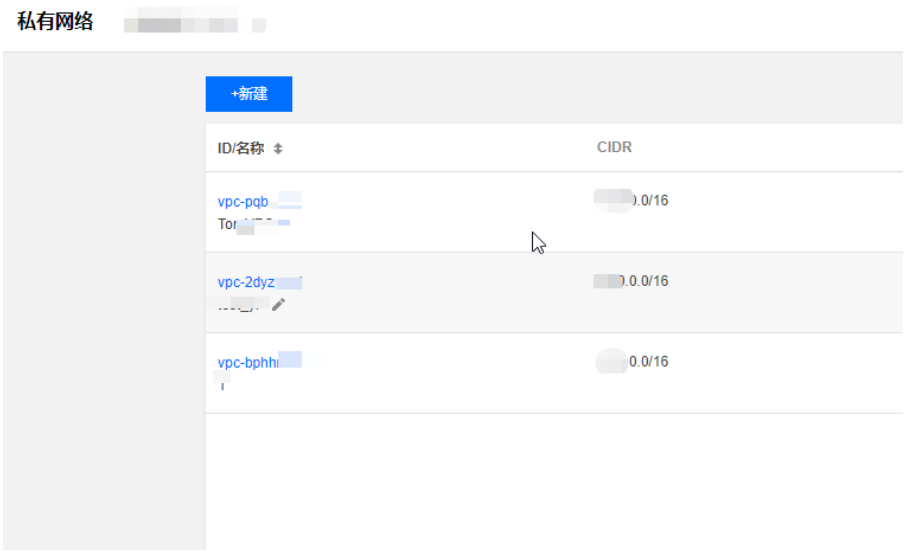
> 注意：在创建使用 CDB 的 WordPress 服务之前，您必须拥有：
>
> - 一个 TCE 账户。有关如何创建 TCE 账户，请在 注册页面 填写相关信息注册 TCE 账户。
> - 一个创建好的集群。有关如何创建集群的详细信息，参见 新建集群 。

创建 WordPress 服务

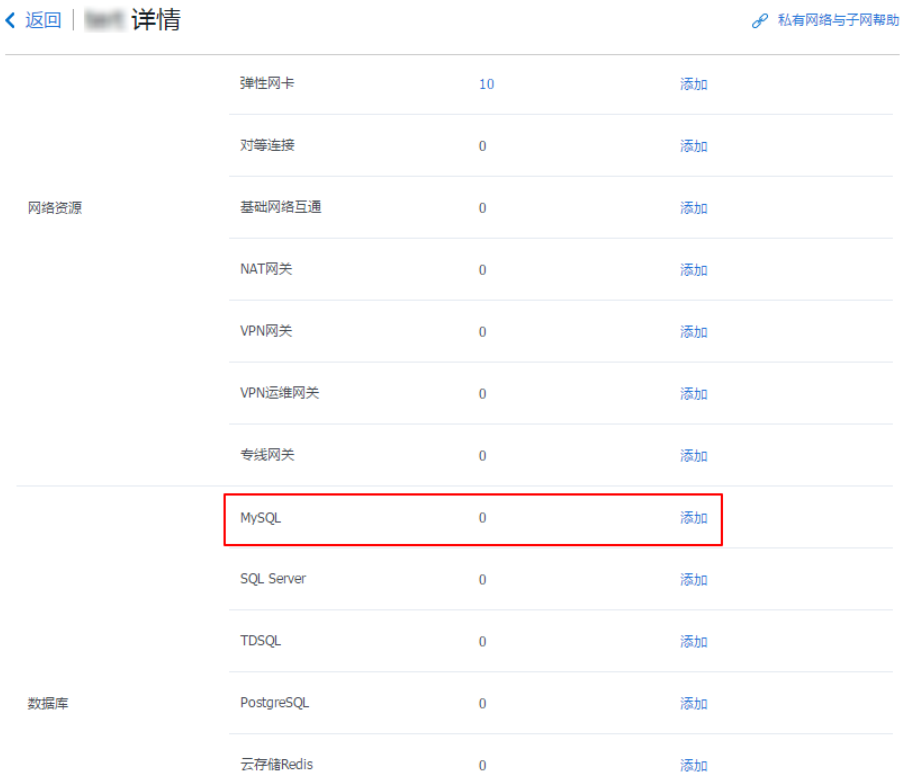
第一步：创建云数据库 CDB

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】。



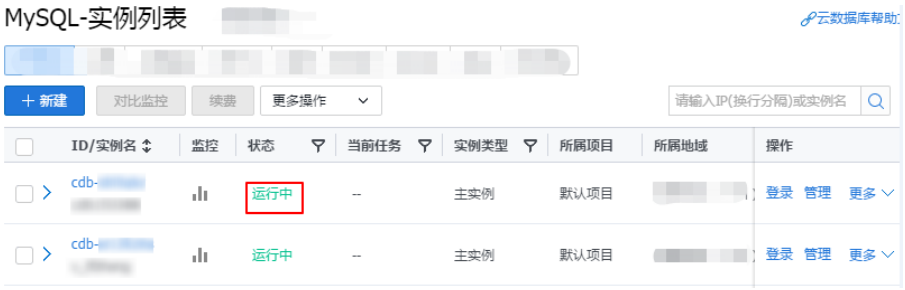
4. 选择购买配置，完成系列支付操作。相关详情请参见 数据库 MySQL。
5. 购买的 MySQL 将出现在 MySQL 实例列表中。

6. 初始化 MySQL 实例。单击右侧 **操作** 栏下的【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。

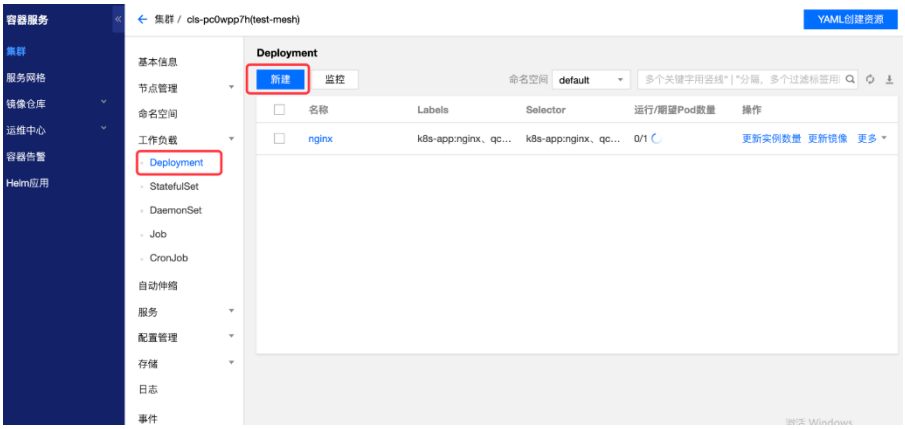
8. 目标 MySQL 实例的状态变为 **运行中**，说明已初始化成功。



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】。

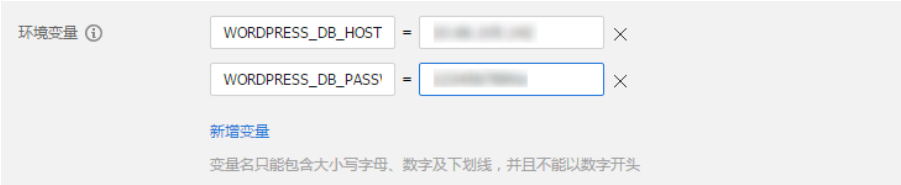
3. 在【集群】页面，选择【工作负载】 > 【Deployment】，在 Deployment 列表里面单击【新建】。



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` 。

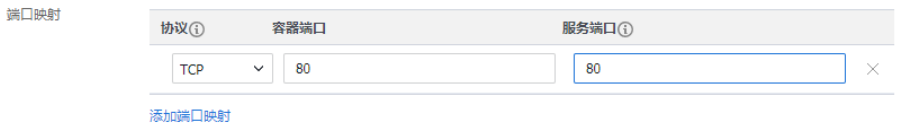
- **版本（Tag）**：填写 latest。

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。

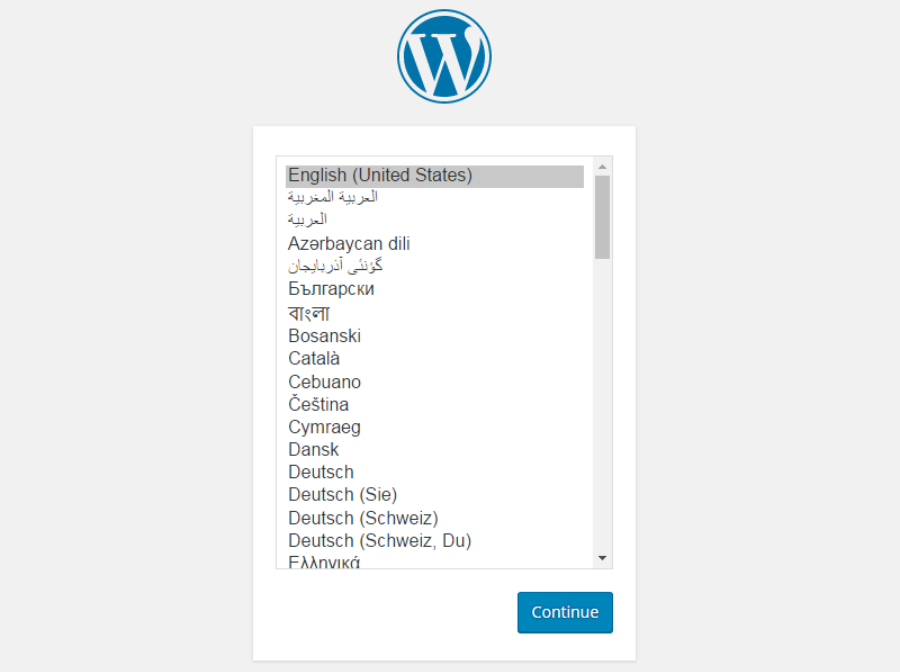
访问 WordPress 服务

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。

I 用户指南

II 集群管理

III 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 TCE 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:

- CVM 容器集群
- 独立部署集群（Master、Etcd 采用用户自有主机搭建）

独立部署模式集群

模式简介

TKE 也为客户提供集群完全自主可控的 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。

> 注意:
>
> - 该模式仅适用于 Kubernetes 1.10.x 以上版本。
> - 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
约 100 个节点
约 500 个节点
Master 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

为了保证您集群的稳定性，以及发生异常后的恢复效率，在此我们有以下建议：

- 独立部署模式，请不要删除 Master 节点下支撑 Kuberntes 运行的核心组件。
- 独立部署模式，请不要修改 Master 核心组件的配置参数。
- 独立部署模式，请不要修改/删除集群内部的核心资源。
- 独立部署模式，请不要修改/删除 Master 节点的相关证书文件（拓展名是.crt, .key)。
- 非必要情况下，请不要修改任何节点的 docker 版本。
- 非必要情况下，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
- 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，

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

III 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
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

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，CBS。

III 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

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

20. 登录 TKE 控制台。
21. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
22. 在“集群管理”页面中，单击【新建】。如下图所示：

23. 在“集群信息”页面，设置集群的基本信息。如下图所示：
    Documentation。 - **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
    作系统**：根据实际需求进行选择。 - **集群描述**：填写集群的相关信息，该信息将显示在**集群信息
    自动继承集群标签。
24. 单击【下一步】。

选择机型

25. 在“选择机型”中，选择部署模式和机型。如下图所示：
    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
    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
    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。 - **
    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
    ：机型选择方案参看 实例类型概述 和 确定云服务器配置方案。 - **系统盘：默认为“本地硬盘 50G
    IP**，系统将免费分配公网 IP。提供两种计费模式，详细对比请参看 公网计费模式。 - **数量**：设置
26. 单击【下一步】，配置云服务器。

配置云服务器

27. 在“云服务器配置”中，配置云服务器的其他配置。如下图所示：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**：密钥对是的伸缩组。
28. 单击【下一步】，检查并确认配置信息。
29. 单击【完成】，即可完成创建。

III 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

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

移除节点

具体操作请参考 移除节点。

III 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/d
  ```

- **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/l
  ```

- **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/wi
  ```

操作步骤

安装 Kubectl 工具

35. 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。

36. 执行以下命令，添加执行权限。

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

38. 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", GitComm
```

获取集群账号密码以及证书信息

39. 登录 TKE 控制台。
40. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
41. 单击需要连接集群的【ID/名称】，进入该集群的管理页面。
42. 在左侧导航栏中，单击【基本信息】，进入“基本信息”页面。如下图所示：

43. 在“基本信息”中，单击【集群凭证】中的 【显示凭证】。
44. 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。

45. 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
    - 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
    - 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

      ```
      sudo sed -i '$a **IP 地址** **域名**' /etc/hosts
      ```
46. 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。

47. 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。

48. 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.tencent-cloud.com" --username=admin --pass
```

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。

49. 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=6666o9oI
kubectl config set-cluster default-cluster --server=https://cls-66668888.ccs.tenc
kubectl config set-context default-system --cluster=default-cluster --user=defaul
kubectl config use-context default-system
```

