容器服务

用户指南

产品文档

腾讯云

文档目录

用户指南
容器服务高危操作

最近更新时间：2022-01-19 11:05:18

业务部署或运行过程中，用户可能会触发不同层面的高危操作，导致不同程度上的业务故障。为了能够更好地帮助用户预估及避免操作风险，本文将从集群、网络与负载均衡、日志、云硬盘多个维度出发，为用户展示哪些高危操作会导致怎样的后果，以及为用户提供相应的误操作解决方案。

集群

分类	高危操作	导致后果	误操作解决方案
master 及 etcd 节点	修改集群内节点安全组	可能导致 master 节点无法使用	按照官网推荐配置放通安全组
	节点到期或被销毁	该 master 节点不可用	不可恢复
	重装操作系统	master 组件被删除	不可恢复
	自行升级 master 或者 etcd 组件版本	可能导致集群无法使用	回退到原始版本
	删除或格式化节点 /etc/kubernetes 等核心目录数据	该 master 节点不可用	不可恢复
	更改节点 IP	该 master 节点不可用	改回原 IP
	自行修改核心组件（etcd、kube-apiserver、docker 等）参数	可能导致 master 节点不可用	按照官网推荐配置参数
	自行更换 master 或 etcd 证书	可能导致集群不可用	不可恢复
worker 节点	修改集群内节点安全组	可能导致节点无法使用	按照官网推荐配置放通安全组
	节点到期或被销毁	该节点不可用	不可恢复
	重装操作系统	节点组件被删除	节点移出再加入集群
	自行升级节点组件版本	可能导致节点无法使用	回退到原始版本
	更改节点 IP	节点不可用	改回原 IP
	自行修改核心组件（etcd、kube-apiserver、docker 等）参数	可能导致节点不可用	按照官网推荐配置参数
	修改操作系统配置	可能导致节点不可用	尝试还原配置项或删除节点重新购买
其他	在 CAM 中执行权限变更或修改的操作	集群部分资源如负载均衡可能无法创建成功	恢复权限

网络与负载均衡

高危操作	导致后果	误操作解决方案
修改内核参数 net.ipv4.ip_forward=0	网络不通	修改内核参数为 net.ipv4.ip_forward=1
修改内核参数 net.ipv4.tcp_tw_recycle = 1	导致 nat 异常	修改内核参数 net.ipv4.tcp_tw_recycle = 0
节点安全组配置未放通容器 CIDR 的53端口 udp	集群内 DNS 无法正常工作	按照官网推荐配置放通安全组
修改或者删除 TKE 添加的 LB 的标签	购买新的 LB	恢复 LB 的标签
通过 LB 的控制台在 TKE 管理的 LB 创建自定义的监听器	所做修改被 TKE 侧重置	通过 service 的 yaml 来自动创建监听器
通过 LB 的控制台在 TKE 管理的 LB 绑定自定义的后端 rs		禁止手动绑定后端 rs

通过 LB 的控制台修改 TKE 管理的 LB 的证书		通过 ingress 的 yaml 来自动管理证书
通过 LB 的控制台修改 TKE 管理的 LB 监听器名称		禁止修改 TKE 管理的 LB 监听器名称

日志

高危操作	导致后果	误操作解决方案	备注
删除宿主机 /tmp/ccs-log-collector/pos 目录	日志重复采集	无	Pos 里面的文件记录了文件的采集位置
删除宿主机 /tmp/ccs-log-collector/buffer 目录	日志丢失	无	Buffer 里面是待消费的日志缓存文件

云硬盘

高危操作	导致后果	误操作解决方案
控制台手动解挂 CBS	Pod 写入报 io error	删掉 node上mount 目录，重新调度 Pod
节点上 umount 磁盘挂载路径	Pod 写入本地磁盘	重新 mount 对应目录到 Pod 中
节点上直接操作 CBS 块设备	Pod 写入本地磁盘	无

云上容器应用部署 Check List

最近更新时间：2022-06-14 15:22:28

简介

业务上云安全高效、稳定高可用是每一位涉云从业者的共同诉求。这一诉求实现的前提，离不开系统可用性、数据可靠性及运维稳定性三者的完美配合。本文将从评估项目、影响说明及评估参考三个角度为您阐述云上容器应用部署的各个检查项，以便帮助您扫除上云障碍、顺利高效地完成业务迁移至容器服务（TKE）。

检查项

系统可用性

类别	评估项目	类型	影响说明	评估参考
集群	创建集群前，结合业务场景提前规划节点网络和容器网络，避免后续业务扩容受限。	网络规划	集群所在子网或容器网段较小，将可能导致集群实际支持的可用节点数少于业务所需容量。	• 网络规划 • 容器及节点网络设置
	创建集群前，提前梳理专线接入、对等连接、容器网段和子网网段等相关网段的规划，避免之后出现网段冲突，影响业务。	网络规划	简单组网场景按照页面提示配置集群相关网段，避免冲突；业务复杂组网 场景，例如对等连接、专线接入、VPN 等，网络规划不当将影响整体业务正常互访。	VPC 连接
	创建集群时，会自动新建并绑定默认安全组，支持根据业务需求设置自定义安全组规则。	部署	安全组是重要的安全隔离手段，不当的安全策略配置可能会引起安全相关的隐患及服务连通性等问题。	容器服务安全组设置
	Containerd 和 Docker 作为 TKE 当前支持的运行时组件，有不同的适用场景。创建集群时，请根据业务场景选择合适的容器运行时（Container Runtime）组件。	部署	集群创建后，如修改运行时组件及版本，只对集群内无节点池归属的增量节点生效，不会影响存量节点。	如何选择 Containerd 和 Docker
	默认情况下，Kube-proxy 使用 iptables 来实现 Service 到 Pod 之间的负载均衡。创建集群时，支持快速开启 IPVS 来承接流量并实现负载均衡。	部署	当前支持在创建集群时开启 IPVS，之后对全集群生效且将不可关闭。	集群启用 IPVS
	创建集群时，根据业务场景选择合适的集群模式：独立集群、托管集群。	部署	托管集群的 Master 和 Etcd 不属于用户资源，由腾讯云技术团队集中管理和维护，用户无法修改 Master 和 Etcd 的部署规模和服务参数。如需修改，请选用独立部署模式集群。	• 集群概述 • 集群的托管模式说明
工作负载	创建工作负载时需设置 CPU 和内存的限制范围，提高业务的健壮性。	部署	同一个节点上部署多个应用，当未设置资源上下限的应用出现应用异常资源泄露问题时，将会导致其它应用分配不到资源而异常，且其监控信息将会出现误差。	设置工作负载的资源限制
	创建工作负载时可设置容器健康检查：“容器存活检查”和“容器就绪检查”。	可靠性	容器健康检查未配置，会导致用户业务出现异常时 Pod 无法感知，从而导致不会自动重启恢复业务，最终将会出现 Pod 状态正常，但 Pod 中的业务异常的现象。	服务健康检查设置
	创建服务时需要根据实际访问需求选择合适的访问方式，目前支持以下四种：提供公网访问、仅在集群内访问、VPC 内网访问及主机端口访问。	部署	选择不当的访问方式，可能造成服务内外部访问逻辑混乱和资源浪费。	Service 管理
	工作负载创建时，避免单 Pod 副本数设置，请根据自身业务合理设置节点调度策略。	可靠性	如设置单 Pod 副本数，当节点异常或实例异常会导致服务异常。为确保您的 Pod 能够调度成功，请确保您在设置调度规则后，节点有空余的资源用于容器的调度。	• 调整 Pod 数量 • 设置工作负载的调度规则