53. 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)
```

III 升级集群

操作场景

TCE 容器服务暂不提供界面升级 Kubernetes 版本的功能，如有需要，请与 TCE 支持人员联系。

III 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

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

58. 登录 TKE 控制台。

59. 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
60. 按照页面提示逐步操作，完成集群的创建。

III 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 TCE 容器服务支持 GPU 功能，通过该功能可

- 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

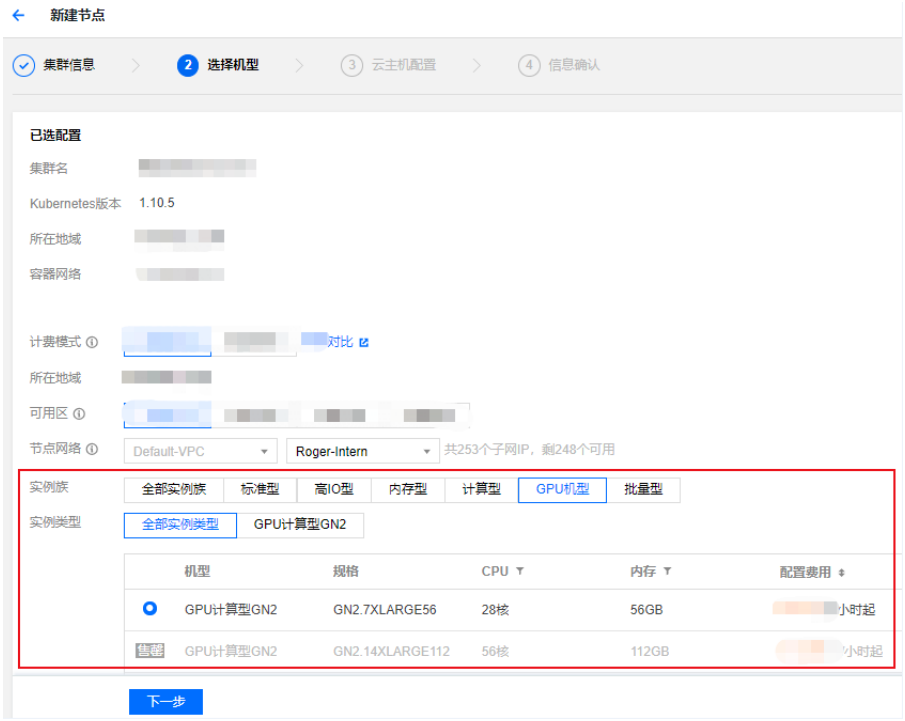
在集群中添加 GPU 节点

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

72. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
73. 在需要创建 GPU 云服务器的集群行中，单击【新建节点】。
74. 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。如下图所示：

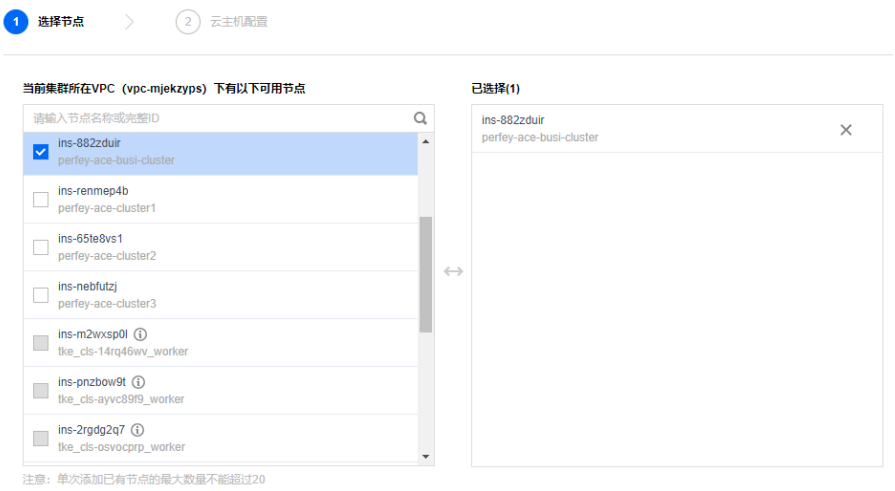


75. 按照页面提示逐步操作，完成创建。

    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

添加已有 GPU 云服务器

76. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
77. 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】。
78. 在“选择节点”页面，勾选已有的 GPU 节点，单击【下一步】。如下图所示：



79. 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

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

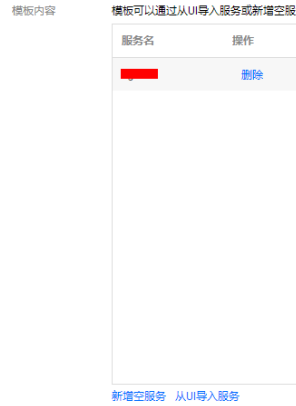
通过控制台方式创建

82. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
83. 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。

84. 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】 > 【Da
85. 单击【新建】，进入“新建 Workload”页面。

86. 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。如下
87. 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。如下图所示：

III 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
- **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
  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 TCE 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。
- 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

- 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

III 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 TCE 上的资源通信链路

- 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**
- 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

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

107. 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

108. 在左侧导航栏中，选择【基本信息】，进入“基本信息”页面。如下图所示：
109. 记录 “基本信息” 中 “容器网络” 的网段和掩码。
110. 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

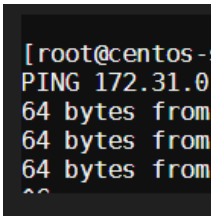
111. 切换至私有网络控制台。
112. 在左侧导航栏中，单击【子网】，进入“子网”页面。

113. 单击对等连接本端指定子网的关联路由表。如下图所示：
114. 在关联路由表的【默认详情】页面，单击【+新增路由策略】。
115. 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。
116. 单击【确定】，完成本端路由表的配置。
117. 重复执行 步骤 2 - 步骤 6，完成对端路由表的配置。

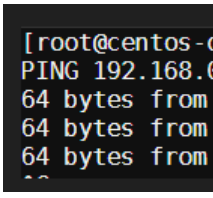
预期结果

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。

118. 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：



119. 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：



III 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 TCE 上的资源通信链路

- 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**

操作步骤

获取容器的基本信息

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

124. 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。如下图所示：

125. 在左侧导航栏中，选择【基本信息】，进入“基本信息”页面。如下图所示：
126. 记录 “基本信息” 中 “所在地域”、“节点网络” 和 “容器网络” 的信息。
127. 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

配置路由表

128. 切换至私有网络控制台。

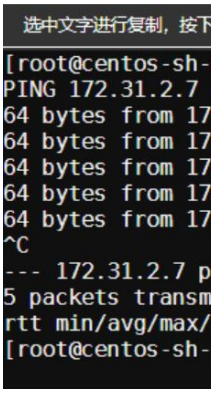
129. 在左侧导航栏中，单击【对等连接】，进入对等连接管理页面，并记录对等连接的【ID/名称】。
130. 在左侧导航栏中，单击【子网】，进入“子网”管理页面。

131. 单击对等连接本端指定子网的关联路由表。如下图所示：
132. 在关联路由表的“默认详情”页面，单击【+新增路由策略】。
133. 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。
134. 单击【确定】，完成本端路由表的配置。
135. 重复执行 步骤 3 - 步骤 7，完成对端路由表的配置。

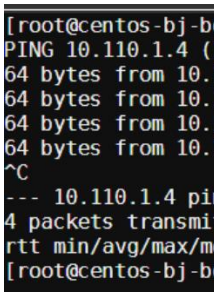
预期结果

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。

136. 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：



137. 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：



III 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

操作步骤

通过专线方式互通

140. 参考 申请物理专线，申请物理专线。
141. 参考 申请通道，申请通道。
142. 参考 创建专线网关，创建专线网关。
143. 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。

144. 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
145. 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。
146. 验证容器与与 IDC 互通。

通过 VPN 方式互通

配置 SPD 策略

147. 登录私有网络控制台。
148. 在左侧导航栏中，单击【VPN 链接】>【VPN 通道】，进入“VPN 通道”管理页面。

149. 单击需要配置 SPD 策略的本端 VPN 通道的 ID/名称。如下图所示：

150. 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。如下图所示：
151. 单击【保存】。
152. 重复执行 步骤 3 - 步骤 5，配置对端 VPN 通道的 SPD 策略。

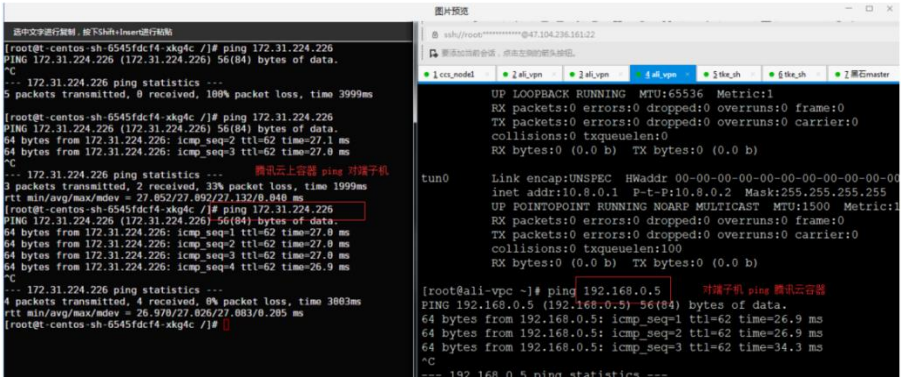
添加容器网段

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。

153. 在左侧导航栏中，单击【路由表】，进入路由表管理页面。
154. 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
155. 单击【+新增路由策略】，追加容器网段。
156. 选择【关联子网】页签，单击【新建关联子网】，关联子机所在的子网。
157. 重复执行 步骤 2 - 步骤 4，在您对端的路由设备上，添加 TCE 容器所在网段。

预期结果

容 器 和 对 端 子



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**。

II 镜像仓库

III 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 Registr

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

III 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 Registr

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。

160. 登录 TKE 控制台。

161. 选择左侧导航栏中的【镜像仓库】>【我的镜像】。

162. 根据以下提示填写相关信息后单击【开通】进行初始化。如下图所示：

- **用户名**：默认是当前用户的账号，是您登录到 TCE Docker 镜像仓库的身份。
- **密码**：是您登录到 TCE Docker 镜像仓库的凭证。

创建命名空间

163. 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。

164. 在“我的镜像”页面中，选择【命名空间】页签并单击【新建】。如下图所示：

165. 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】。如下图所示：

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

创建镜像

166. 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。

167. 在“我的镜像”页面，单击镜像列表页上方的【新建】。如下图所示：

168. 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 TCE registry

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

```
$ sudo docker login --username=[username] ccr.ccs.tencentyun.com
```

username：TCE 账号，开通时已注册。

170. 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.ccs.tencentyun.com/[namespace]/[ImageName]:[镜像版本号]
$ sudo docker push ccr.ccs.tencentyun.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

174. 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.ccs.tencentyun.com
```

175. 替换命令中的相关信息并执行，下载镜像。

```
$ sudo docker pull ccr.ccs.tencentyun.com/[namespace]/[ImageName]:[镜像版本号]
```

删除镜像

176. 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。

177. 选择需删除镜像所在行右侧【删除】。

178. 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**。如下图所示：

III 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

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

搭建 registry

179. 在服务器上执行如下命令安装 Docker，这里选择 TCE（Ubuntu Server 14.04.1 LTS 64 位）镜像

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

180. 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

```
curl -L https://github.com/docker/compose/releases/download/1.8.0/docker-compose-
chmod a+x /usr/local/bin/docker-compose
```

181. 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

182. 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-regist

```
DOCKER_OPTS="--insecure-registry localhost"
```

183. 重启 Docker。

```
service docker restart
```

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

```
185. docker pull hello-world
186. 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 htt
    {
       "schemaVersion": 2,
       "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
       "config": {
          "mediaType": "application/vnd.docker.container.image.v1+json",
          "size": 1473,
          "digest": "sha256:c54a2cc56cbb2f04003c1cd4507e118af7c0d340fe7e2720f70976c4b752
       },
       "layers": [
          {
             "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
             "size": 974,
             "digest": "sha256:c04b14da8d1441880ed3fe6106fb2cc6fa1c9661846ac0266b8a5ec8e
          }
       ]
    }
```

其中 `c54a2cc56cbb2f04003c1cd4507e118af7c0d340fe7e2720f70976c4b75237dc` 即为执行 docker images

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

```
    # curl -s -X GET http://localhost/v2/library/hello-world/blobs/sha256:c04b14da8d1441
    # 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
404。

```
    # curl -i -H "Accept: application/vnd.docker.distribution.manifest.v2+json"  -X GET

    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:a18ed77532f6d6781500db650194e0f9396ba5f05f8b50d4046b29
    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:c54a2cc56cbb2f04003c1cd4507e118af7c0d340fe7e2720f70976c4b752
       },
       "layers": [
          {
             "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
             "size": 974,
             "digest": "sha256:c04b14da8d1441880ed3fe6106fb2cc6fa1c9661846ac0266b8a5ec8e
          }
       ]
    }
```

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:a

    * 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:a18ed77532f6d6781500db650194e0f939
    > 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

```
  docker exec -it myregistry_registry_1 /bin/registry garbage-collect --dry-run /etc/reg
```

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

```
  docker exec -it myregistry_registry_1 /bin/registry garbage-collect /etc/registry/con
```

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
      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
```

II 运维中心

III 日志采集

操作场景

日志采集功能是容器服务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

日志采集功能需要为每个集群手动开启。日志采集功能开启后，日志采集 Agent 会在集群内以 DaemonSet 的形

- 采集容器标准输出日志
- 采集容器内文件日志
- 采集主机内文件日志

- 配置日志消费端

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，默认会占用集群
  - 占用 CPU 资源：默认 0.3 核，日志量过大时可根据情况自行调大，建议最大设置为 request 1 核
  - 占用内存资源：默认 250MB，日志量过大时可根据情况自行调大，建议最大设置为 request 1G
  - 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消费端。且以下日志采集功能仅支

概念

- **日志采集 Agent**：TKE 用于采集日志信息的 Agent，基于 Fluentd 开发，在集群内以 DaemonSet 的方式
- 日志采集规则：用户可以使用日志采集规则指定日志的采集源以及将采集的日志发送至指定消费端。
  - 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
- 日志源：包含指定容器日志以及主机路径日志。
  - 在需要采集集群内服务打印到标准输出的日志时，用户将日志的采集源为指定容器日志、所有
  - 在需要采集集群内节点特定路径的日志时，用户可以设定日志的采集源为主机路径日志，例如
- 消费端：日志采集 Agent 在采集指定采集源的日志后，会将采集到的日志发送至用户指定的消费端。
  - 日志采集服务支持用户自建的 Kafka 作为日志的消费端。
  - 日志采集 Agent 会将采集到的日志以 JSON 的形式发送至用户指定的消费端。

操作步骤

采集容器标准输出日志

日志采集功能支持采集 Kubernetes 集群内指定容器的标准输出日志，用户可以根据自己的需求，灵活的配置采

配置方法

197. 登录【容器服务控制台】，选择左侧导航栏中的【日志采集】。
198. 在日志采集页面上方选择地域与集群后，单击【新建】。如下图所示：

199. 在新建日志收集规则页面，选择【容器标准输出】采集类型，并配置日志源。如下图所示：

选择容器标准输出采集类型时，会默认为每条日志添加以下 metadata，其中 log 为原始日志信息。且

字段名
docker.container_id
kubernetes.annotations
kubernetes.container_name
kubernetes.host
kubernetes.labels
kubernetes.namespace_name
kubernetes.pod_id
kubernetes.pod_name
log

200. 配置日志消费端，目前支持用户自建的 Kakfa 服务做为消费端（注：目前仅支持无访问认证的

201. 单击【完成】，完成创建。

采集容器内文件日志

日志采集功能也支持采集集群内指定 pod 内文件的日志。 采集到的日志信息将会以 JSON 格式输出到用户指定

注意： 目前仅支持采集存储在 volume 的日志文件，即需要在工作负载创建时挂载 emptyDir、hostpath 等 volu

配置方法

202. 登录 【容器服务控制台】，选择左侧导航栏中的【日志采集】。

203. 在日志采集页面上方选择地域与集群后，单击【新建】。如下图所示：

204. 指定【容器文件路径】采集类型，并配置日志源。如下图所示：

说明： 用户可以通过指定日志文件的路径来采集 pod 上相应路径的日志文件，路径支持文件路径和通

选择容器文件路径采集类型时，会默认为每条日志添加以下 metadata，其中 message 为原始日志信息

字段名
docker.container_id
kubernetes.annotations

kubernetes.container_name
kubernetes.host
kubernetes.labels
kubernetes.namespace_name
kubernetes.pod_id
kubernetes.pod_name
file
message

205. 配置日志消费端，目前支持用户自建的 Kakfa 服务做为消费端（注：目前仅支持无访问认证的

206. 单击【完成】，完成创建。

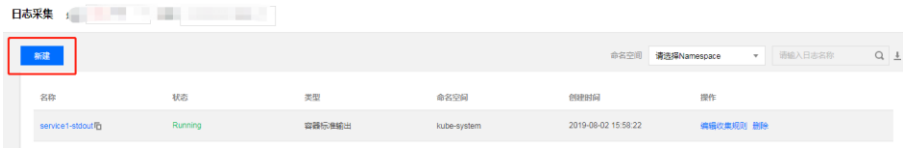
采集主机内文件日志

日志采集功能支持采集集群内所有节点的指定主机路径的日志。用户可以根据自己的需求，灵活的配置所需的
户自定义的 metadata。

配置方法

207. 登录 【容器服务控制台】，选择左侧导航栏中的【日志采集】。

208. 在日志采集页面上方选择地域与集群后，单击【新建】。如下图所示：



209. 在新建日志采集规则页面，指定【节点文件路径】采集类型。如下图所示：

说明： 用户可以通过指定日志文件的路径来采集集群内节点上相应路径的日志文件，路径支持文件路径

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 形式的附加 metadata 将会以 json field 的形式添加到日志记录中。如下图所示：

例如，当不指定附加 metadata 时，采集到的日志如下图所示：

```
{
  "_index": "ccs-log",
  "_type": "logs",
  "_id": "AWDZgNdmPgA9x5PmFg2U",
  "_version": 1,
  "_score": null,
  "_source": {
    "@version": "1",
    "path": "/var/log/example.log",
    "@timestamp": "2018-01-09T05:59:47.891Z",
    "message": "This is an example log line"
  },
  "fields": {
    "@timestamp": [
      1515477587891
    ]
  },
  "sort": [
    1515477587891
  ]
}
```

当用户指定附加 metadata 时，采集到的日志如下图所示：

```
{
  "_index": "ccs-log",
  "_type": "logs",
  "_id": "AWDZgJ8mPgA9x5PmFg2T",
  "_version": 1,
  "_score": null,
  "_source": {
    "@version": "1",
    "path": "/var/log/example.log",
    "@timestamp": "2018-01-09T05:59:33.489Z",
    "message": "This is an example log line",
    "service": "apiserver"
  },
  "fields": {
    "@timestamp": [
      1515477573489
    ]
  },
  "sort": [
    1515477573489
  ]
}
```


相比不指定附加 Metadata 时，附加 Metadata 的 json 日志增加了 key:service。

日志 metadata 含义如下表:

字段名
path
message
自定义 key

配置日志消费端

日志采集功能支持指定用户自建的 Kafka 实例作为日志内容的消费端。日志采集 Agent 会将采集到的日志发送

配置 Kafka 作为日志消费端

目前仅支持无访问认证的 Kafka 实例，且需要保证集群内所有节点都能够访问到用户指定的 Kafka Topic。请填

II 节点管理

III 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
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。

III 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

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

III 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

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

218. 在“选择机型”页面，根据实际需求，选择计费模式、可用区、节点网络和实例类型等。如下图所
219. 单击【下一步】。

220. 在“云主机配置”页面，配置云服务器。如下图所示：
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
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
单击【新建安全组】，放通其他端口。
221. 单击【下一步】。
222. 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。

223. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。

224. 在需要添加已有节点的集群行中，单击【添加已有节点】。如下图所示：
225. 在“选择节点”页面，勾选需要添加的节点，单击【下一步】。
226. 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
  - 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
- 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
  - 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

227. 单击【完成】。

III 移除节点

操作场景

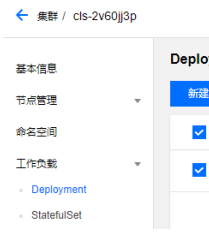
本文档指导您移除集群下的节点。

注意事项

- 包年包月节点移出集群后不销毁。
- 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

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



233. 单击需要移除节点的集群【ID/名称】，进入该集群的管理页面。如下图所示：
234. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。
235. 在节点列表中，选择需要移除节点的节点行，单击【移除】。
236. 在弹出的“您确定要移出以下节点么？”窗口中，单击【确定】，即可完成移除。

III 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

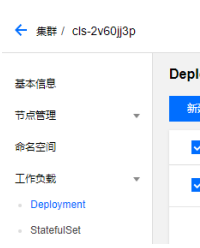
方法一

新增节点时，在"云主机配

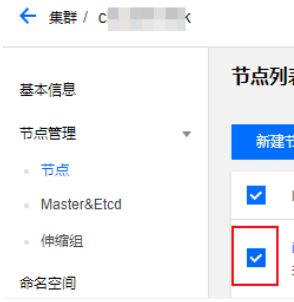
方法二

237. 登录 TKE 控制台。
238. 在左侧导航栏中，单击【集群】，进入"集群管理"页面。

239. 单击需要封锁节点的集群【ID/名称】，进入该集群的管理页面。如下图所示：



240. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。



241. 在节点列表中，选择需要封锁的节点行，单击【封锁】。如下图所示：
242. 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