数据可靠性

类别	评估项目	类型	影响说明	评估参考
容器数据持久化	应用 Pod 数据存储，根据实际需求选择合适的数据卷类型。	可靠性	节点异常无法恢复时，存在本地磁盘中的数据无法恢复，而云存储此时可以提供极高的数据可靠性。	Volume 管理

运维稳定性

类别	评估项目	类型	影响说明	评估参考
工程	CVM、VPC、子网及 CBS 等资源配额是否满足客户需求。	部署	配额不足将会导致创建资源失败，对于配置了自动扩容的用户尤其需要保障所使用的云服务配额充足。	• 购买集群配额限制 • 配额限制
	集群的节点上不建议用户随意修改内核参数、系统配置、集群核心组件版本、安全组及 LB 相关参数等。	部署	可能会导致 TKE 集群功能异常或安装在节点上的 Kubernetes 组件异常，节点状态变成不可用，无法部署应用到此节点。	容器服务高危操作
主动运维	容器服务提供多维度的监控和告警功能，同时结合云监控提供的基础资源监控，能保证更细的指标覆盖。配置监控告警，以便于异常时及时收到告警和故障定位。	监控	未配置监控告警，将无法建立容器集群性能的正常标准，在出现异常时无法及时收到告警，需要人工巡检环境。	• 设置告警 • 查看监控数据 • 监控及告警指标列表

开源组件

最近更新时间：2022-01-19 11:14:21

tencentcloud-cloud-controller-manager

tencentcloud-cloud-controller-manager 是腾讯云容器服务的 Cloud Controller Manager 的实现。使用该组件，可以在通过腾讯云云服务器自建的 Kubenrentes 集群上实现以下功能：

- nodecontroller：更新 Kubernetes node 相关的 addresses 信息。
- routecontroller：负责创建私有网络内 pod 网段内的路由。
- servicecontroller：当集群中创建了类型为负载均衡的 service 的时候，创建相应的负载均衡。

更多安装使用说明，可查看 GitHub tencentcloud-cloud-controller-manager。

kubernetes-csi-tencentcloud

kubernetes-csi-tencentcloud 是腾讯云云硬盘服务的一个满足 CSI 标准实现的插件。使用该组件，可以在通过腾讯云云服务器自建的 Kubenrentes 集群使用云硬盘。
该插件适用与自建 Kubernetes 集群的时候使用云硬盘的插件，与容器服务集群自带的 provisioner cloud.tencent.com/qcloud-cbs 不相同。

更多安装使用说明，可查看 GitHub kubernetes-csi-tencentcloud。

集群管理

集群概述

最近更新时间：2022-04-22 17:20:03

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、负载均衡器等腾讯云资源。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 容器集群支持下述类型：

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

集群类型详情可参见 集群模式说明。

集群生命周期

关于 TKE 集群的生命周期，请参见 集群生命周期。

集群相关操作

- 创建集群
- 更改集群操作系统
- 集群扩缩容
- 连接集群
- 升级集群
- 集群启用 IPVS
- 集群启用 GPU 调度
- 选择容器网络模式
- 删除集群
- 自定义 Kubernetes 组件启动参数

集群的托管模式说明

最近更新时间：2022-04-08 16:59:10

Master 托管模式

简介

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

Master 托管模式注意事项

- 针对不同规格的托管集群，会收取相应的集群管理费用，以及用户实际使用的云资源（云服务器、持久化存储、负载均衡等）费用。关于收费模式和具体价格，请参阅 容器服务计费概述。
- Master、Etcd 不属于用户资源，您在该模式下无法自主修改 Master 和 Etcd 的部署规模和服务参数。如果您有修改的需求，请使用 Master 独立部署模式。
- 该模式下，即使您删除集群的全部工作节点，集群仍会不断尝试运行您未删除的工作负载和服务，导致在此过程中可能会产生费用。如果您决定终止集群服务和费用产生，请直接删除该集群。

Master 独立部署模式

简介

腾讯云 TKE 也为您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。

Master 独立部署模式注意事项

- 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	建议 Master 节点配置	建议节点数量
约100个节点	8核16GB SSD 系统盘	3台以上
约500个节点	16核32GB SSD 系统盘	3台以上
1000个节点以上	咨询我们	3台以上

购置限制说明

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

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

注意事项

为了保证集群的稳定性，以及发生异常后的恢复效率，建议如下：

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

- 请不要修改任何节点操作系统的 kernel、nfs-utils 等相关组件。

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

如果您对以上建议有疑问，请 联系我们。

TKE-Optimized 系列镜像说明

最近更新时间：2022-04-13 16:38:44

TencentOS-kernel 由腾讯云团队维护定制内核。Tencent Linux 是腾讯云包含该内核版本的公共镜像，容器服务 TKE 目前已经支持该镜像并作为缺省选项。

在 Tencent Linux 公共镜像上线之前，为了提升镜像稳定性，并提供更多特性，容器服务 TKE 团队制作并维护 TKE-Optimized 系列镜像。目前控制台已不支持新建集群选择 TKE-Optimized 镜像。