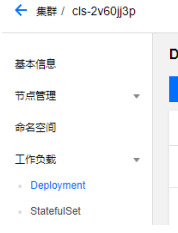
通过执行脚本的方式新增节点时，您可以在该脚本中添加取消封锁节点的命令，即可取消封锁。其示例如下：

```
#!/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

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



245. 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。如下图所示：

246. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。

247. 在节点列表中，选择需要取消封锁的节点行，单击【取消封锁】。如下图所示：
248. 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

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

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

251. 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。如下图所示：
252. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。

253. 在需要驱逐节点的节点行中，单击【更多】>【驱逐】。如下图所示：
254. 在弹出的对话框中，单击【确定】，即可完成驱逐。

III 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

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。如下图所示：

- 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。如下图所示：

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。如下图所示：

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：

III 设置节点 Label

操作场景

本文档指导您设置节点 Label。

使用限制

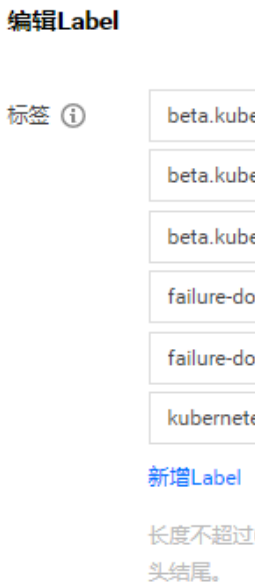
- *kubernetes* 和 *qcloud* 相关标签禁用编辑和删除。
- *kubernetes* 和 *qcloud* 标签为保留键，不支持添加。
- 当前仅支持单个节点设置 Label，不支持批量设置。

操作步骤

控制台设置节点 Label

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

268. 选择需要设置节点 Label 的集群【ID/名称】，进入集群详情。
269. 在左侧导航栏中，选择【节点管理】 > 【节点】，进入“节点列表”页面。
270. 选择需要设置 Label 的节点行，单击【更多】>【编辑标签】。



271. 在弹出的“编辑 Label”窗口中，编辑 Label，单击【提交】。如下图所示：

Kubectl 设置节点 Label

272. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
273. 执行以下命令，设置节点 Label。

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

274. 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
domain.beta.kubernetes.io/zone=200001,kubernetes.io/hostname=172.17.124.5
172.17.124.8        Ready             <none>          12d                   v1.10
domain.beta.kubernetes.io/zone=200001,kubernetes.io/hostname=172.17.124.8
```

II Kubernetes 对象管理

III 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

- 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
  - 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，

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 Kub

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

III 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

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。
- 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

- 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 ResourceQu

```
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

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

III 工作负载

IV 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

Deployment 控制台操作指引

创建 Deployment

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

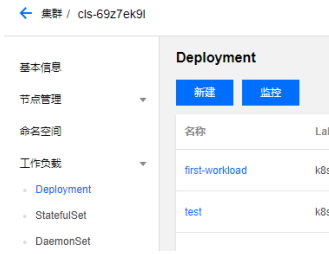
293. 单击需要创建 Deployment 的集群【ID/名称】，进入待创建 Deployment 的集群管理页面。如下

294. 单击【新建】，进入“新建 Workload”页面。如下图所示：

295. 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。
  - 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

        - 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
    - 实例数量：根据实际需求选择调节方式，设置实例数量。
296. 单击【创建 Workload】，完成创建。如下图所示：



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

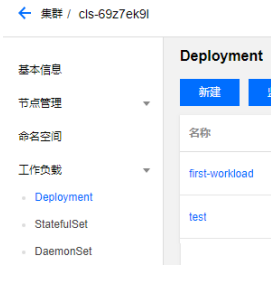
297. 登录 TKE 控制台。
298. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。

299. 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。如下
300. 在需要更新 YAML 的 Deployment 行中，单击【更多】>【编辑 YAML】，进入更新 Deployment

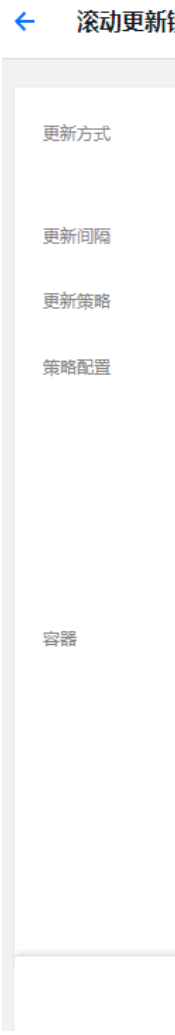
301. 在“更新 Deployment”页面，编辑 YAML，单击【完成】，即可更新 YAML。如下图所示：

更新镜像

302. 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



303. 在需要更新镜像的 Deployment 行中，单击【更新镜像】。如下图所示：



304. 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。如下图所示：
305. 单击【完成】，即可更新镜像。

回滚 Deployment

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

308. 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
309. 单击需要回滚的 Deployment 名称，进入 Deployment 信息页面。

310. 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：
311. 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

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

314. 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 Deploymen

315. 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。
316. 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 TCE 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

323. 参考 YAML 示例，准备 Deployment YAML 文件。
324. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
325. 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
```

326. 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**

- 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、

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

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

329. 执行以下命令，查看 Deployment 的更新历史。

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

330. 执行以下命令，查看指定版本详情。

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

331. 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]
```

IV StatefulSet 管理

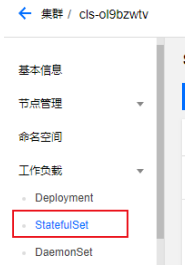
简介

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

StatefulSet 控制台操作指引

创建 StatefulSet

332. 登录 TKE 控制台。
333. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
334. 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。



335. 选择【工作负载】 > 【StatefulSet】，进入 StatefulSet 信息页面。如下图所示：

336. 单击【新建】，进入”新建 Workload“页面。如下图所示：

337. 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

        - 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
    - 实例数量：根据实际需求选择调节方式，设置实例数量。
338. 单击【创建 Workload】，完成创建。

更新 StatefulSet

更新 YAML

339. 登录 TKE 控制台。
340. 在左侧导航栏中，单击【集群】，进入”集群管理“页面。
341. 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。

342. 选择【工作负载】 > 【StatefulSet】，进入 StatefulSet 信息页面。如下图所示：
343. 在需要更新 YAML 的 StatefulSet 行中，单击【编辑 YAML】，进入更新 StatefulSet 页面。
344. 在【更新 StatefulSet】页面，编辑 YAML，单击【完成】，即可更新 YAML。

更新镜像

345. 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 Sta

346. 在需要更新镜像的 StatefulSet 行中，单击【更新镜像】。如下图所示：

347. 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。如下图所示：
348. 单击【完成】，即可更新镜像。

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 TCE 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

354. 参考 YAML 示例，准备 StatefulSet YAML 文件。
355. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
356. 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
```

357. 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
- 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 Stateful

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/ter
```

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。

IV 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

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

DaemonSet 控制台操作指引

创建 DaemonSet

360. 登录 TKE 控制台。
361. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
362. 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。

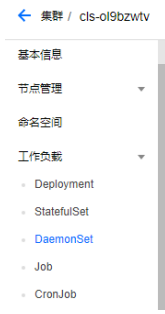
363. 选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。如下图所示：

364. 单击【新建】，进入“新建 Workload”页面。如下图所示：
365. 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
        - 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
366. 单击【创建 Workload】，完成创建。

更新 DaemonSet

更新 YAML

367. 登录 TKE 控制台。
368. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
369. 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。



370. 选择【工作负载】 > 【DaemonSet】，进入“DaemonSet”信息页面。如下图所示：
371. 在需要更新 YAML 的 DaemonSet 行中，单击【编辑 YAML】，进入“更新 DaemonSet”页面。
372. 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。

373. 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 Dae

374. 在需要更新镜像的 DaemonSet 行中，单击【更新镜像】。如下图所示：

375. 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。如下图所示：
376. 单击【完成】，即可更新镜像。

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

- 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 TCE 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

381. 参考 YAML 示例，准备 StatefulSet YAML 文件。
382. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
383. 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
```

384. 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
```

```
frontend   0         0         0         0            0           app=frontend-no
```

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
- 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 Daem

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

方法二

执行以下命令，更新指定容器的镜像。

```
kubectl set image ds/<a href="#">daemonset-name</a>=[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

387. 执行以下命令，查看 DaemonSet 的更新历史。

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

388. 执行以下命令，查看指定版本详情。

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

389. 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]
```

IV Job 管理

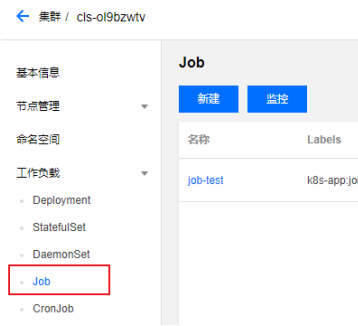
简介

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
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

390. 登录 TKE 控制台。
391. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
392. 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。
393. 选择【工作负载】 > 【Job】，进入“Job”信息页面。如下图所示：



394. 单击【新建】，进入“新建 Workload”页面。如下图所示：

395. 根据实际需求，设置 Job 参数。关键参数信息如下：

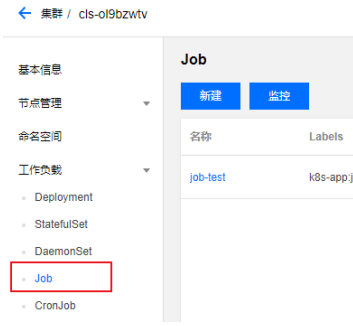
- 工作负载名：自定义。
- 命名空间：根据实际需求进行选择。
- 类型：选择 “Job（单次任务）”。
- Job 设置：根据实际需求，为 Job 的一个 Pod 设置一个或多个不同的容器。
  - 重复次数：设置 Job 管理的 Pod 需要重复执行的次数。
  - 并行度：设置 Job 并行执行的 Pod 数量。
  - 失败重启策略：设置 Pod 下容器异常退出后的重启策略。
    - 选择 Never：不重启容器，直至 Pod 下所有容器退出。
    - 选择 OnFailure：Pod 继续运行，容器将重新启动。
- 实例内容器：根据实际需求，为 Job 的一个 Pod 设置一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
- 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

396. 单击【创建 Workload】，完成创建。

查看 Job 状态

397. 登录 TKE 控制台。
398. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
399. 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。



400. 选择【工作负载】 > 【Job】，进入“Job”信息页面。如下图所示：
401. 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

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 TCE 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

408. 参考 YAML 示例，准备 Job YAML 文件。
409. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
410. 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
```

411. 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]
```

IV CronJob 管理

简介

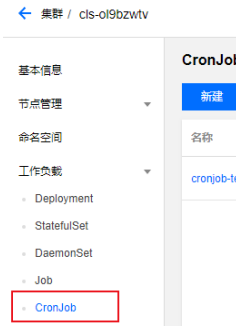
一个 CronJob 对象类似于 crontab（cron table）文件中的一行。它根据指定的预定计划周期性地运行一个 Job。

```
# 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

412. 登录 TKE 控制台。
413. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
414. 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。



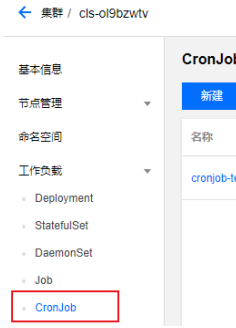
415. 选择【工作负载】 > 【CronJob】，进入“CronJob”信息页面。如下图所示：

416. 单击【新建】，进入“新建 Workload”页面。如下图所示：
417. 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
        - 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
418. 单击【创建 Workload】，完成创建。

查看 CronJob 状态

419. 登录 TKE 控制台。
420. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
421. 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。



422. 选择【工作负载】 > 【CronJob】，进入“CronJob”信息页面。如下图所示：
423. 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 TCE 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

429. 参考 YAML 示例，准备 CronJob YAML 文件。
430. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
431. 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

432. 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 -
```

433. 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]
```

IV 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

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”
- 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：

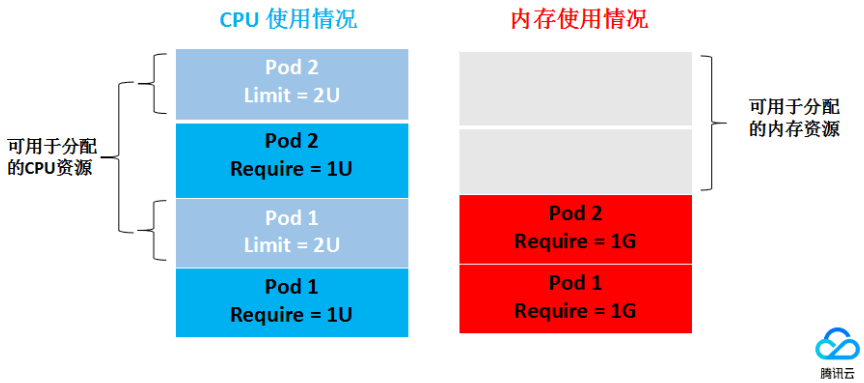
- 内存 Request 作为调度时的依据，在创建时为该容器在节点上分配内存，称为 “已分配内存” 资源。
- 内存资源为不可伸缩资源。当节点上所有容器使用内存均超量时，存在 OOM 的风险。因此不设置 Lim

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
- 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



已经分配的CPU资源为：1U（分配
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

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

服务资源限制推荐

CCS 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

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

```
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

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：

443. 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
444. 推荐值小于您当前容器已经配置的 Request 或者 Limit。

说明：

445. 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
446. 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
447. 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。

IV 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

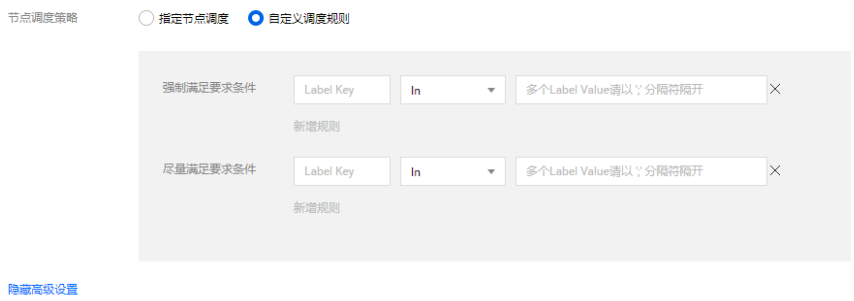
- 设置工作负载高级设置中的调度规则，且集群的 Kubernetes 版本必须是 1.7 以上的版本。
- 为确保您的 Pod 能够调度成功，请确保您设置的调度规则完成后，节点有空余的资源用于容器的调度。
- 使用自定义调度功能时，需要为节点设置对应 Label。详情请参见 设置节点 Label。

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

- **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
- 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

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 P

IV 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

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
- **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
  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
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。
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/h`

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

- 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
- 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 ht`

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，

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
- **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
- **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

  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

IV 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，

- 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。

命令和参数的使用

单击【详情】，了解如何将 docker run 命令适配到 TCE 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
建时的默认命令（即 “Entrypoint” 和 “CMD”）。其规则如下：

镜像 Entrypoint	镜像 CMD
[ls]	[/home]
[ls]	[/home]
[ls]	[/home]
[ls]	[/home]

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

III 服务

IV 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

- 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.`
- 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。

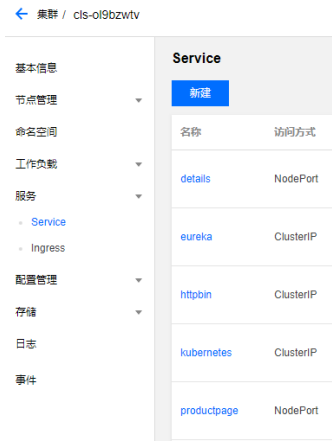
Service 控制台操作指引

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。

创建 Service

485. 登录 TKE 控制台。
486. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
487. 单击需要创建 Service 的集群【ID/名称】，进入待创建 Service 的集群管理页面。



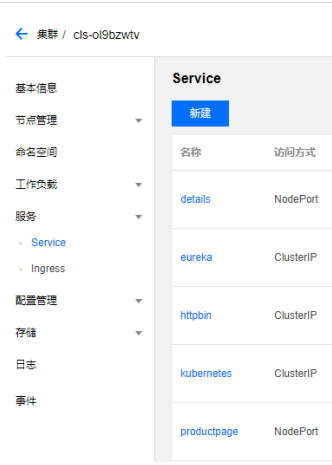
488. 选择【服务】 > 【Service】，进入“Service”信息页面。如下图所示：

489. 单击【新建】，进入“新建 Service”页面。如下图所示：
490. 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。
    - 服务访问方式：根据实际需求，选择对应的访问方式。
    - 端口映射：根据实际需求进行设置。
491. 单击【创建服务】，完成创建。

更新 Service

更新 YAML

492. 登录 TKE 控制台。
493. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
494. 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。



495. 选择【服务】 > 【Service】，进入“Service”信息页面。如下图所示：
496. 在需要更新 YAML 的 Service 行中，单击【编辑 YAML】，进入更新 Service 页面。
497. 在“更新 Service”页面，编辑 YAML，单击【完成】，即可更新 YAML。

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

- 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 TCE TKE 的额外增强能力。
- 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 TCE 提供的负载均衡器公开服务，默认创建公网 CLB， 指定 annotations 可
  - ExternalName：将服务映射到 DNS，仅适用于 kube-dns1.7 及更高版本。

annotations: 使用已有负载均衡器创建公网/内网访问的 Service

如果您已有的包年包月传统型 CLB 为空闲状态，需要提供给 TKE 创建的 Service 使用，您可以通过以下 annota

```
metadata:
  annotations:
    service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
```

annotations: 指定节点绑定 Loadbalances

如果您的集群规模较大，入口类型的应用需设置亲和性调度到部分节点， 您可以通过配置 Service 的 Loadbala

```
metadata:
  annotations:
    service.kubernetes.io/qcloud-loadbalancer-backends-label: `key in (value1, value2)
```

- 建议配合工作负载的亲和性调度使用。
- 前提条件是 Node 已根据业务需求设置 Label。
- 采用原生 LabelSelector 格式如：
  - service.kubernetes.io/qcloud-loadbalancer-backends-label: key1=values1, key2=values2
  - service.kubernetes.io/qcloud-loadbalancer-backends-label: key1 in (value1),key2 in (value2)
  - service.kubernetes.io/qcloud-loadbalancer-backends-label: key in (value1, value2)
  - service.kubernetes.io/qcloud-loadbalancer-backends-label: key1, key2 notin (value2)
- 增量的节点若匹配，将自动绑定到该 Loadbalance。
- 修改存量节点的 Label， 根据匹配规则将动态绑定和解绑 Loadbalance。

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：

- service.kubernetes.io/qcloud-loadbalancer-internet-charge-type 公网带宽计费方式，可
- service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out 带宽上限，范围

```
metadata:
annotations:
  service.kubernetes.io/qcloud-loadbalancer-internet-charge-type: TRAFFIC_POSTPAI
  service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out: "10"
```

创建 Service

512. 参考 YAML 示例，准备 StatefulSet YAML 文件。
513. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
514. 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
```

515. 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

方法一

执行以下命令，更新 Service。

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

方法二

516. 手动删除旧的 Service。
517. 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]
```

IV 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
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

518. 登录 TKE 控制台。
519. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
520. 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。

521. 选择【服务】 > 【Ingress】，进入“Ingress”信息页面。如下图所示：

522. 单击【新建】，进入“新建 Ingress”页面。如下图所示：
523. 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。
524. 单击【创建 Ingress】，完成创建。

更新 Ingress

更新 YAML

525. 登录 TKE 控制台。
526. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
527. 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。

528. 选择【服务】 > 【Ingress】，进入“Ingress”信息页面。如下图所示：
529. 在需要更新 YAML 的 Ingress 行中，单击【编辑 YAML】，进入更新 Ingress 页面。
530. 在“更新 Ingress”页面，编辑 YAML，单击【完成】，即可更新 YAML。

更新转发规则

531. 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。

532. 选择【服务】 > 【Ingress】，进入“Ingress”信息页面。如下图所示：

533. 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。如下图
534. 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（ngin
    ## kubernetes.io/ingress.subnetId: subnet-xxxxxxxx  ##若是创建 CLB 类型内网 ingress 需指
  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 TCE 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

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
- 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

541. 参考 YAML 示例，准备 Ingress YAML 文件。
542. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
543. 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
```

544. 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

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

```
kubectl create/apply
```

III 配置

IV ConfigMap 管理

简介

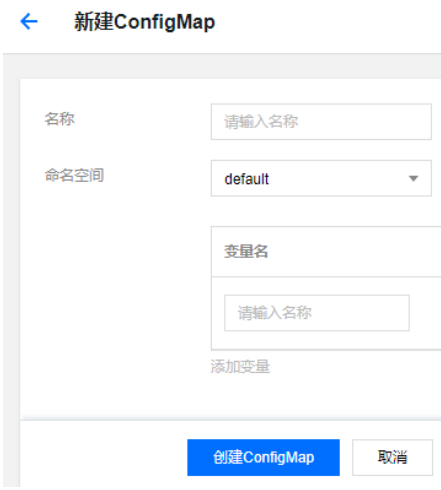
通过 ConfigMap 您可以将配置和运行的镜像进行解耦，使得应用程序有更强的移植性。ConfigMap 是有 key-v

ConfigMap 控制台操作指引

创建 ConfigMap

547. 登录 TKE 控制台。
548. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
549. 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。

550. 选择【配置管理】 > 【ConfigMap】，进入“ConfigMap”信息页面。如下图所示：



551. 单击【新建】，进入“新建 ConfigMap”页面。如下图所示：
552. 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。
553. 单击【创建 ConfigMap】，完成创建。

使用 ConfigMap

方式一：数据卷使用 ConfigMap 类型

554. 登录 TKE 控制台。
555. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
556. 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。

557. 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】
558. 单击【新建】，进入“新建 Workload”页面。

559. 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】，

560. 选择【使用 ConfigMap】方式，填写名称，单击【选择配置项】。如下图所示：
561. 在弹出的“设置 ConfigMap”窗口中，配置挂载点，单击【确认】。
562. 单击【创建 Workload】，完成创建。

方式二： 环境变量中使用 ConfigMap 类型

563. 登录 TKE 控制台。
564. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
565. 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。

566. 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】
567. 单击【新建】，进入“新建 Workload”页面。

568. 根据页面信息，设置工作负载名、命名空间等信息。并在 “实例内容器” 的 “环境变量” 中，单击

569. 选择 “ConfigMap” 环境变量方式，并根据实际需求选择资源。如下图所示：
570. 单击【创建 Workload】，完成创建。

更新 ConfigMap

571. 登录 TKE 控制台。
572. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
573. 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。

574. 选择【配置管理】 > 【ConfigMap】，进入“ConfigMap”信息页面。如下图所示：
575. 在需要更新 YAML 的 ConfigMap 行中，单击【编辑 YAML】，进入“更新 ConfigMap”页面。
576. 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 TCE 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

581. 参考 YAML 示例，准备 ConfigMap YAML 文件。
582. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
583. 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
```

584. 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
```

IV Secret 管理

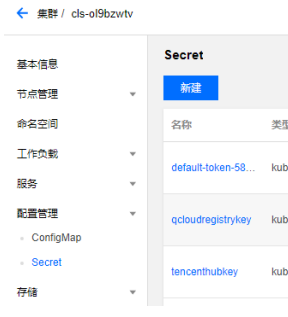
简介

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

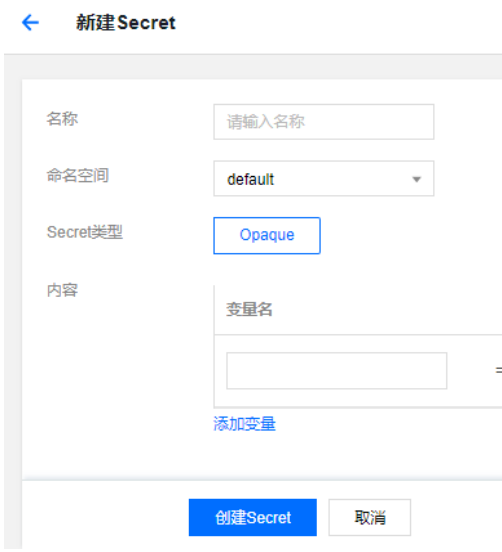
Secret 控制台操作指引

创建 Secret

587. 登录 TKE 控制台。
588. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
589. 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。



590. 选择【配置管理】 > 【Secret】，进入“Secret”信息页面。如下图所示：



591. 单击【新建】，进入“新建 Secret”页面。如下图所示：
592. 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。
593. 单击【创建 Secret】，完成创建。

使用 Secret

方式一：数据卷使用 Secret 类型

594. 登录 TKE 控制台。
595. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
596. 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。

597. 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】
598. 单击【新建】，进入“新建 Workload”页面。

599. 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】，

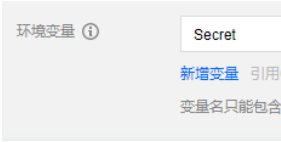
600. 选择 “使用 Secret” 方式，填写名称，单击【选择 Secret】。如下图所示：
601. 在弹出的 “设置 Secret” 窗口中，配置挂载点，单击【确认】。
602. 单击【创建 Workload】，完成创建。

方式二：环境变量中使用 Secret 类型

603. 登录 TKE 控制台。
604. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
605. 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。

606. 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】
607. 单击【新建】，进入“新建 Workload”页面。

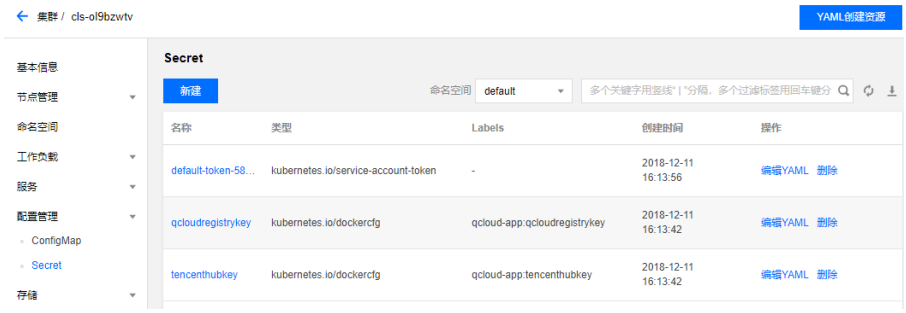
608. 根据页面信息，设置工作负载名、命名空间等信息。并在 “实例内容器” 的 “环境变量” 中，单击



609. 选择 “Secret” 环境变量方式，并根据实际需求选择资源。如下图所示：
610. 单击【创建 Workload】，完成创建。

更新 Secret

611. 登录 TKE 控制台。
612. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
613. 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。
614. 选择【配置管理】 > 【Secret】，进入“Secret”信息页面。如下图所示：



615. 在需要更新 YAML 的 Secret 行中，单击【编辑 YAML】，进入更新 Secret 页面。
616. 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。

Kubectl 操作 Secret 指引

创建 Secret

方式一：通过指定文件创建 Secret

617. 执行以下命令，获取 Pod 的用户名和密码。

```
618. $ echo -n 'username' > ./username.txt
     $ echo -n 'password' > ./password.txt
```

619. 执行 Kubectl 命令，创建 Secret。

```
620. $ kubectl create secret generic test-secret --from-file=./username.txt --from-fi
     secret "testSecret" created
```

621. 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
```

III 存储

IV 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 HostPa
  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 TCE 的 文件存储 CFS，也可使用自建的文件存储 NFS。
- **使用已有 PersistentVolumeClaim**：使用已有 PersistentVolumeClaim 声明工作负载的存储，自动分配或
- **使用新的 PersistentVolumeClaim**：新建一个 PersistentVolumeClaim 声明工作负载的存储，自动分配或
- **使用 TCE 硬盘**：TCE 基于 CBS 扩展的 Kubernetes 的块存储插件。您可以指定一块 TCE 的 CBS 云硬盘挂
- **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
- **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

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/kuberne
- 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

633. 登录 TKE 控制台。
634. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
635. 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。

636. 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】
637. 单击【新建】，进入“新建 Workload”页面。

638. 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】，
639. 根据实际需求，选择数据卷的存储方式，配置挂载点。
640. 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 Persisten
- spec.containers.volumeMounts：填写数据卷的挂载点。

IV 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

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。

646. 登录 TKE 控制台。

647. 在左侧导航栏单击【集群】，进入“集群管理”页面。

648. 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。

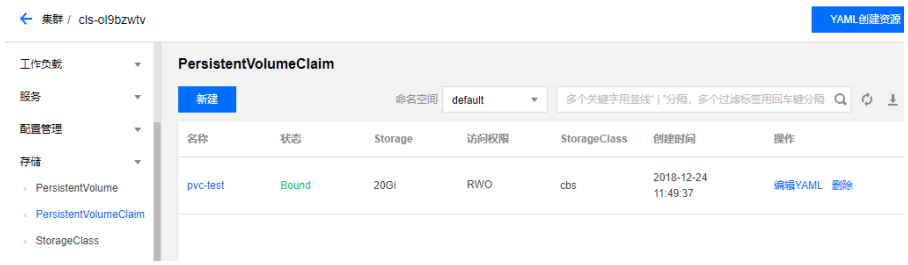
649. 选择【存储】 > 【PersistentVolume】，进入“PersistentVolume”信息页面。如下图所示：

650. 单击【新建】，进入“新建 PersistentVolume”页面。如下图所示：

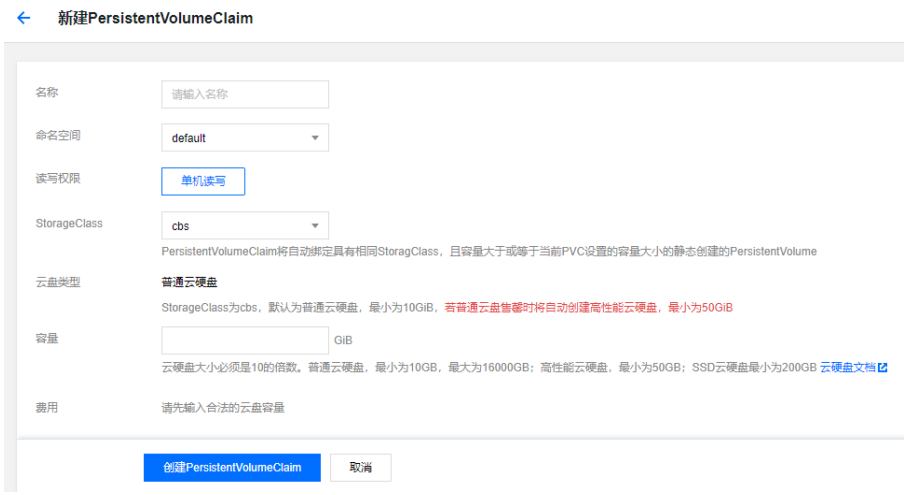
651. 根据实际需求，设置 PV 参数。关键参数信息如下：
   - 来源设置：根据实际需求进行选择，默认为 “静态创建”。
   - 名称：自定义。
   - 选择云盘：根据实际需求进行选择。
   - StorageClass：根据实际需求进行选择。
652. 单击【创建 PersistentVolume】，完成创建。

创建 PVC

653. 登录 TKE 控制台。
654. 在左侧导航栏单击【集群】，进入“集群管理”页面。
655. 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。
656. 选择【存储】 > 【PersistentVolumeClaim】，进入“PersistentVolumeClaim”信息页面。如下图所



657. 单击【新建】，进入【新建 PersistentVolumeClaim】页面。如下图所示：

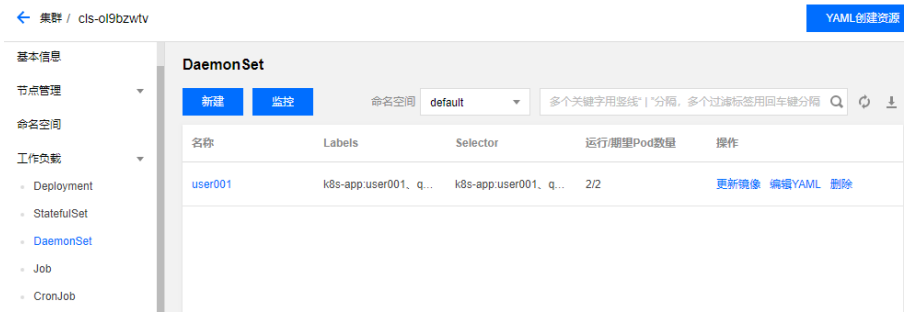


658. 根据实际需求，设置 PVC 参数。关键参数信息如下：
   - 名称：自定义。
   - 命名空间：根据实际需求进行选择命名空间类型。
   - StorageClass：根据实际需求进行选择。
   - 容量：根据实际需求进行设置。
659. 单击【创建 PersistentVolumeClaim】，完成创建。

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

660. 登录 TKE 控制台。
661. 在左侧导航栏单击【集群】，进入“集群管理”页面。
662. 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。
663. 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】



664. 单击【新建】，进入“新建 Workload”页面。

665. 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】，

666. 选择【使用已有 PVC】方式，填写名称，选择已有的 PVC。如下图所示：

667. 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

（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
```

IV 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。TCE TKE 服务默认提供块存储类型

StorageClass 控制台操作指引

创建 StorageClass

671. 登录 TKE 控制台。
672. 在左侧导航栏中，单击【集群】，进入“集群管理”页面。
673. 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。
674. 选择【存储】 > 【StorageClass】，进入“StorageClass”信息页面。
675. 单击【新建】，进入“新建 StorageClass”页面。
676. 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：根据实际需求进行选择。
    - 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。
677. 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