⚠ 注意：

- 仍在使用 TKE-Optimized 镜像的集群不受影响，可继续使用。建议您切换至到 Tencent Linux 2.4，新增节点使用 Tencent Linux 2.4，存量节点不受影响可继续使用。
- Centos7.6 TKE Optimized 镜像与使用 Tencent Linux 2.4镜像完全兼容。
- Ubuntu 18.04 TKE Optimized 镜像用户空间工具与 Tencent Linux 并不完全兼容，已对节点做配置变更的脚本需您自行适配新版本。

集群生命周期

最近更新时间：2022-04-08 10:18:14

集群生命周期状态说明

状态	说明
创建中	集群正在创建，正在申请云资源。
规模调整中	集群的节点数量变更，添加节点或销毁节点中。
运行中	集群正常运行。
升级中	升级集群中。
删除中	集群在删除中。
异常	集群中存在异常，如节点网络不可达等。
隔离中	因为欠费超过24小时导致托管集群进入隔离中状态，停止扣除集群管理费用。

注意：

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负载均衡（CLB）、云硬盘（CBS）盘等 IaaS 资源，现在不再需要使用 CLB 和 CBS，请在 容器服务控制台 中删除对应的 Service 和 PersistentVolumeClaim 对象。如果您只在 CLB 控制台中删除 CLB 或者在 CBS 控制台中删除 CBS，容器服务会重新创建新的 CLB 和 CBS， 并继续扣除相关费用。

创建集群

最近更新时间：2022-06-16 17:01:59

操作场景

腾讯云容器服务新建集群时提供 使用模板新建集群 及 自定义新建集群 两种创建方式，本文介绍如何使用上述两种方式进行集群创建，以及如何创建集群所需的私有网络、子网、安全组等资源。

前提条件

在创建集群前，您需要完成以下工作：

- 注册腾讯云账号，并完成 实名认证。
- 当您首次登录 容器服务控制台 时，需对当前账号授予腾讯云容器服务操作云服务器 CVM、负载均衡 CLB、云硬盘 CBS 等云资源的权限。详情请参见 服务授权。
- 如果要创建网络类型为私有网络的容器集群，需要在目标地域 创建一个私有网络，并且在私有网络下的目标可用区 创建一个子网。
- 如果不使用系统自动创建的默认安全组，需要在目标地域 创建一个安全组 并添加能满足您业务需求的安全组规则。
- 如果创建 Linux 实例时需要绑定 SSH 密钥对，需要在目标项目下 创建一个 SSH 密钥。
- 集群创建过程中将使用私有网络、子网、安全组等多种资源。资源所在地域具备一定的配额限制，详情请参见 购买集群配额限制。

操作步骤

使用模板新建集群

1. 登录 容器服务控制台 ，单击左侧导航栏中**集群**。
2. 在“集群管理”页面中，单击集群列表上方的**使用模板新建**。如下图所示：

3. 使用模板新建功能为您提供托管集群、独立集群、弹性集群等多种创建模板，请根据实际需求进行选择：
   - **托管集群**：创建一个 Kubernetes 托管集群，无需购买并管理集群的管理节点，只需购买其中工作节点资源即可部署业务应用。
   - **独立集群**：创建一个 Kubernetes 独立集群，同时购买并管理集群的管理及工作节点，拥有集群的所有管理和操作权限。
   - **弹性集群**：创建一个 Serverless Kubernetes 集群，无需管理集群的任何节点资源，即可快速部署业务应用。
4. 本文以使用托管集群下的“入门集群”模板为例，选择**入门集群**即可前往“创建入门集群”页面。

> 说明：
> 使用集群模板创建集群过程中，各配置项已采用默认值，可以直接单击**下一步**，也可参照 自定义新建集群 步骤进行自定义配置。

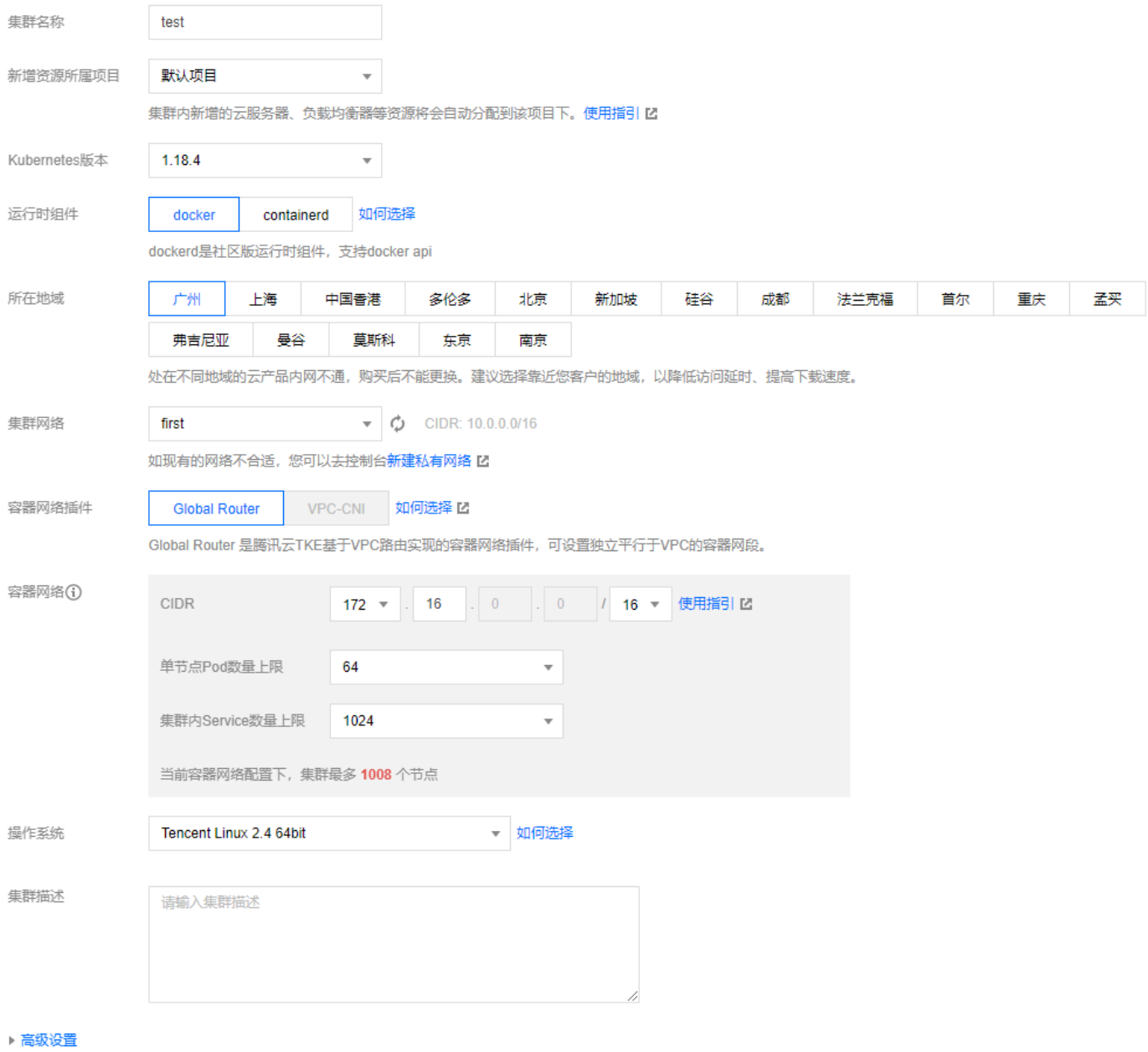
5. 单击**完成**即可创建成功。

自定义新建集群

1. 填写集群信息

1. 登录 容器服务控制台 ，单击左侧导航栏中的**集群**。
2. 在“集群管理”页面，单击集群列表上方的**新建**。

3. 在“创建集群”页面，设置集群的基本信息。如下图所示：



- **集群名称**：输入要创建的集群名称，不超过50个字符。
- **新增资源所属项目**：根据实际需求进行选择，新增的资源将会自动分配到该项目下。
- **Kubernetes版本**：提供多个 Kubernetes 版本选择，可前往 Supported Versions of the Kubernetes Documentation 查看各版本特性对比。
- **运行时组件**：提供**docker**和**containerd**两种选择。详情请参见 如何选择 Containerd 和 Docker。
- **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**：为集群内主机分配在节点网络地址范围内的 IP 地址。详情请参见 容器及节点网络设置。
- **容器网络插件**：提供**GlobalRouter**和**VPC-CNI**两种网络模式。详情请参见 如何选择容器网络模式。
- **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。详情请参见 容器及节点网络设置。
- **操作系统**：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**页面。
- **高级设置**（可选）：
  - **腾讯云标签**：为集群绑定标签后可实现资源的分类管理。详情请参见 通过标签查询资源。
  - **删除保护**：开启后可阻止通过控制台或云API误删除本集群。
  - **Kube-proxy 代理模式**：可选择 iptables 或 ipvs。ipvs 适用于将在集群中运行大规模服务的场景，开启后不能关闭。详情请参见 集群启用 IPVS。
  - **自定义参数**：指定自定义参数来配置集群。
  - **运行时版本**：选择容器运行时组件的版本。

4. 单击**下一步**。

2. 选择机型

在“选择机型”步骤中，确认计费模式、选择可用区及对应的子网、确认节点的机型。

1. 选择**节点来源**。提供**新增节点**和**已有节点**两个选项。**新增节点**

通过新增节点，即新增云服务器创建集群，详情如下：

- **集群类型**：提供**托管集群**和**独立集群**两个选项。
  - 托管集群：集群的 Master 和 Etcd 由腾讯云进行管理和维护。
  - 独立集群：集群的 Master 和 Etcd 将会部署在您购置的 CVM 上。
- **集群规格**：根据业务实际情况选择合适的集群规格，详情见 如何选择集群规格。集群规格可手动调整，或者通过自动升配能力自动调整。
- **计费模式**：提供**按量计费**和**包年包月**两种计费模式。详情请参见 计费模式。
- **Worker 配置**：当**节点来源**选择**新增节点**、**集群类型**选择**托管集群**时，该模块下所有设置项为默认项，您可根据实际需求进行更改。
  - **可用区**：可以同时选择多个可用区部署您的 Master 或 Etcd，保证集群更高的可用性。
  - **节点网络**：可以同时选择多个子网的资源部署您的 Master 或 Etcd，保证集群更高的可用性。
  - **机型**：选择大于 CPU 4核的机型，具体选择方案请参看 实例规格 和 快速入门 Linux 云服务器。
    - **系统盘**：默认为“普通云硬盘 50G”，您可以根据机型选择本地硬盘、云硬盘、SSD 云硬盘及高性能云硬盘。详情请参见 存储概述 。
    - **数据盘**：Master 和 Etcd 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。
    - **公网宽带**：勾选**分配免费公网IP**，系统将免费分配公网 IP。提供两种计费模式，详情请参见 公网计费模式。
    - **主机名**：操作系统内部的计算机名（kubectl get nodes 命令展示的 node name），该属性为集群属性。主机名有如下两种命名模式：
      - **自动命名**：节点 hostname 默认为节点内网 IP 地址。
      - **手动命名**：支持批量连续命名或指定模式串命名。长度限制2 – 60个字符，仅支持小写字母、数字、连字符 "–" 、点号 "." ，符号不能用于开头或结尾且不能连续使用，更多命名规则指引请查看 批量连续命名或指定模式串命名。

        > ⚠ 注意：
        > 由于 kubernetes node 命名限制，手动命名主机名时仅支持小写字母，例如 cvm{R:13}-big{R:2}-test 。

    - **实例名称**：控制台显示的 CVM 实例名称，该属性受主机名命名模式限制。
      - 主机名为自动命名模式：支持批量连续命名或指定模式串命名，最多输入60个字符。默认自动生成实例名，格式为 tke_集群id_worker。
      - 主机名为手动命名模式：实例名称与主机名相同，无需重新配置。
  - **云服务器数量**：实例数量，根据实际需求进行设置。

    > ? 说明：
    > 当**集群类型**选择**独立集群**时，Master&Etcd 节点配置项设置亦可参考 Worker 配置，其数量最少部署3台，可跨可用区部署。

已有节点

通过已有节点，即使用已有云服务器创建集群，详情如下：

> ⚠ 注意：
> - 所选的云服务器需重装系统，重装后云服务器系统盘的所有数据将被清除。
> - 所选的云服务器将迁移至集群所属项目，且云服务器迁移项目会导致安全组解绑，需要重新绑定安全组。
> - 如果您在配置云服务器时填写了数据盘挂载参数，该参数会对 **Master 和 Woker 节点全部生效**。更多相关注意事项请参考 添加已有节点 中的**数据盘挂载**参数说明。