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: 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，PREPAID 模式（包年包月：仅支持 Retain 保留的回收策略），默认是 POS
  # 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，
- 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
- 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/v1beta1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: nginx
        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
```

II 监控与告警

III 监控告警概述

概述

良好的监控环境为 TCE 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

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。

告警

为了方便您及时发现容器服务的异常状况，以保证您业务的稳定性和可靠性。建议您为所有生产集群配置必要

说明： 容器服务提供的监控和告警功能主要覆盖 Kubernetes 对象的核心指标或事件，请结合 云监控 提供的基

III 设置监控告警

操作场景

容器服务平台支持为集群设置集群、节点、Pod 3 个维度的告警。为您的集群设置合理的告警，有助于避免和

注意：

- 容器服务仅提供 Kubernetes 集群健康运行最关心的指标或者事件告警。
- 容器服务中设置的告警只能在容器服务控制台管理。
- 容器服务和 [云监控] 中设置的告警可以共存，建议云服务器、负载均衡等相关资源的告警依然通过云监

操作步骤

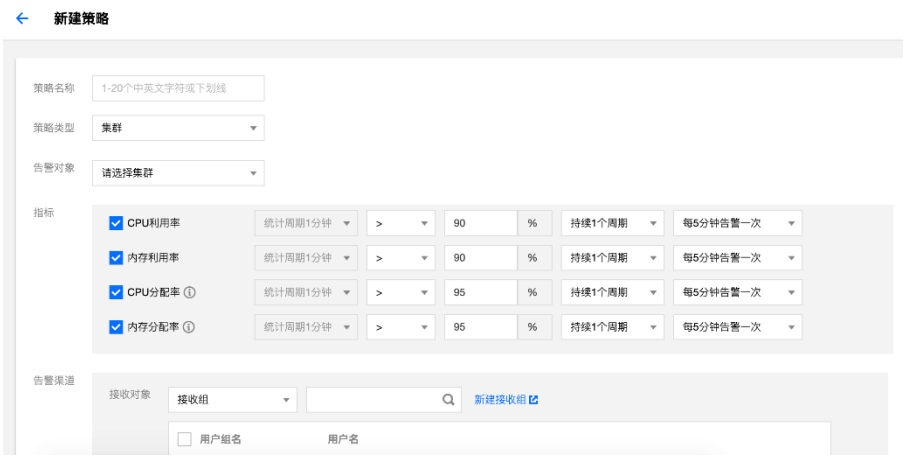
根据您的实际需求选择告警设置。

说明： 集群列表中尚未设置任何告警的集群会有特殊提示“未配告警”，设置告警后该提示消失。

685. 选择【容器告警】，即可查看当前集群的告警列表。如下图所示：

686. 单击【新建】，即可创建新的告警策略。如下图所示：

注意： 所有告警仅对所配置的集群生效。



III 查看监控数据

操作场景

TCE 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。如下图所示：

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。

693. 选择需要查看的集群【ID/名称】，进入该集群的管理页面。
694. 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。如下图所示：

- 选择【Master&Etcd】>【监控】，即可进入“Master&Etcd 监控”页面，查看监控信息。如下图所

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。

695. 选择需要查看的集群【ID/名称】，进入该集群的管理页面。
696. 选择【节点管理】>【节点】，进入节点列表页面。
697. 根据实际需求，选择查看节点内 Pod 指标的方式。
   - 在节点列表中查看 Pod 指标。
     a. 单击【监控】，进入“节点监控”页面。

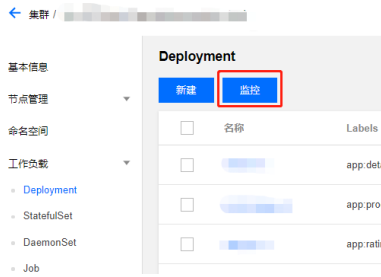
b. 单击【Pod】，将【所属节点】选择为您想查看的节点，即可查看到该节点内 Pod 的监

- 在节点详情页面中查看 Pod 指标。
  . 选择需要查看的节点【ID/节点名】，进入该节点的管理页面。

a. 选择【Pod 管理】页签，单击【监控】，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

698. 选择需要查看的集群【ID/名称】，进入该集群的管理页面。
699. 选择【工作负载】>【任意类型工作负载】。例如，选择【Deployment】，进入 Deployment 管



700. 单击【监控】，即可查看该工作负载的监控信息。如下图所示：

查看工作负载内 Pod 指标

701. 选择需要查看的集群【ID/名称】，进入该集群的管理页面。
702. 选择【工作负载】>【任意类型工作负载】。例如，选择【Deployment】，进入 Deployment 管理
703. 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。

704. 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。如

查看 Pod 内 Container 指标

705. 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。
706. 选择【Pod 管理】页签，单击【监控】，进入“Pod 监控”页面。

707. 单击【Container】，将【所属 Pod】选择为您想查看的 Pod，即可查看该 Pod 内 Container 的监

III 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
CPU 利用率	%
内存利用率	%

Master&Etcd 和普通节点监控指标

监控指标	单位
Pod 重启次数	次
异常状态	-
CPU 利用率	%
内存利用率	%
内网入带宽	bps
内网出带宽	bps
外网入带宽	bps
外网出带宽	bps
TCP 连接数	个

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
Pod 重启次数	次
CPU 使用量	核
CPU 利用率（占集群）	%
内存使用量	B
内存利用率（占集群）	%
网络入带宽	bps
网络出带宽	bps
网络入流量	B
网络出流量	B
网络入包量	个/s
网络出包量	个/s

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
异常状态	-
CPU 使用量	核
CPU 利用率（占节点）	%
CPU 利用率（占 Request）	%
CPU 利用率（占 Limit）	%
内存使用量	B
内存使用量（不包含 Cache）	B
内存利用率（占节点）	%
内存利用率（占节点，不包含 Cache）	%
内存利用率（占 Request）	%
内存利用率（占 Request，不包含 Cache）	%
内存利用率（占 Limit）	%
内存利用率（占 Limit，不包含 Cache）	%

网络入带宽	b
网络出带宽	b
网络入流量	B
网络出流量	B
网络入包量	个
网络出包量	个

Container 监控指标

监控指标
CPU 使用量
CPU 利用率（占节点）
CPU 利用率（占 Request）
CPU 利用率（占 Limit）
内存使用量
内存使用量（不包含 Cache）

内存利用率（占节点）
内存利用率（占节点，不包含 Cache）
内存利用率（占 Request）
内存利用率（占 Request，不包含 Cache）
内存利用率（占 Limit）
内存利用率（占 Limit，不包含 Cache）
块设备读带宽
块设备写带宽
块设备读 IOPS
块设备写 IOPS

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

CPU 利用率	%	集
内存利用率	%	集
CPU 分配率	%	集
内存分配率	%	集
Apiserver 正常	-	Ap
Etcd 正常	-	Et
Scheduler 正常	-	Sc
Controll Manager 正常	-	Co

节点告警指标

监控指标	单位
CPU 利用率	%
内存利用率	%
节点上 Pod 重启次数	次
Node Ready	-

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
CPU 利用率（占节点）	%
内存利用率（占节点）	%
实际内存利用率（占节点）	%
CPU 利用率（占 Limit）	%
内存利用率（占 Limit）	%
实际内存利用率（占 Limit）	%
Pod 重启次数	次
Pod Ready	-
CPU 使用量	核
内存使用量	MB
实际内存使用量	MB

II Helm 应用

III 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

III 添 Helm 应用管理

操作场景

您可以通过控制台管理 Helm 应用的创建、删除、修改等操作，也可以通过本地安装 Helm 客户端连接到集群

前提条件

- 已有 TCE TKE 集群，且拥有 0.28 核 CPU，180MiB 内存的资源。
- 已在集群内开通 Helm 应用管理功能。
- 仅支持 Kubernetes 1.8 以上版本的集群。
- 若未开通 Helm 应用管理功能，请在 Helm 应用中申请开通。
- 开通 Helm 应用管理功能后，将在集群内安装 Helm tiller 相关组件。

操作步骤

创建 Helm 应用

713. 登录 TKE 控制台。
714. 在左侧导航栏中，选择【应用中心】，单击【Helm 应用】，进入 Helm 应用管理页面。
715. 单击【新建】，进入【新建 Helm 应用】页面。

716. 输入应用名，选择 “所在地域”、“运行集群” 以及需要安装的 Helm Chart 和对应的版本。如下图
库。 >! 当您选择 “其他” 类型来源时，Chart_url 属性必须设置为以 http 开头 tgz 结尾的参数值。
717. 单击【完成】。

更新 Helm 应用

718. 前往 Helm 应用控制台。
719. 在【Helm 应用】列表中，选择需要更新的 Helm 应用，单击【更新应用】。

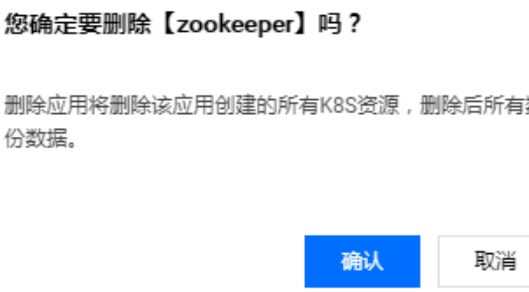
720. 在弹出的【更新 Helm 应用】窗口中，选择需要更新的版本，并根据业务需求填写自定义参数。
721. 单击【完成】。

回滚 Helm 应用

722. 前往 Helm 应用控制台。
723. 在【Helm 应用】列表中，单击需要更新的 Helm 应用的应用名，进入应用详情页面。
724. 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

725. 前往 Helm 应用控制台。
726. 在【Helm 应用】列表中，选择需要删除的 Helm 应用，单击【删除】。



727. 在弹出的提示框中，单击【确认】。如下图所示：

III Update 本地 Helm 客户端连接集群

操作场景

本文档指导您通过本地 Helm 客户端连接集群。

操作步骤

下载 Helm 客户端

执行以下命令，下载 Helm 客户端。

```
curl -O https://storage.googleapis.com/kubernetes-helm/helm-v2.10.0-linux-amd64.tar.gz
tar xzvf helm-v2.10.0-linux-amd64.tar.gz
sudo cp linux-amd64/helm /usr/local/bin/helm
```

配置 Helm 为 Client-only

执行以下命令，将 Helm 配置为 Client-only。

```
helm init --client-only
```

内网通过 Helm 客户端连接集群

目标集群节点

您可以直接使用。

非目标集群节点

728. 执行以下命令，将目标集群 Tiller 服务的 type 修改为内网 Loadbalancer 模式。

请将以下命令中 “service.kubernetes.io/qcloud-loadbalancer-internal-subnetid” 的值替换为需要生产 CL

```
kubectl patch svc $(kubectl get svc -l app=helm,name=tiller -n kube-sys
88888888"}},"spec":{"type":"LoadBalancer"}}'
```

729. 执行以下命令，在目标集群节点上获取 $EXTERNALIP。

```
kubectl get svc -l app=helm,name=tiller -n kube-system -o=jsonpath={.items[0].sta
```

730. 执行以下命令，在目标集群节点上获取 $PORT。

```
kubectl get svc -l app=helm,name=tiller -n kube-system -o=jsonpath={.items[0].spe
```

731. 执行以下命令，在 Helm 客户端节点上导入环境变量。

```
export HELM_HOST=$EXTERNALIP:$PORT
```

732. 执行以下命令，命令验证 Helm 客户端。

```
733. [centos ~]# helm ls
734. NAME         REVISION    UPDATED                     STATUS      CHART
   wordpress    1           Mon Jan 21 14:22:30 2019    DEPLOYED    wordpress-5.0.1
```

II 健康检查

操作场景

集群健康检查功能是容器服务（Tencent Kubernetes Engine，TKE）为集群提供检查各个资源状态及运行情况的

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

主要检查项目

检查类别	检查项
资源状态	kube-apiserver 的状态

	kube-scheduler 的状态
	kube-controller-manager 的状态
	etcd 的状态
	kubelet 的状态
	kube-proxy 的状态
	dockerd 的状态
	master 节点的状态
	worker 节点的状态
	各个工作负载的状态
运行情况	kube-apiserver 的参数配置
	kube-scheduler 的参数配置
	kube-controller-manager 的参数配置

	etcd 的参数配置
	master 节点的配置合理性
	node 高可用
	工作负载的 Request 和 Limit 配置
	工作负载的反亲和性配置
	工作负载的 PDB 配置
	工作负载的健康检查配置
	HPA-IP 配置

操作步骤

741. 登录 容器服务控制台，选择左侧导航栏中的【健康检查】。

742. 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

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：

743. 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：

744. 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】

I 最佳实践

II 构建简单 web 应用

操作场景

本文档指导您使用 TCE 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，r

操作步骤

创建容器集群

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

749. 单击【新建】，进入“创建集群”页面。如下图所示：

750. 根据实际需求，填写以下参数。

- 集群名称：自定义。
- Kubernetes 版本：指定 Kubernetes 版本。
- 所在地域：指定集群的位置。
- 集群网络：指定集群的节点网络，节点网络必须位于某个 VPC 内，如果您当前没有 VPC，请先
- 容器网络插件：根据实际情况选择。
- 容器网络：指定容器网络，集群内容器的 IP 将从该网络分配。

751. 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
752. 单击【下一步】。

753. 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：
754. 单击【下一步】。
755. 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

756. 在左侧导航栏中，选择【服务】 > 【Service】，进入”Service“页面。
757. 单击【新建】，进入”新建服务“页面。

758. 根据实际需求，填写以下参数。如下图所示：
`ccr.ccs.tencentyun.com/library/redis`。 - 镜像版本：latest。 - CPU/内存限制：（可选）设置 CPU 和内
redis-master 以及端口 6379 访问到 master 容器。
759. 单击【创建服务】。

创建 redis-slave 服务

760. 单击【新建】，进入”新建服务“页面。

761. 根据实际需求，填写以下参数。如下图所示：
`ccr.ccs.tencentyun.com/library/gb-redisslave`。 - 镜像版本：latest。 - CPU/内存限制：（可选）设置 C
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
762. 单击【创建服务】。

创建 frontend 服务

763. 单击【新建】，进入”新建服务“页面。

764. 根据实际需求，填写以下参数。如下图所示：

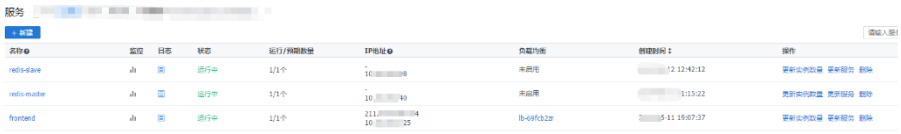
`ccr.ccs.tencentyun.com/library/gb-frontend`。 - 镜像版本：latest。 - CPU/内存限制：（可选）设置 CP

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 fron

765. 单击【创建服务】。

查看服务

在 左 侧 导 航 栏 中 ， 单 击 【



- 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
- 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

返回如下图所示页面，即表示可以正常访问 frontend 服务。
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

- 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
- 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

通过示例的 Web App，需要注意以下两点:

770. 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
（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
771. 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

II 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 TCE 容器服务平台中运行时，对于 dock

一、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
```

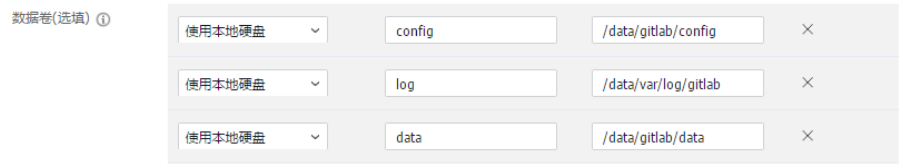
`-d`:容器在后台运行。容器平台都是以后台的形式来运行容器，所以本参数不需要在容器控制台指定。

`-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

`--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，

-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

首先我们



接着我们在容器的高级

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

--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

二、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
```

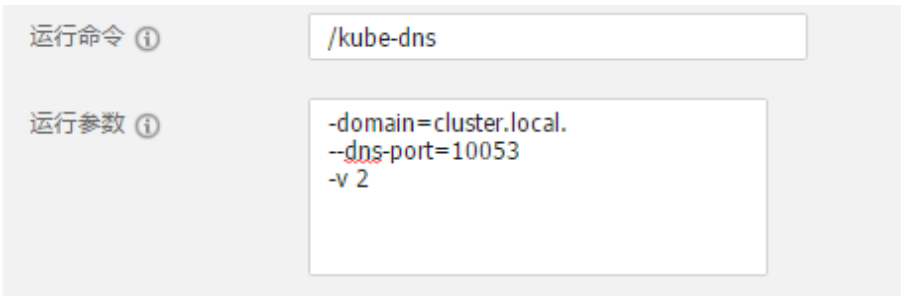
这里用户希望为容器添加两个环境变量，在我们的容器控制台创建服务时，在容器的高级设置里，可以添加容

三、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=
```

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-por



II 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
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

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

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。

I 访问管理

II 概述

如果您在 TCE 中使用到了容器服务（Tencnet Kubernetes Engines，TKE），且该服务虽然由不同的人管理，但都

- 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

简介

访问管理（Cloud Access Management，CAM）是 TCE 提供的一套 Web 服务，主要用于帮助客户安全管理 TC

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

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

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

II TKE 镜像仓库资源级权限设置

容器镜像服务权限介绍

TCE 容器镜像的地址格式是：`ccr.ccs.tencentyun.com/${namespace}/${name}:${tag}`。 镜像仓库的权限

- `${namespace}`: 镜像仓库所属命名空间。
- `${name}`: 镜像仓库名称。

说明：命名空间 `${namespace}` 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
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
如果您需要细致地管理
容器镜像服务权限基于 TCE 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
- 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，
  - `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
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
- **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：