- **集群类型**：提供**托管集群**和**独立集群**两个选项。
  - 托管集群：集群的 Master 和 Etcd 由腾讯云进行管理和维护。
  - 独立集群：集群的 Master 和 Etcd 将会部署在您购置的 CVM 上。
- **集群规格**：根据业务实际情况选择合适的集群规格，详情见 如何选择集群规格。集群规格可手动调整，或者通过自动升配能力自动调整。

- **Worker 配置**：根据实际需求勾选已有云服务器即可。

2. 单击**下一步**，开始 配置云服务器。

3. 云服务器配置

1. 在“云服务器配置”步骤中，参考以下信息进行云服务器配置。如下图所示：

- **容器目录**：勾选即可设置容器和镜像存储目录，建议存储到数据盘。例如 /var/lib/docker。
- **安全组**：安全组具有防火墙的功能，用于设置云服务器的网络访问控制。支持以下设置：
  - 新建并绑定默认安全组，可预览默认安全组规则。
  - 添加安全组，可根据业务需要自定义配置安全组规则。

  更多信息请参见 容器服务安全组设置。
- **登录方式**：提供三种登录方式：
  - **立即关联密钥**：密钥对是通过算法生成的一对参数，是一种比常规密码更安全的登录云服务器的方式。详情请参见 SSH 密钥。
  - **自动生成密码**：自动生成的密码将通过 站内信 发送给您。
  - **设置密码**：请根据提示设置对应密码。
- **安全加固**：默认免费开通 DDoS 防护、WAF 和云镜主机防护，详情请参见 T-Sec 主机安全官网页。
- **云监控**：默认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，详情请参见 云监控 CM 官网主页。

2. （可选）单击**高级设置**，查看或配置更多信息。如下图所示：

- **CAM角色**：可为本批次创建的所有节点绑定相同的 CAM 角色，从而赋予节点该角色绑定的授权策略。详情请参见 管理实例角色。
- **节点启动配置**：指定自定义数据来配置节点，即当节点启动后运行配置的脚本。需确保脚本的可重入及重试逻辑，脚本及其生成的日志文件可在节点的 /usr/local/qcloud/tke/userscript 路径查看。
- **封锁**：勾选“开启封锁”后，将不接受新的 Pod 调度到该节点，需要手动取消封锁的节点，或在自定义数据中执行 取消封锁命令，请按需设置。
- **Label**：单击**新增**，即可进行 Label 自定义设置。集群初始化创建的节点均将自动增加此处设置的 Label，可用于后续根据 Label 筛选、管理节点。

3. 单击**下一步**，开始配置组件。

4. 组件配置

1. 在“组件配置”步骤中，参考以下信息进行组件配置。如下图所示：

- **组件**：组件包含存储、监控、镜像等，您可按需选择。详情请参见 扩展组件概述 。
- **日志服务**：默认开通集群审计服务。详情请参见 集群审计。

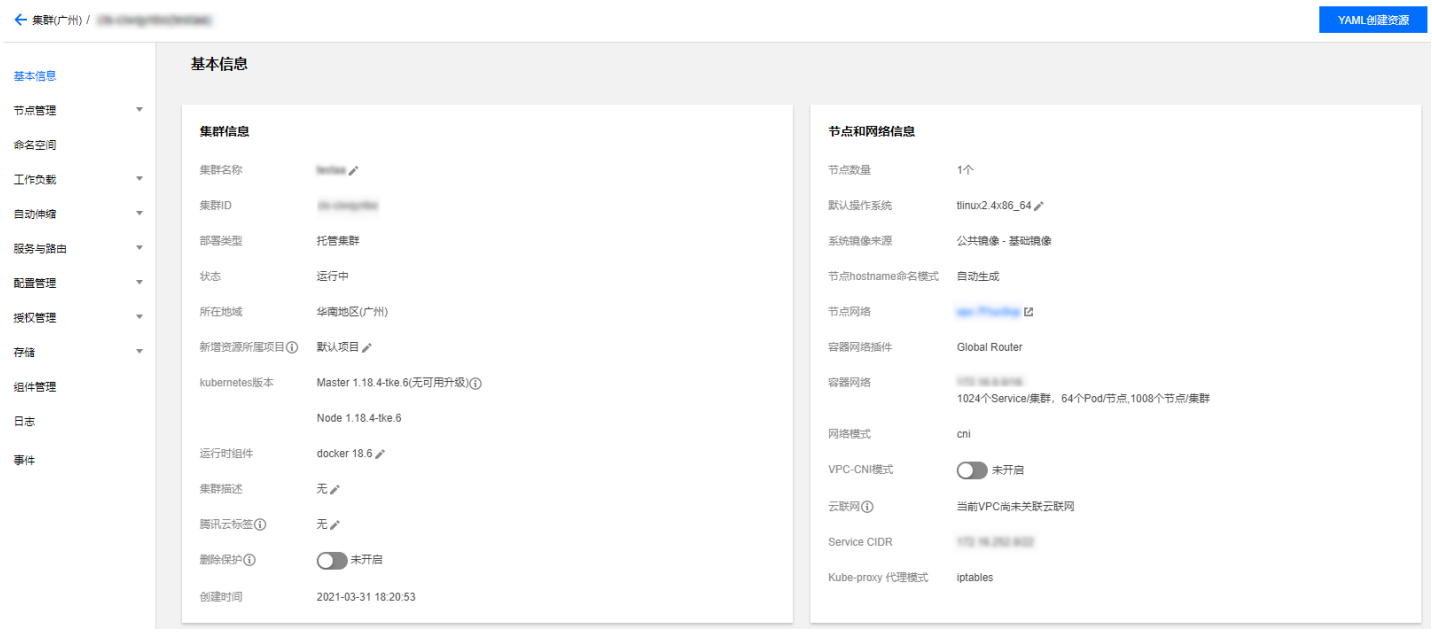
2. 单击**下一步**，检查并确认配置信息。

5. 信息确认

在“信息确认”页面，确认集群的已选配置信息和费用，单击**完成**即可创建一个集群。

查看集群

创建完成的集群将出现在 集群列表 中。您可单击集群 ID 进入集群详情页面。在集群的“基本信息”页面中，您可查看集群信息、节点和网络信息等。如下图所示：



相关文档

您还可以使用 CreateCluster 接口创建集群。详细信息请参见 创建集群 API 文档。

更改集群操作系统

最近更新时间：2022-06-09 14:48:44

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，对存量节点的操作系统无影响。
- 同一集群下节点使用不同版本操作系统，不会对集群功能产生影响。
- 同一脚本不一定适用于所有操作系统，建议您对节点进行脚本配置之后，验证该节点操作系统是否与此脚本相适配。
- 如需使用自定义镜像功能，请 在线咨询 申请。

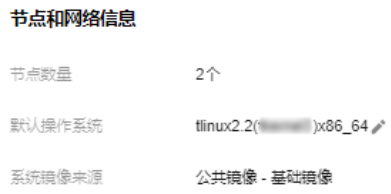
> 注意：
> 如果您需要使用自定义镜像功能，请使用容器服务提供的基础镜像来制作自定义镜像。详情请参见 自定义镜像说明。

操作步骤

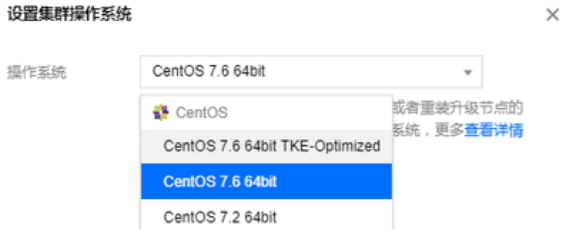
更改集群默认操作系统

> 说明：
> 进行集群默认操作系统更改操作之前，请仔细阅读 操作系统说明 以知悉相关风险。

1. 登录 容器服务控制台 ，单击左侧导航栏中的**集群**。
2. 在“集群管理”页面，选择目标集群所在行右侧的**查看集群凭证**，进入集群基本信息页。
3. 在“节点和网络信息”中，单击**默认操作系统**最右侧的✎。如下图所示：



4. 在弹出的“设置集群操作系统”窗口中，进行操作系统更改，并单击**提交**即可。如下图所示：



删除集群

最近更新时间：2022-06-09 11:36:39

操作场景

本文介绍如何通过腾讯云容器服务控制台 删除不再使用的集群，以免产生不必要的费用。删除集群界面支持展示集群内已有的全部资源，您可查看被销毁的资源，并按需选择是否保留部分资源，请确保您是在知晓操作风险的情况下进行删除操作。

操作步骤

1. 登录 腾讯云容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”列表页面中，选择需删除集群所在行右侧的**更多 > 删除**。如下图所示：

3. 在弹出的“删除集群”窗口中，按需选择保留或删除该集群下已有资源。如下图所示：

4. 查阅集群删除操作风险提示，勾选“我已知晓以上信息并确认删除集群”。

5. 单击**确定**即可删除集群。

集群扩缩容

最近更新时间：2022-06-24 11:31:39

操作场景

本文档指导您对集群进行扩缩容，手动或自动处理应用对资源需求量的变化。TKE 支持以下三种扩缩容方法，您可结合实际情况进行选择：

- 手动添加/移出节点
- 通过弹性伸缩自动添加/移出节点
- 通过超级节点完成应用层的扩缩容（无需通过节点来扩缩容）

前提条件

1. 已登录 容器服务控制台 。
2. 已 创建集群。

操作步骤

手动添加/移出节点

您可通过新建节点或添加已有节点两种方式进行手动添加节点，实现集群的手动扩容。通过移出节点，实现集群的手动缩容。

新建节点

新建节点过程中，您可以在 “新建节点” 页面配置云服务器（CVM），对集群进行扩容。
具体操作请参考 新建节点。

添加已有节点

> 注意：
>
> - 当前仅支持添加同一 VPC 下的 CVM。
> - 添加已有节点到集群，会根据您的设置重装该 CVM 的操作系统。
> - 添加已有节点到集群，会迁移 CVM 所属项目到集群所设置的项目。
> - 有且仅有一块数据盘的节点加入到集群，可以选择设置数据盘挂载相关参数。

添加过程中，您可以在 “添加已有节点” 页面选择并配置需要添加到集群的 CVM，对集群进行扩容。
具体操作请参考 添加已有节点。

移出节点

请参考 移出节点 对集群进行缩容。

通过弹性伸缩自动添加/移出节点

弹性伸缩依赖社区组件 Cluster Autoscaler（CA），可以动态地调整集群的节点数量来满足业务的资源需求。更多弹性伸缩原理请参见 节点池概述 。

通过超级节点进行业务扩容

超级节点是一种调度能力，支持将标准 Kubernetes 集群中的 Pod 调度到集群服务器节点之外的资源中，实现资源不足时的动态资源补给。详情可参见 超级节点概述 。

常见问题

扩容缩容的相关问题可参见 扩容缩容相关。

连接集群

最近更新时间：2022-03-30 10:49:00

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。

前提条件

请安装 curl 软件。
请根据操作系统的类型，选择获取 Kubectl 工具的方式：

> 说明：
> 请根据实际需求，将命令行中的 “v1.18.4” 替换成业务所需的 Kubectl 版本。一般来说，客户端的 Kubectl 与服务端的 Kubernetes 的最高版本保持一致即可。您可以在**基本信息**的“集群信息”模块里查看 Kubernetes 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/v1.18.4/bin/darwin/amd64/kubectl
```

- **Linux 系统**
  执行以下命令，获取 Kubectl 工具：

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/linux/amd64/kubectl
```