792. 创建自定义策略 CAM 文档。
    i. 使用开发商账号登录 TCE-控制台。
    ii. 进入 CAM 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

iii. 选择【按策略语法创建】>【空白模板】。

iv. 单击【下一步】，进入【编辑策略】页面。

v. 设置策略名称，并将以下内容填入【编辑策略内容】编辑框中。

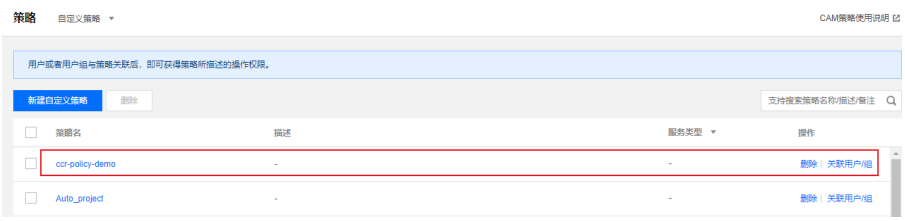
```
vi.   {
vii.      "version": "2.0",
viii.     "statement": [{
ix.         "action": "ccr:CreateRepository",
```

```
        "resource": "qcs::ccr:::repo/*",
        "effect": "allow"
    }]
}
```

例如，将策略名称设置为 ccr-policy-demo，如下图所示：

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。

xiii. 单击【创建策略】，结束策略创建过程。



793. 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

- qcs::ccr::: 为固定格式，表示开发商的 TCE 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

```
{
    "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

```
{
    "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 :
action:

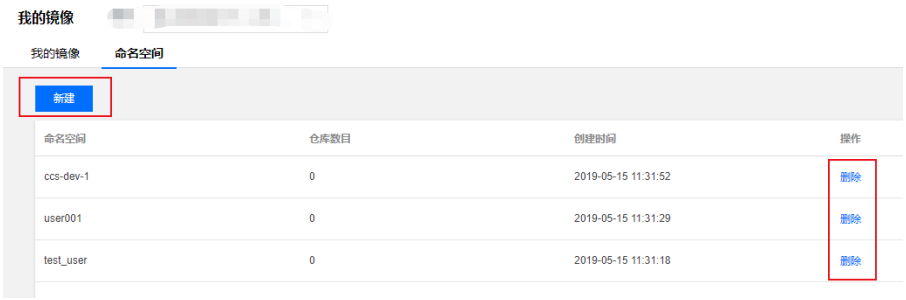
- `ccr:pull` 使用 docker 命令行 pull 镜像
- `ccr:push` 使用 docker 命令行 push 镜像

命名空间权限

resource :
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

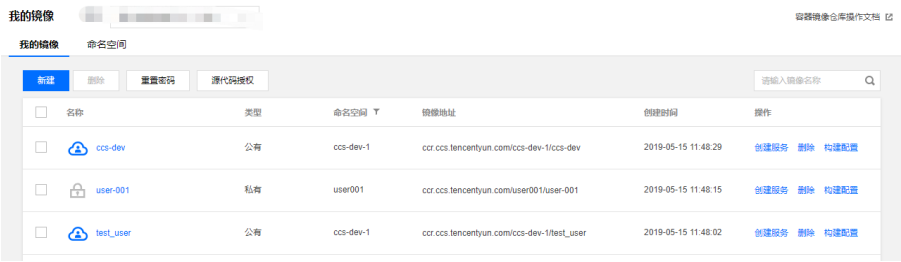


镜像仓库权限

resource :
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



! 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"
   }]
}
```

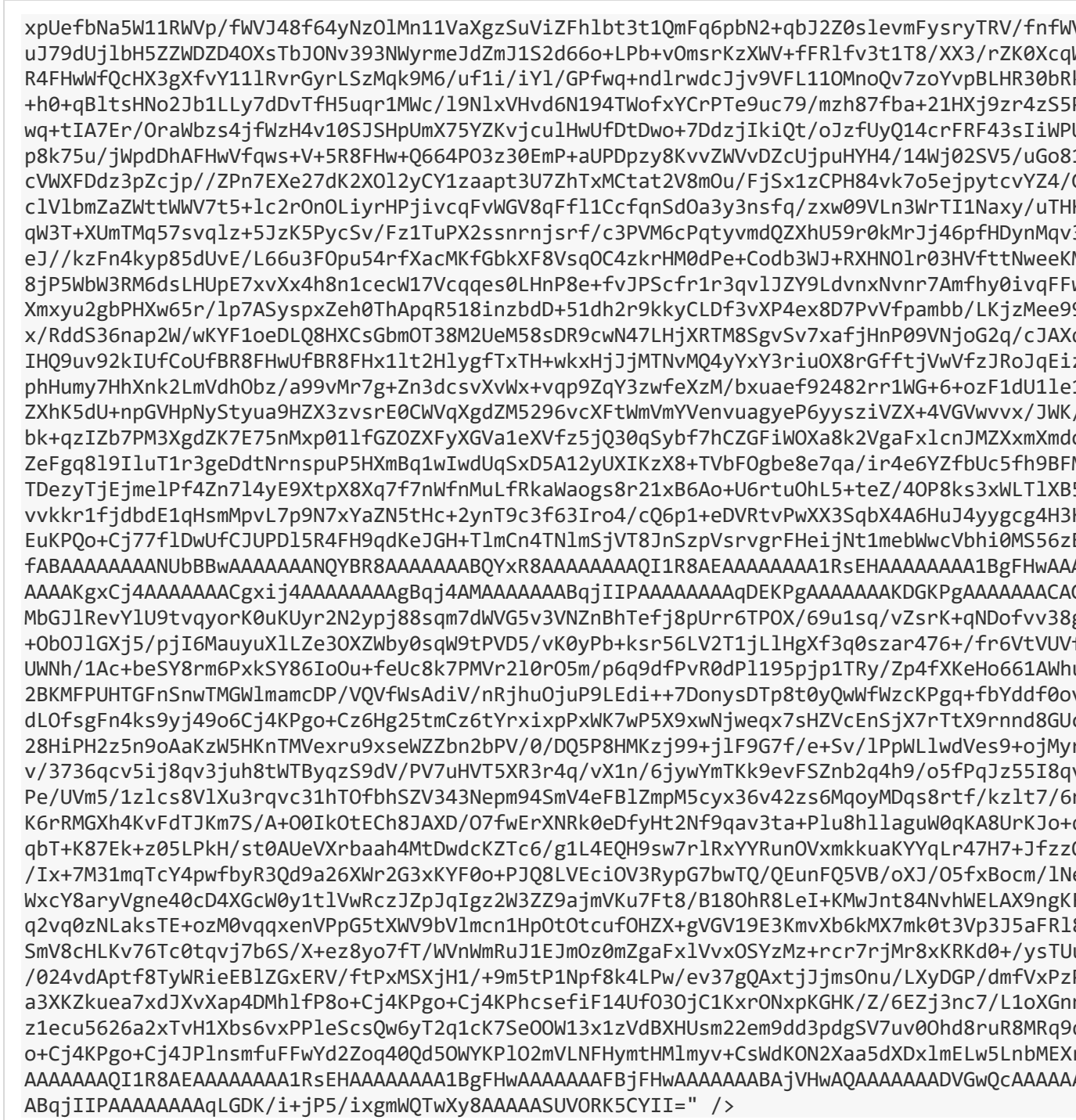
```
<span id="Tag"></span>
####  镜像 Tag 权限

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
<img
src="
ff7LUqkSIkSSUnUu1QPVanUUofjnIgT8f+fOBGZJJOsjKzvB/ihyMzIeGVExolfpaS3IgAAAAAAAAC1RcEHAAAAA
AAAAADsWqiMIq1PFQq+PXr9+nX0pz/9Kfruu++il99+m2X74cPo6bPPo6++/toGAAAAAACgE4XKKNL6VKHg2wVT6
3tffvllsMwLJVTsyZhv9H0eT7d3j6KT//wfRX//n/8xepg+smf3/xj9T/8gnuc/+GfRyfvpYwAAAAAAAAF+EdVuu
ffmz1H0j99LZmbKvZmt9IkfgFd/+lO0fvtOsMSrSqjMC8V8k29Pf1x35t9Gf/8f/KPo5zPp761AwQcAAAAAAJrkP
3fblKZY/jWYRi0nXxeT4jcF9fOUggAAAAAA/LD4RVQrI/94rSnbbIFWVvClZaCZXn47r8BNk/63quAz83Hzt+VeH
l88vSMLPlP4OW56/5uB5rXmNW6d7M/pc/uVKpUF37/4tDizZvL9yQun7JtmhW+9NTFN+u2+X52bPrCC7+HtBbvMr
5mcmY/+0//sPxfr5RV6ofVOH8v/2Kz/jbrQNjYxTbYs/U08v7xzv7f0H/gAAAAAAAC1EuqGWhFXvBV+N2WeX/CJy
f8obuiP/fp/3PcgUqWz/pGN0LfcfM1ME3v2+ecHXvBtP/SKMrGuP0/LtfyP57rCrZltrZguNE32GN/gAwAAAAAAI
f4OvuwxbzsAAAAAAEBn8ouoVsX/Bp+JKdlkiZYVfHHct/bM7/LbeT47z7QQDBV82d/nlz7vliXjr4d7zK3LfqRKU
7wtSpWmCz7ju9dR9PYvkpn+w38TRX/+c/pEu/H/jrnQt8kaTPPll3m5d1AF3/GZp+nSJf8biKl0/YtFWqDAk8Wg1
B1YwbdwtepPhwMAAAAAALSnUBlFWp8qFHwVHj56fGAF39a97XSpAAAAAAAA9REqo0jrU4WCr4Gl68v7XvANjY1Fb
CKtTxUKPgAAAAAAAOxaqIwirU8VCj4AAAAAAACgxij4AAAAAAAAgBqj4AMAAAAAAAAqDEKPgAAA
AAAKDG9rfgGzsSvfXWW9GRsfR3Z+t4dCh+/K23jkSj6UPadnT8L+Ln/+J4/JMxGh2x0x+Kjm/ZBwLS1/w4nmPZc
290RbKb5vQmOrmprnidujkPF3ylN93mFVuFW/uUKTbMtPUu+My250WE26aS+EWH2YfpY34JlKSk+LGvOxIdafBe2
Vqa96i6KEreC1106iJDKptfI828brfz/p58L/s/f13CzD9fBz/+dPnvjlw/f95tzu7/dL/Jn2Pu8yO8P8x2yvfE/
WGes9trPpPstsh90OyxKJKWdfo9F/NM38/j8eeZ2ef2s5GCDwAAAADQQvtW8Pk32OGiISArGERxkRUKyc23KkiyG
```

AAAWmP//oju2Gh0PC3Lkhvn5CZalgk6poCQ0yQ3wn5RWIwuIPxvxiSvj6cZOx4dSW+0s8f8G233LcMfH7FFiVmnZ
S+GZDGSzSdY8KWxy3Zlkl/aycflz+l8C+vdzoqF2aETo9VlRbrfRn8cb7Mqpgzxfsr9a0sNuR/Ncv392gzzunD5l
9f2hlvMUxYlYt2SdXXzcjfl6e+1Kvliu/gGmeVel5YU2T7OSovjYv9sR9vZPM1+8vZZWdS+bDBtlvD6t60D3P9Je
PxYRSHQdlSBsYPixi8+XNGU7dt4v5j/FvaF2W9un8mfPYXXmmnz9zusmWnazAHuf/vNv3QZrngqlE5pkoLQvTfpa
aasyYs3s+1m/+XlRcq9zhUTcv/Z9zEpfZLfZSkkSz35s0eVU2a6pAgx65G8n8ljZtmurDLseqrltbkD3/9pWSf2W
Mm3Q+9nVUjIEsIVH/YJb9/afeO9n9m+lMWI24clsfNP39v09WY5wfJEFjBeWdLWDnz/J/NPpqvg7UN1bqh9q5dVk
7OCKS1ATPGhpq3cb26fyZ89Wfni8QoXGVXM1MTB7/+ygk+UVGncNLogTH8X0wUTeu/aTPb35xni8+DImHsvvO2w0
wAAAAAAANQYBR8AAAAAAABQY2/dunUrKstbh+MJSMsT2teEEEIIIYQQQgghhOwmb3355ZdRWULlFNl7QvuaEEIII
CGEEEIIqXEo+AghhBBCCCGEEEIIqXEo+AghhBBCCCGEEEIIqXEo+AghhBBCCCGEEEIIqXEo+AghhBBCCCGEEEIIq
7RvCCGEEEIIIYQQQuocCr4OTGjfm3Lrh46CjxBCCCGEEEIIIZ0YCr4OTGjfU/BR8BFCCCGEEEIIIaQzQ8HXgQnte
/zkJzah5wjpmILvL1ej6PM48rHL30TRb8TvLmbaZmSvPRXP+0H683h4Ocb1+DnzfIh6TWAerUxo3+9Pwfc0ujrQ0
P9ldO6m9/jNc9Evj0/ox2Ti5bjlm/Uu2x8HmY4p+EzJdv1B2uSUiaf5y3jaQhkoC7w0qhz0Cj5fw7LOL/Q6quAzK
8VbWrC516iCzxZzxXX85flz3rbGMfsmW49j8WvEeqno12WloCoU01QVg/uYzij4TOnmFXQmVd/ga7bg+41XGpoSM
Zibv3w+K0ztest95+/Lg01HFHymjHOF2fW0zAm5fCqZZk9/RLdiGfaP6MbPm/XxUfDFk9vnr8avGYiuPk3LPe9be
5V1WcEnvkUnv8FnfpYFnyrYKr7BV/Z8VUwBqL452JHf4JN//144AwOhgm8yLwN74+cD85iMp9mPko+CjxBCCCGEE
Mya/Qaf+7adTTMFXzqP34iffXZ687woGYP5QX+DT/8RXcdMa6fZRxR8hBBCCCGEEELqEvctvqdPnxaee/TokX3OT
a7gM2We/5wr+P79v//3hecKyQo+72ebpKTSZVTyLTVZVvmR05vSKzSNjC7b4mUeT/6IbP6tN7Meob/7bpcFny0po
zamHNvHb/Cl62F+zl4rSjT1eFYC5gWfnVf63+qCT29D2XrKx2Vhp+Ktm/s7+LJ1t9P9Mjp3Pl5++prQsvYzHfkNR
dfygq8wvSn40nVMnncFoVvP4jcWs8fNP85h5+PKwfS1gfLRzDtbf/tcMt/kH/jQ8z+odGzBZ363fx+f/y2+eNrCY
T/7tpubhynR4t+95blluDItmS7+2f0x3qqCz22P3dak3DO/J0VePr2bzq1/8lxS1OXLdK/P90m+Tvn0dn+Ox9uWz
haatjyuiMoLryyu9Dt+OZ2myYjCyxVnwXhlnI5ZL/d377l1NEnWc+J4+g06uzxT8CV/B56ZxszTlXO2YEuLv0Kx5
gghhBBCOjMUfB2Y0L6n4KPgI4QQQgghhBBCSGeGgq8DE9r3FHwUfIQQQgghhBBCCOnMUPB1YEL7noKPgo8QQggh
gghhBBCCCGEkBqHgo8QQgghhBBCCCGEkBqHgo8QQgghhBBCCCGEkBqHgo8QQgghhBBCCCGEkBrnQAu+L774ghBC
psHAAAAAAAAdDbThZlO7CC+xXcgBZ9pLLe3t9PNAwAAAAAAADqb6cJMJ9YRBZ/5KqL588f3799PNw8AAAAAAADob
Ao+DwUfAAAAAAAA6oSCz1OHgm/hw59EP3m3J3qQ/m7NfxD95Cc/id7tVY8mtnuid+PnPpg3vzyIet6NXx//nufdo
IM+HHejmd/kJh8U+Q1BOi91UhULvsLJWPnBtNMPL9SDu8ik+dB8zIr3WhR8yQdxPl3ph7n7sHbFoYguScLHlJtvs
DEKA1P3YfKuGvyGkgy4k5OqWPCVfLh4g/aqcONbX8lNX3pTld289eQXKvtYftPlbqiKN1nyBkr+nB4f7psnH5YcV
623v6fX4mDi/YgwCj5Puxd8OzlR7IeD+6AwHyb2JEk/GOzjbgCcfODIQW7TBZ9Hrp8egKOukvdff2Ni4cP49/n4W
txS16/0OuWeq9s4Mx9/m2u7GYck2/WBuY9Uv9upk+t/Ol5xr03+Z3/Z9Ro+Cj5P23+DLz4Z7EGeDjrtye4PQO1gX
wbR3jSlyqZLH0+W6w/Q0zGHOzetwGNi3u4cLxtfuH2QnIvFMY17nZ3OFYbmueB+5ZwGALRAdi0sK6rSsac/FnXTu
KbJ/rikPS7FcsT02Qe+e1yuZ/AYRV1kn1vxe2l+loVBdvF3N+bi/Q7GPJ8OirLjgWMONWGPr/QY3ZdrfnputKTgo
3ZCyIGtvFlwJ4o+2czAVn4gmd+T18obAPkBlP5s10n+jPpK38f0QzY5Hrz3Whxz5vjIit7CDZo8blLyBiy9uJmf3
uX3UvKaLuVjUPdM8brkptHXzPaVrL+Ju+anxNhbMuN//6/YMYJjE5Si4PPUsuCzJ46f/MQITe/+jj03fXaCZSdc8
2w3OV2+Tua3Jgq+dL6h+NthHlPnuidZtrxW5dta2HZv3wAAsBvquqfGnTl5fUq463bxuvRBja5L2Zik4lpuo675e
uwxZ6/V6WesPdbya+uervleiZUc3/Jz2j+nKnjzctSNTfa7d71w62Ppx5J1ctOn0mUl06TrWJhPTJ2X3rao9Q2sE
dNnx2ToWBPToJ7sexv6nHE39nGyi7s6htJjzD1m5uF+9o4hGY45tCt53Y5/a+k1P1zweQkd70Hl139/vvl5EijWA
mhA7qSvKwzGUWfZBUrEHjvyxkp8eLuLlTwegzHHkCxGsg/8+Gd54+UeM+T0ijhmTTgGa80eO+lnjDr+Qu+rf0wEj
AAAaoyCDwAAAAAAAKgxCj4AAAAAAAACgxij4AAAAAAAAAgBqj4AMAAAAAAAABqjIIPAAAAAAAAAqDEKPgAAAAAAAKDGh
HwAAAAAAABAjVHwAQAAAAAAAADVGwQcAAAAAAAADUGAUfAAAAAAAAAUGMUfAAAAAAAAAECNUfABAAAAAAAAANUbBBwAAA
Nw1lb1aW11VubO6orJ5/5GK782rFyqrd+6q+Brtn/WbKyov/xS/RiR6863KxPC0yqs3kcrGxpbKtcUZlXuPnqn4N
FaWZlW2NrdUxoYGVZ5981plr9ZXr+ls3FOJTzqVxaUbKv7xej/exzLRn16qTI6NpktGM7Y2N1SuLU6pjIxMqJjPM
kYo+Dy1K/imj0Zvv300mkt/3au5d96ODp9ufMNQZq+vR3sy7+vbb7t4x9uDruiwe+6dwJFon/deY4/b9DU/6ooqj
Pzz2z+fVm/dUbl2Y11lZfm6iiz3DrLgM+kkdkzhjbXkOKd0/ODGKoVxmhvvpGOz4Nik8fjyIJgSzBR1R0xhZwo8U
b2IG2Yb8tfY9SwpRDri/OOYy3TuMVe+j9SyvHVXz8ltrJhfgX2d3B/6tXYZ8vhJ97e/naF9chAo+PaGgk8XAD4KF
97LC0p+LKkbXjtdfLGZ9o63Jo9Hq2L+9uemLlzmwikuenL9zc9NzSMpAMy05oKZHd/puXh02ltGTE6n11WXCQnz-
5xSGRseV/nk9AmVj8+cUznxH36mMjB7XeWjj8+rTI4Pq0xNjahcPNursvHgscr43KzKYN9llfOXulX8/dN14aLKH
WXjocrjjasq50/3qHxw7JRK16UzKldGJlT892NpaUrl//l/f6Vy7+mmyvzENZX+3iEVv+Dq6RtWGe7rVVlcvqUyN
RAfAzJDA70q3RfPq3zwh3dVznzyR5Vr29+o1J0dT8TjhLkG48fSsVQ6fu2Kn5cFoJn+6GkxLrNjGD0mdK9t+RhlH
esuzkH3vP94mbJiIbxuHrP9gde2M465DjzmKvaRek/l4/EczHUo3++irKuan1T2uGSXH++XwPZKZftkzyj4KPhEH
vCYzY9BO4+S46l8/vKYdo+X3LT76yS2wV92M2VH+BivKlu8dY2p5Zj1Sbex8fKTQUBZuVR1/iXbmrxWpuyzoG1wz
38YZDdQ+XTyZq1qfuqDRHyAVN3soebS91zd5LsLT/pr4fg05DGXksfW4XeOps/rG+rspr3R/K38Rt8WFuk8Cse8Y

```
MVMiWYSKuJMdlv2mfVoppQr+xYffwefDgVfKwRvYpLfKz883HTmZK/4gJE3a40+jBw5XVM3e6gfe5Eou1nOj8fgE
Nnc3FSZmltSWZiOb8RF5uYXVbbu3FHxC4Bbaysql/sHVSaG+lVWrt9QuTQwrrK0OK2yfG1Bpbf7osrE7LLKubPdK
//4+JDKxPS4yvCVz1QG+gZUenr7VRZvrKq8iV6rLE6NqcyOT6lcOPOZyuDYgEqj92Pr3m2V4dFZldfRNypry1teV
+eQzFr8Hsic7zqjcvPWmsrg6KTK0kL8uSwyNjWrsrQUPy4yeHlQ5cWfIpVOocdkAWVjFjH2MPNI/oqXdIwix6v2Z
Sr2p143yZt/yTw65vzjmFM685gr30fF/ZVPp0o19R5U7XOPfZ3cFv0+FIu78PtUnK41KPgo+GQo+Cj4KPgo+Cj46
mcRx5E4HoMXJ3kBSql5hl4jlc1fPl5SXuSSksBO69+Qy/k0cbMeLjSKZYs/ndzmspJdXeD9/aa2NzyP8Lo1R7/WE
9TkN/hChVyV0fi1av+m83YJFYaq4IuXdchMGz8m50PBR8EHAABQiYKPgk+Ggo+Cj4KPgu+HVvChM5gibvRBXo6Zg
ABQYxR8AAAAAAAAQI1R8AEAAAAAAAA1RsEHAAAAAAAA1BgFHwAAAAAAAFBjFHwAAAAAAABAjVHwAQAAAAAADVGw
AMAAAAAAABqjIIPAAAAAAAAqDEKPgAAAAAAAKDGKPgAAAAAAACAGqPgAwAAAAAAGqMgg8AAAAAAACoMQo+AAAAA
EJlZGRC5dWbSMX36usnKjdurKuMDQ2qPPvmtcra6orK03trKk+eP1eZm5lSmb66qlLw5pXKxMiwyt27WyoTC9dUt
RtrKp8/21YZnllS8Y+H2cXrKr5XXz1SGZtfVmnkxZO7Kk9evFbx+dtnPgJkXsTbJONv3934M0zGP94XZqdUtrfuc
7HT7MF2vMy34mXtSLxMua6SWafjxcve946Cz9P+Bd9cdPTtw1FXGx5M6Fxz77wdvf22y9H4KBQedEWH3XPvqGcS9
6t8/bXO62+/VPE/f5AIjQXlGKd8jOeNZTPbUdePzPPFTsCNbdw89Xg6ya7HbLtgSjBT1B0xhZ0p8EzBlpZmpnAz/
ujnQvutuH41xTGX6exjrvy9tPvOraPaxqp1anBsCMl7I98/+dr2RsFHwUfBl4eCj4JPhoKPgq/dlI4FS8cu+ZhOj
HDTJc+bdfIu0tmy5HSamk5tv56feh+aXr+UmW9wO9sUx1xnHnMV+9ruv2zd5b6uWKe9vnel78UB8wqcf/fXv1Lpv
UufdatMTQ2rnD/5scrHJy6onOzuVXn98iuVsalxlZmpMZXjf/trlWMfnVH5+sUXKgPDYyqnT/xa5f1fvq9yvm9Ap
eRHx1Q+OHlO5Vfv/UHl4oVTKv76HD/xiYq/fU+/faFyeWBIp/uUSvf5bhV/fp+/fKkyOBYfUyIXus6qfHr8mIq/v
1Th1yQzfzO/stLORZaK2R8HTtdVouDLQ8HXEvrmRp209kZH3PgA+0DdNPs3xKoASPkXJ/819vmS47aZ+RtV8xB04
TQ7LfkM+cd+C0SRV7XeZhoKvjwUfC0hT2T/A+FgG3n8AMmb6VjhwhO6cQ7ekOfOHqZ2HmKfU1PzT475hwRCz8yuUH
6hv1e1Ss9/gUwLrQsGXh4KvJYoFn3/Slt1oAXvilSSWf6McuqH2y5aYvOk//M7R9PnkYqWO44bzT18jLprqQuZdT
XZV8MVMiWYSKuJMdlv2mfVoppQr+xYffwefDgVfS/gFX/FmC2g5e0McONa8m+3gBcq/Iff5N+VS5fyTi95OLlBNX
bGJSZWhkWmVlcVRldGpFZWV1VWV+5vrKpd6+1S+ebSmshWPIWXG5ldUBkcnVSbHh1Qmpid14ptsmdVbGyqDY1MqH
pbXVMb6L6rcWFlR6RkYVtm4c0tlOz6GZJ4/f6zSOzypcvvObZU/x1spM9Z/SeXO5pZKb/+Yyo1rsyrLN1ZVLnd3c
8bs/0TKMzEzoeJ/XtxY3VTpFA3HeiXjq/KxW/J4Mo4xY5qjUZdYhhyX5eOnXDNjz1axxZ77b1rsNVvwya0+6G/wu
94PCj4KPgq+LBR89eSPBf3f87GMGYuI8W/peDgZ48jxjv1TIel4Uo7L9DjJqB4ftVS83u5frrXfnEsLPVmaGarg8
QcLhlmWWLZXCu0njjmtc4+5WOl7qa9t+rmKdSqdX5F/nKlt8PeDOf7F/Pa0zQ1Q8FHwUfCJ94OCj4KPgi8LBV89M
KlmCj5T2rmSzRV/heJOlHG7LtPcOrh4y7CFYclztuCL11MyxZ6d1iv82gUFn4eCDwAAAAAAHVCwecxJd1eAgAAA
AAAAKgxCj4AAAAAACgxij4AAAAAAAgBqj4AMAAAAAAABqjIIPAAAAAAAAqLGOLfju3btHCCGEEEIIIYQQQkjHh
VHwAQAAAAAADVGwQcAAAAAADUGAUfAAAAAAAAUGMUfJ7e3t49BQAAAAAADhIFHyevZR0FHwAAAAAAA4aBR8H
rWKz98+8xEg8yLeJhl/++7Gn2EyN29vqSzMTqlsb91R8ee3dWdNZeXmusr8+KjK/ftbKv7+unt/W6XoO5WtjW2Vt
Fybm1B5/MVLlbub8XqK3L61orP1QMX31aMHKgvXllSW4nWW8d+P+w+fqvjv7/rGpspiPA8Z37ef31fx38+XXz1V8
fXoLOZ4ePttF+/4s8dk+tw7gSMzdMzK1/yoK6o80srmLx9vcE6Urb9+vPh8iD435qKjhdfHCe2HjHnN4ahLXBAq9
SPzfHEM58Yvbp5yOS4HNV4xTAlmirojprAzBZ4p2NLSzBRu5r+q4EsLuCPx69xauunsPNLHdizeT4dEeViadPlur
uA6yPk1PN5ibn+I80OdL2r9POa50Hml3h+9b5unt1++J2r9vG0sX/fy+RXY18n9pl+bDI7lPkm20X+vKj939hEFH
XzHx+S0nqbyht1XNX/Jm66SXH/JPF62HvGj5kJqLpDmQll2XujnzAVbXEDtOpoLuPe4r2L/SnZfuAu2t02V5YZh5
heA3ee7VObm5lVevHqjMj07ozI5PqiyMP6ZyplP/qhysrtP5XcfdakszlxRGZtcVZmbmVbZWltW6e7tV9m8qjO3u
6wyORgfhyL++/nk2abKfHxMyTy8e0Ol71yvygfHTql0XTqjcmVkRKXv4lmVK719KqfOnFG5eOGcyuSlfpWFpVmVT
VUZGOpX+fDY36r0zd5W8T+vP4+PeZn5ySWVrg9P6Hx8UsX//JmKPxNlTp27pOJ/XkzM31GpOzueiMcJcw3GU6XjH
7/YlZ4bpRdhn1dEeJrbFr0Pk3m69fefC0gHANXLaB8HeswJ4fe6+v0rkJ+/TR7Pkr8Ood/LtsXut/g5fUz5x3r58
m8zDQVfHgq+Vqg4KeWHhzyxCx8O2U1m+Y2qfD2Q8W6Og8eWf3wGj9nkwtToW6KN559cFMvLB0/ZzX12TjRWdm40+
47seCEwVisfRzUa0ydjFPNcPk40jyXjHjl2tONEN5apnOf+MCXYXv6IrqO+VbdLzX6DTwmsCwVfHgq+VlA3J94N
ff46xD6PfRem8fdfsu31/5m1yvfp8nvxe30lU+n5x8QOt4q3iO9Tfm5XbnuDd7zTNnjDYT2c8Pt3i0KPhUKPgo+
oHlVa5fCbHfChdvO798vxWY1za9vt+T7+uYE8Kfg80cL8mArPDaZtbd46+DfU32Xpcd98ny8+IxTbosPc9k2uLg
qrK/c11lUu9fSrfPFpT2YrHkDJj8ysqg6OTKpPjQyoT05M68U22zOqtDZXBsSmVlesLKlOz0yrDVy6p9PWMqPSa
5HI4yePVK7E2yTj77830WuV8cHLKis3bqkMDI2rLMbntMyNa/Mqvd1nVYanF1VmF6+prK0uqfjrsxavg4y/fRt3
DFZMn7pEsuQYyU5hnIark8L2WLP/Tct9pot+NSYK57uIL/B54pJ+Q2+rKyk4MtCwdcK4gZR31wkNzrBmyDvZtae
eeX7DM3Xfm6l8+vwL5O7rfktW56f3/5zzvF6VqDgo+Cj4JPvB8UfBR8IhR8eblHwVcf/ripfBzljb1Kx3nJuESOJ
qWOxwYFRdn85eNx3EXN13D9xXnTDHnRdEKPJednaNv045Xr5+07Pa3ep1XPNUcOFgyznrvfT3vxvR9zqZ29rynvu
f5sBfcNPcMUdabss4WYKM2M0Df4XCnnCjX5Db5QIVdlNH6tGqul83YJFYaq4IuXdchMGz8m50PBR8EHAABQiYKPg
sEHAAAAAAAA1BgFHwAAAAAAFBjFHwAAAAAABAjVHwAQAAAAAADVGwQcAAAAAADUGAUfAAAAAAAAUGMUfAAAA
AAAAKDGKPgAAAAAACAGqPgAwAAAAAAGqMgg8AAAAAACoMQo+AAAAAAAAoMYo+AAAAAAAAIAao+ADAAAAAAAAa
```


xpUefbNa5W11RWVp/fWVJ48f64yNzOlMn11VaXgzSuViZFhlbt3t1QmFq6pbN2+qbJ2Z0slevmFysryTRV/fnfW
uJ79dUjlbH5ZZWDZD4OXsTbJONv393NWyrmeJdZmJ1S2d66o+LPb+vOmsrKzXWV+fFRlfv3t1T8/XX3/rZK0XcqW
R4FHwWfQcHX3gXfvY11lRvrGyrLSzMqk9M6/uf1i/iYl/GPfwq+ndlrwdcJjv9VFL11OMnoQv7zoYvpBLHR30bRl
+h0+qBltsHNo2Jb1LLy7dDvTfH5uqr1MWc/l9NlxVHvd6N194TWofxYCrPTe9uc79/mzh87fba+21HXj9zr4zS5F
wq+tIA7Er/OraWbzs4jfWzH4v10SJSHpUmX75YZKvjculHwUfDtDwo+7DdzjIkiQt/oJzfUyQ14crFRF43sIiWPU
p8k75u/jWpdDhAFHwVfqws+V+5R8FHw+Q664PO3z30EmP+aUPDpzy8KvvZWVvDZcUjpuHYH4/14Wj02SV5/uGo8I
cVWXFDdz3pZcjp//ZPn7EXe27dK2XOl2yCY1zaapt3U7ZhTxMCtat2V8mOu/FjSx1zCPH84vk7o5ejpytcvYZ4/C
clVlbmZaZWttWWV7t5+lc2rOnOLiyrHPjivcqFvWGV8qFfl1CcfqnSdOa3y3nsfq/zxw09VLn3WrTI1Naxy/uTHH
qW3T+XUmTMq57svqlz+5JzK5PycSv/Fz1TuPX2ssnrnjsrf/c3PVM6cPqtyvmdQZXhU59r0kMrJj46pfHDynMqv3
eJ//kzFn4kyp85dUvE/L66u3FOpu54rfXacMKfGbkXF8VsqOC4zkrHM0dPe+Codb3WJ+RXHNOlr03HVfttNweeKM
8jP5WbW3RM6dsLHUpE7xvXx4h8n1cecW17Vcqqes0LHnP8e+fvJPScfr1r3qvlJZY9LdvnxNvnr7Amfhy0ivqFFw
Xmxyu2gbPHXw65r/lp7ASyspxZeh0ThApqR518inzbdD+51dh2r9kkyCLDf3vXP4ex8D7PvVfpambb/LKjzMee9S
x/RddS36nap2W/wKYF1oeDLQ8HXCsGbmOT38M2UeM58sDR9cwN47LHjXRTM8SgvSv7xafjHnP09VNjoG2q/cJAXd
IHQ9uv92kIUfCoUfBR8FHwUfBR8FHx1lt2HlygfTxTH+wkxHjJjMTNvMQ4yYxY3riuOX8rGfftjVwVfzJRoJqEiz
phHumy7HhXnk2LmVdhObz/a99vMr7g+Zn3dcsvXvWx+vqp9ZqY3zwfeXzM/bxuaef92482rr1WG+6+ozF1dU1le1
ZXhK5dU+npGVHpNyStyua9HZX3zvsrE0CWVqXgdZM5296vcXFtWmVmYVenvuagyeP6yysziVZX+4VGVwvvx/JWK/
bk+qzIZb7PM3XgdZK7E75nMxp01lfGZOZXFyXGVa1eXVfz5jQ30qSybf7hCZGFiWOXa8k2VgaFxlcnJMZXxmXmdc
ZeFgq8l9IluT1r3geDdtNrnspuP5HXmBq1wIwdUqSxD5A12yUXIKzX8+TVbFOgbe8e7qa/ir4e6YZfbUc5fh9BFM
TDezyTjEjmelPf4Zn7l4yE9XtpX8Xq7f7nWfnMuLfRkaWaogs8r21xB6Ao+U6rtuOhL5+teZ/4OP8ks3xWLTlXB5
vvkkr1fjdbdE1qHsmMpvL7p9N7xYaZN5tHc+2ynT9c3f63Iro4/cQ6p1+eDVRtvPwXX3SqbX4A6HuJ4yygcg4H3F
EuKPQo+Cj77flDwUfCJUPDl5R4FH9qdKeJGH+TlmCn4TNlmSjVT8JnSzpVsrvgrFHeijNt1mebWwcVbhi0MS56zE
fABAAAAAAAANUbBBwAAAAAANQYBR8AAAAAAABQYxR8AAAAAAAAQI1R8AEAAAAAAAA1RsEHAAAAAAAA1BgFHwAAA
AAAKgxCj4AAAAAAACgxij4AAAAAAAAgBqj4AMAAAAAAABqjIIPAAAAAAAAqDEKPgAAAAAAAKDGKPgAAAAAAACAC
MbGJlRevYlU9tvqyorK0uKUyr2N2ypj88sqm7dWVG5v3VNZnBhTefj8pUrr6TPOX/69u1sq/vZsrK+qNDofvv38g
+ObOJlGXj5/pjI6MauyuXlLZe3OXZWby0sqW9tPVD5/vK0yPb+ksr56LV2T1jLlHgXf3q0szar476+/fr6VtVUVf
UWNh/1Ac+beSY8rm6PxkSY86IoOu+feUc8k7PMVr2l0rO5m/p6q9dfPvR0dPl195pjp1TRy/Zp4fXKeHo661AWhu
2BKMFPUHTGFnSnwTMGWlmamcDP/VQVfWsAdiV/nRjhuOjuP9LEdi++7DonysDTp8t0yQwWfWzcKPgq+fbYddf0ov
dLOfsgFn4ks9yj49o6Cj4KPgo+Cz6Hg25tmCz6tYrxixpPxWK7wP5X9xwNjweqx7sHZVcEnSjX7rTtX9rnnd8GUc
28HiPH2z5n9oAaKzW5HKnTMVexru9xseWZZbn2bPV/0/DQ5P8HMKzj99+jlF9G7f/e+Sv/lPpWLlwdVes9+ojMyr
v/3736qcv5ij8qv3juh8tWTByqzS9dV/PV7uHVT5XR3r4q/vX1n/6jywYmTKk9evFSZnb2q4h9/o5fPqJz55I8qv
Pe/UVm5/1zlcs8VlXu3rqvc31hTOfbhSZV343Nepm94SmV4eFBlZmpM5cyx36v42zs6MqoyMDqs8rtf/kzlt7/6r
K6rRMGXh4KvFdTJKm7S/A+O0IkOtECh8JAXD/O7fwErXNRk0eDfyHt2Nf9qav3ta+Plu8hllaguW0qKA8UrKJo+c
qbT+K87Ek+z05LPkH/st0AUeVXrbaah4MtDwdcKZTc6/g1L4EQH9sw7rlRxYYRunOVxmkkuaKYYqLr47H7+JfzzC
/Ix+7M31mqTcY4pwfbyR3Qd9a26XWr2G3xKYF0o+PJQ8LVEciOV3RypG7bwTQ/QEunFQ5VB/oXJ/O5fxBocm/lNe
WxcY8aryVgne40cD4XGcW0y1tlVwRczJZpJqIgz2W3ZZ9ajmVKu7Ft8/B18OhR8LeI+KMwJnt84NvhWELAX9ngKI
q2vq0zNLaksTE+ozM0vqqxenVPpG5tXWV9bVlmcn1HpOtOtcufOHZX+gVGV19E3KmvXb6kMX7mk0t3Vp3J5aFRl8
SmV8cHLKv76Tc0tqvj7b6S/X+ez8yo7fT/WVnWmRuJ1EJmOz0mZgaFxlVvxOSYzMz+rcr7rjMr8xKRKd0+/ysTUu
/024vdAptf8TyWRieEBlZGxERV/ftPxMSXjH1/+9m5tP1Npf8k4LPw/ev37gQAxtjJjmsOnu/LXyDGP/dmfVxPzP
a3XKZkuea7xdJXvXap4DMhlfP8o+Cj4KPgo+Cj4KPhcsefiF14UfO3OjC1KxrONxpKGHK/Z/6EZj3nc7/L1oXGnr
z1ecu5626a2xTvH1Xbs6vxPPleScsQw6yT2q1cK7SeOOW13x1zVdBXHUsm22em9dd3pdgSV7uv0Ohd8ruR8MRq9d
o+Cj4KPgo+Cj4JPlnsmfuFFwYd2Zoq40Qd5OWYKPlO2mVLNFHymtHMlmyv+CsWdKON2Xaa5dXDxlmELw5LnbMEXr
AAAAAAAQI1R8AEAAAAAAAA1RsEHAAAAAAAA1BgFHwAAAAAAAFBjFHwAAAAAAABAjVHwAQAAAAAAADVGwQcAAAAA
ABqjIIPAAAAAAAAqLGDK/i+jP5/ixgmWQTwXy8AAAAASUVORK5CYII=" />

II 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

资源类型	授权策略中的资源描述方
集群相关	qcs::ccs:$region::cl

下表将介绍当前支持资源级权限的 TKE（Tencnet Kubernetes Engines，TKE） API 操作。指定资源路径的时候，

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

API 操作
DescribeClusterService
DescribeClusterServiceInfo
CreateClusterService
ModifyClusterService

DeleteClusterService
ModifyServiceDescription
DescribeServiceEvent
ResumeClusterService
PauseClusterService
RollBackClusterService
ModifyClusterServiceImage
RedeployClusterService
DescribeServiceInstance

ModifyServiceReplicas
DeleteInstances
DescribeClusterNameSpaces
CreateClusterNamespace
DeleteClusterNamespace
DescribeCluster
CreateCluster
DeleteCluster
DescribeClusterInstances

AddClusterInstances
DeleteClusterInstances
AddClusterInstancesFromExistedCvm

II 使用示例

III 配置子账号对 TKE 服务全读写或只读权限

操作场景

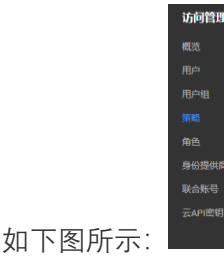
您可以通过使用访问管理（Cloud Access Management，CAM）策略让用户拥有在容器服务（Tencent Kuberne

操作步骤

配置全读写权限

804. 登录 CAM 控制台。
805. 在左侧导航栏中，单击 策略，进入策略管理页面。

806. 在策略管理页面中，单击 **QcloudCCSFullAccess** 策略行的【关联用户/组】。如下图所示：
807. 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】



808. 在策略管理页面中，单击 **QcloudCCRFullAccess** 策略行的【关联用户/组】。如下图所示：
809. 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】

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）

配置只读权限

810. 登录 CAM 控制台。
811. 在左侧导航栏中，单击 策略，进入策略管理页面。

812. 在策略管理页面中，单击 **QcloudCCSReadOnlyAccess** 策略行的【关联用户/组】。如下图所示：
813. 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】，

814. 在策略管理页面中，单击 **QcloudCCRReadOnlyAccess** 策略行的【关联用户/组】。如下图所示

815. 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】，

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）

III 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（Tencent Kuberne

操作步骤

配置对单个集群全读写权限

816. 登录 CAM 控制台。

817. 在左侧导航栏中，单击 策略，进入策略管理页面。

818. 单击【新建自定义策略】，选择 “按策略语法创建” 方式。

819. 选择 “空白模板” 类型，单击【下一步】。

820. 自定义策略名称，将 “编辑策略内容” 替换为以下内容。