- **Windows 系统**
  执行以下命令，获取 Kubectl 工具：

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/windows/amd64/kubectl.exe
```

操作步骤

安装 Kubectl 工具

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

> 说明：
> - 如果您已经安装 Kubectl 工具，请忽略本步骤。
> - 此步骤以 Linux 系统为例。

2. 执行以下命令，添加执行权限。

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

3. 执行以下命令，测试安装结果。

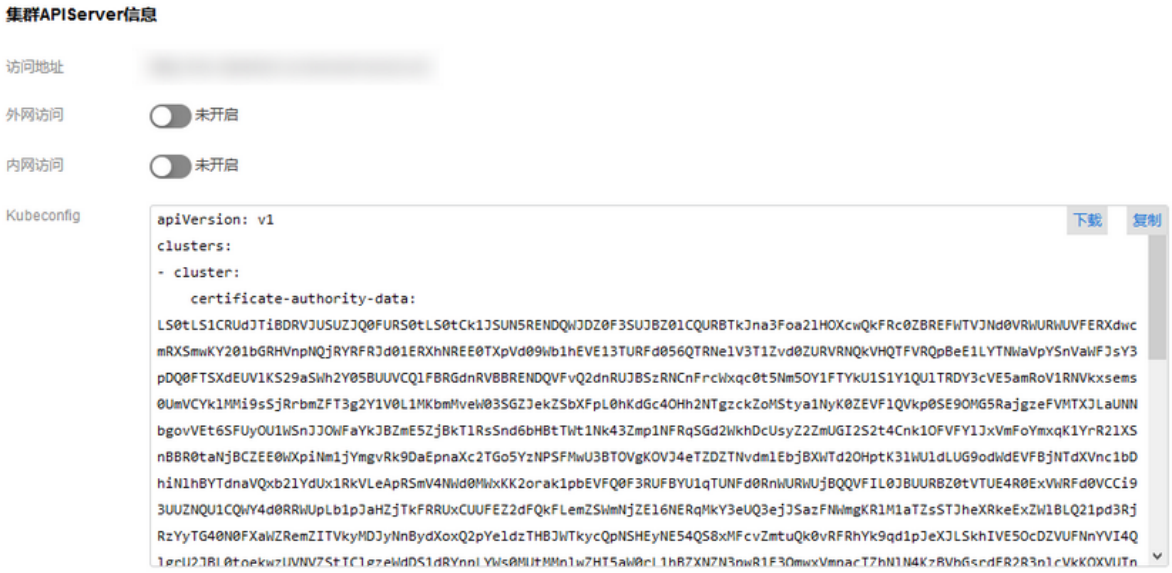
```
kubectl version
```

如若输出类似以下版本信息，即表示安装成功。

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

配置 Kubeconfig

1. 登录容器服务控制台，选择左侧导航栏中的 集群，进入集群管理界面。
2. 单击需要连接的**集群 ID/名称**，进入集群详情页。
3. 选择左侧导航栏中的**基本信息**，即可在“基本信息”页面中查看“集群APIServer信息”模块中该集群的访问地址、外网/内网访问状态、Kubeconfig 访问凭证内容等信息。如下图所示：



- **访问地址**：集群 APIServer 地址。请注意该地址不支持复制粘贴至浏览器进行访问。
- **获取访问入口**：请根据实际需求进行设置。
  - **外网访问**：默认不开启。开启外网访问**会将集群 apiserver 暴露到公网，请谨慎操作**。且需配置来源授权，默认全拒绝，您可配置放通单个 IP 或 CIDR，强烈不建议配置 0.0.0.0/0 放通全部来源。
  - **内网访问**：默认不开启。开启内网访问时，需配置一个子网，开启成功后将在已配置的子网中分配 IP 地址。
  - **使用 Kubernetes 的 service IP**：在集群详情页面中，选择左侧的**服务与路由** > **Service**获取 default 命名空间下 Kubernetes 的 service IP。将 Kubeconfig 文件中 clusters.cluster.server 字段替换为 https://<IP>:443 即可。**注意**：Kubernetes service 是 ClusterIP 模式，仅适用于集群内访问。
- **Kubeconfig**：该集群的访问凭证，可复制、下载。

4. 根据实际情况进行集群凭据配置。

   配置前，请判断当前访问客户端是否已经配置过任何集群的访问凭证：

   - **否**，即 ~/.kube/config 文件内容为空，可直接复制已获取的 Kubeconfig 访问凭证内容并粘贴入 ~/.kube/config 中。若客户端无 ~/.kube/config 文件，您可直接创建。
   - **是**，您可下载已获取的 Kubeconfig 至指定位置，并依次执行以下命令以合并多个集群的 config。

     ```
     KUBECONFIG=~/.kube/config:~/Downloads/cls-3jju4zdc-config kubectl config view --merge --flatten > ~/.kube/config
     ```

     ```
     export KUBECONFIG=~/.kube/config
     ```

     其中，~/Downloads/cls-3jju4zdc-config 为本集群的 Kubeconfig 的文件路径，请替换为下载至本地后的实际路径。

访问 Kubernetes 集群

1. 完成 Kubeconfig 配置后，依次执行以下命令查看并切换 context 以访问本集群。

   ```
   kubectl config get-contexts
   ```

```
kubectl config use-context cls-3jju4zdc-context-default
```

2. 执行以下命令， 测试是否可正常访问集群。