```
821. {
822.     "version": "2.0",
823.     "statement": [
824.         {
825.             "action": [
826.                 "ccs:*"
827.             ],
828.             "resource": [
829.                 "qcs::ccs:sh::cluster/cls-XXXXXXX", // 替换成您想赋予权限的指定地
830.                 "qcs::cvm:sh::instance/*"
831.             ],
832.             "effect": "allow"
833.         },
834.         {
835.             "action": [
836.                 "cvm:*"
837.             ],
838.             "resource": "*",
839.             "effect": "allow"
840.         },
841.         {
```

```
            "action": [
                "vpc:*"
            ],
            "resource": "*",
            "effect": "allow"
        },
        {
            "action": [
                "clb:*"
            ],
            "resource": "*",
            "effect": "allow"
        },
        {
            "action": [
                "monitor:*",
                "cam:ListUsersForGroup",
                "cam:ListGroups",
                "cam:GetGroup",
                "cam:GetRole"
            ],
            "resource": "*",
            "effect": "allow"
        }
    ]
}
```

867. 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

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

868. 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

869. 登录 CAM 控制台。

870. 在左侧导航栏中，单击 策略，进入策略管理页面。

871. 单击【新建自定义策略】，选择 “按策略语法创建” 方式。

872. 选择 “空白模板” 类型，单击【下一步】。

873. 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
            "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": "*"
        }
```

```
    ]
}
```

921. 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

说明： 请替换成您想赋予权限的指定地域下的集群 ID。

922. 单击【创建策略】，即可完成对单个集群只读权限的配置。

I 购买容器集群

II 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：

923. 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。
924. 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

925. 建议集群内的所有资源在同一个项目
926. 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。
927. 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

II 容器服务安全组设置

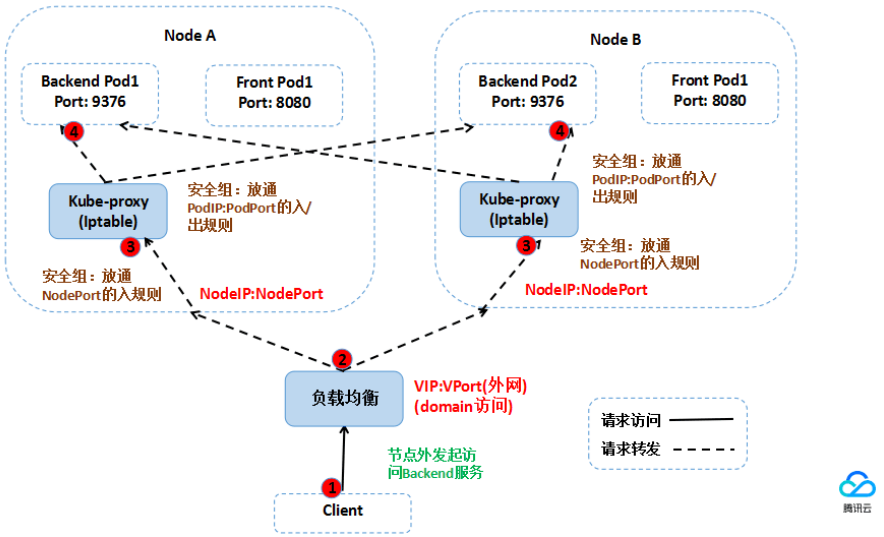
安全问题向来是一个大家非常关注的问题，TCE 将安全性作为产品设计中的最高原则，严格要求产品做到安全
最佳实践，帮助大家选择安全组策略。

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 TCE 提

使用容器服务选择安全组的原则

928. 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
929. 安全组只对外开放最小权限。
930. 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
    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



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
TCP	30000-32768	0.0.0.0/0
UDP	30000-32768	0.0.0.0/0
All	traffic ALL	10.0.0.0/8
All	traffic ALL	172.16.0.0/12
All	traffic ALL	192.168.0.0/16
TCP	22	0.0.0.0/0
All	traffic ALL	0.0.0.0/0

出站规则:

协议	端口
All	traffic ALL

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

II 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 TCE NAT 网关。下文将

公网 IP

在默认的情况下，创建集群会为集群的节点分配公网 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

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 Intern
下两个步骤：

第一步：创建 NAT 网关

933. 登录 私有网络控制台，单击左侧导航栏中的【NAT 网关】。
934. 在 “NAT 网关” 管理页面，单击【新建】。
935. 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。
936. 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。

937. 单击左侧导航栏中的【路由表】。
938. 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。
939. 在“路由策略”栏中，单击【新增路由策略】。
940. 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【
941. 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

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

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
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

! 如果用户账号开通了带宽包共享带宽功能，则 NAT 网关产生的出流量按照带宽包整体结算（不再重复收取 0.

II 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
- **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
  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 TCE 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

950. 容器 CIDR：集群内 Sevice、 Pod 等资源所在网段
951. Services 数量上限/集群：决定分配给 Sevice 的 CIDR 大小

? TCE 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

952. Pod 数量上限/Node：决定分配给每个 Node 的 CIDR 的大小

? TCE 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。

II 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

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。

I 常见问题

II 集群类

III 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

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

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或通过 QQ 群联系我们。

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，具体详情请看 云服务概览页。

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

III 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
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

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

扩缩容最大值可以设置为多少?

目前 TCE 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，

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。
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=tr
```

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。

II 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 TCE 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

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北京时间?

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北京时间差 8 小时的问题，解决方法是在 docke

```
RUN echo "Asia/shanghai" > /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
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、clo

II 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 TCE 账号（QQ 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。

II 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

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 TCE 的源服务器。即复制以下命令

```
cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe multiverse
deb http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe multi
deb http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted universe multiv
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe r
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted universe mu
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。

976. 将以下代码复制并粘贴至容器内运行:

```
977. cat << ENDOF > /etc/yum.repos.d/CentOS-Base.repo
978. [os]
979. name=Qcloud centos os - \$basearch
980. baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/os/\$basearch/
981. enabled=1
982. gpgcheck=1
983. gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
984. [updates]
985. name=Qcloud centos updates - \$basearch
986. baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/updates/\$basearch/
987. enabled=1
988. gpgcheck=1
989. gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
990. #[centosplus]
991. #name=Qcloud centosplus - \$basearch
992. #baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/centosplus/\$basearc
993. #enabled=1
994. #gpgcheck=1
995. #gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
996. #[cloud]
997. #name=Qcloud centos contrib - \$basearch
998. #baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/cloud/$basearch/oper
999. #enabled=1
1000.        #gpgcheck=1
1001.        #gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
1002.        #[cr]
1003.        #name=Qcloud centos cr - \$basearch
1004.        #baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/cr/\$basearc
1005.        #enabled=1
```

```
        #gpgcheck=1
        #gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
        [extras]
        name=Qcloud centos extras - \$basearch
        baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/extras/\$base
        enabled=1
        gpgcheck=1
        gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
        #[fasttrack]
        #name=Qcloud centos fasttrack - \basearch
        #baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/fasttrack/\$
        #enabled=1
        #gpgcheck=1
        #gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
    ENDOF
```

1020. 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

```
RUN cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe multive
deb http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted univers
deb http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted universe
#deb http://mirrors.tencentyun.com/ubuntu/ xenial-proposed main restricted univer
#deb http://mirrors.tencentyun.com/ubuntu/ xenial-backports main restricted unive
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe mul
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted uni
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted univ
#deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-proposed main restricted un
#deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-backports main restricted u
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。

1021. 执行以下命令，升级软件列表。

```
apt-get update
```

1022. 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

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

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 TCE 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

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

II 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

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

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，

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

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

如果以上解决方法未能解决您的问题，请联系客服。

II 与公有云区别

TCE 中的 TKE 与公有云 TKE 服务有哪些区别

TCE 中的 TKE 沿用了公有云版本 TKE 的技术架构，但针对私有化业务场景也稍有区别，主要有以下几点：

1023. TCE 版本没有公有云托管集群功能，因为在私有化部署场景中，托管集群亦需要用户自行维护
1024. TCE 中的 TKE 依赖 TCE 多种 IAAS 资源，例如 CVM/VPC/CM/CBS/CFS/COS/CAM 等，如果用户
1025. TCE 中的 TKE 功能可能相对公有云中的 TKE 稍有滞后，新的功能需要随 TCE 版本升级才能使用

I 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 TCE 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

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。