```
kubectl get node
```

如果无法连接请查看是否已经开启公网访问或内网访问入口，并确保访问客户端在指定的网络环境内。

相关说明

Kubectl 命令行介绍

Kubectl 是一个用于 Kubernetes 集群操作的命令行工具。本文涵盖 kubectl 语法、常见命令操作并提供常见示例。有关每个命令（包括所有主命令和子命令）的详细信息，请参阅 kubectl 参考文档 或使用 kubectl help 命令查看详细帮助，kubectl 安装说明请参见 安装 Kubectl 工具。

升级集群

最近更新时间：2022-04-18 14:13:42

操作场景

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

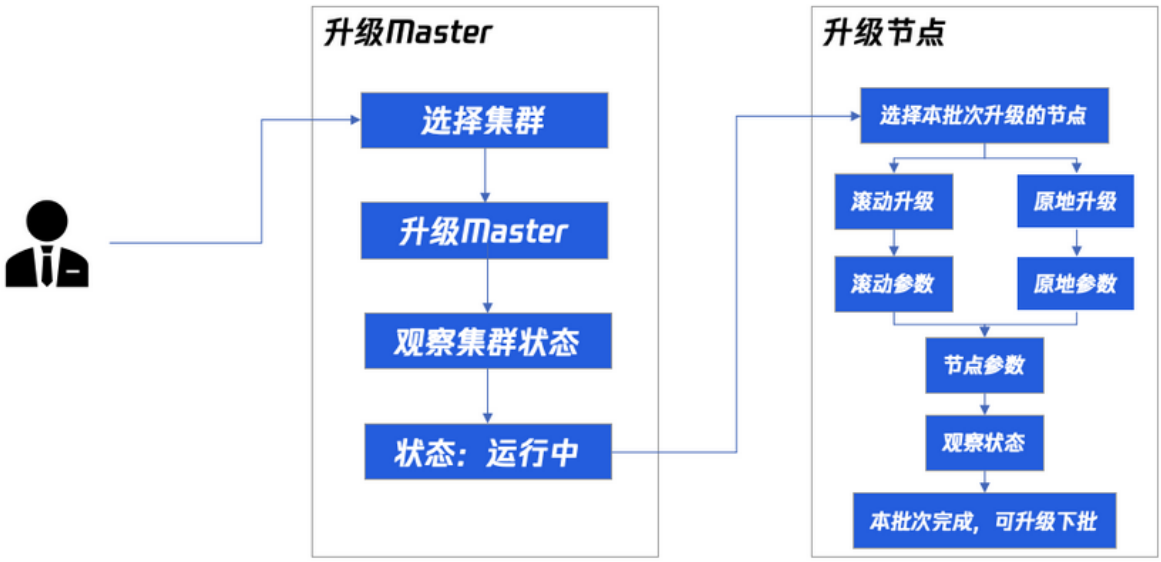
升级须知

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

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

操作步骤

升级集群的两个步骤是 升级 Master kubernetes 版本 和 升级 Node Kubernetes 版本。具体信息如下图所示：



升级 Master Kubernetes 版本

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

Master 大版本与小版本升级说明

目前 Master 升级已支持**大版本升级**（例如从1.14升级到1.16）、**小版本升级**（例如从1.14.3升级到1.14.6，或者从v1.18.4-tke.5升级到v1.18.4-tke.6），强烈建议您升级前先查阅对应的功能发布记录：

- 在升级 kubernetes 大版本之前，建议您查阅 TKE Kubernetes 大版本更新说明。
- 在升级 kubernetes 小版本之前，建议您查阅 TKE Kubernetes Revision 版本历史。

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

注意事项

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

Master 升级技术原理

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

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

Master 升级操作步骤

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 在“集群管理”页面，选择需进行 Master Kubernetes 版本升级的集群 ID，进入集群详情页。
3. 在集群详情页面，选择左侧**基本信息**。
4. 在集群“基本信息”页面的集群信息模块，单击 Master 版本右侧的**升级**。如下图所示：

5. 在弹出窗口中单击**提交**，等待升级完成。

6. 您可以在集群管理页对应的集群状态处查看升级进度，也可以在升级进度弹窗中查看当前升级进展、Master 节点升级进度（托管集群不显示具体 Matser 节点列表）、升级持续时间。如下图所示：

7. 该示例集群 Kubernetes 版本升级前 Master 版本为1.10.5，升级完成后为 Master 1.12.4。如下图所示：

升级 Node Kubernetes 版本

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

注意事项

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

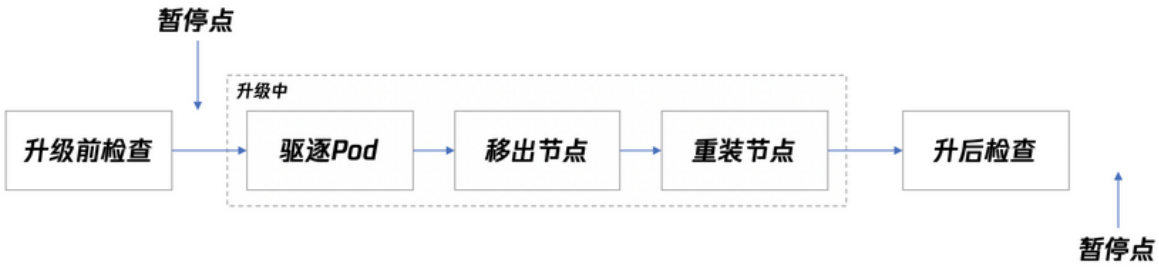
选择升级方式

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

- **重装滚动升级**：采用重装节点的方式升级节点版本。仅支持大版本升级，例如1.10可升级至1.12。
- **原地滚动升级**：原地不重装，仅替换 Kubelet、kube-proxy 等组件。支持大版本、小版本升级，例如1.10可升级至1.12，1.14.3可升级至1.14.8。

重装滚动升级执行原理

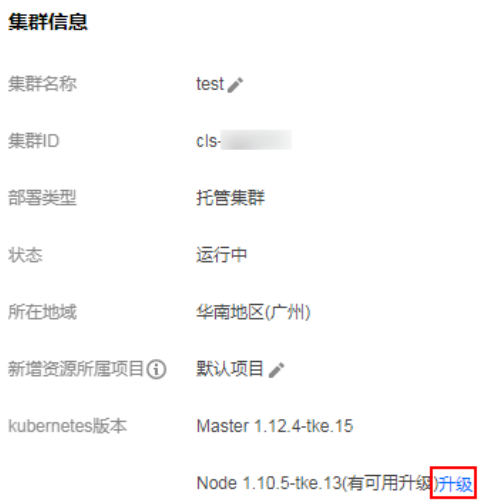
基于重装的节点升级采用滚动升级的方式，同一时间只会对一个节点进行升级，只有当前节点升级成功才会进行下个节点的升级。如下图所示：



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

重装滚动升级操作步骤（Node Kubernetes 版本）

1. 登录容器服务控制台 ，选择左侧导航栏中的 **集群**。
2. 在“集群管理”页面，选择需进行 Node Kubernetes 版本升级的集群 ID，进入集群详情页。
3. 在集群详情页面，选择左侧**基本信息**。
4. 在集群“基本信息”页面的集群信息模块，单击 Node Kubernetes 版本右侧的**升级**。如下图所示：



5. 在“升级须知”步骤中，选择升级方式为**重装滚动升级**，仔细阅读升级须知。勾选**我已阅读并同意上述技术条款**，并单击**下一步**。如下图所示：

> ⚠ 注意：
> 该升级方式将重装系统，原有数据将会被清除，请注意提前备份数据。

6. 在“节点选择”步骤中，选择本批次需要升级的节点，并单击**下一步**。
7. 在“升级设置”步骤中，按需填写节点信息，并单击**下一步**。
8. 在“确认”步骤中，确认信息并单击**完成**即可开始升级。
9. 查看节点升级进度， 直至所有节点升级完成。

原地滚动升级执行原理

节点原地升级采用滚动升级的方式，同一时间只会对一个节点进行升级，只有当前节点升级成功才会进行下个节点的升级。原地升级目前已同时支持大版本升级以及大版本的不同小版本升级。如下图所示：

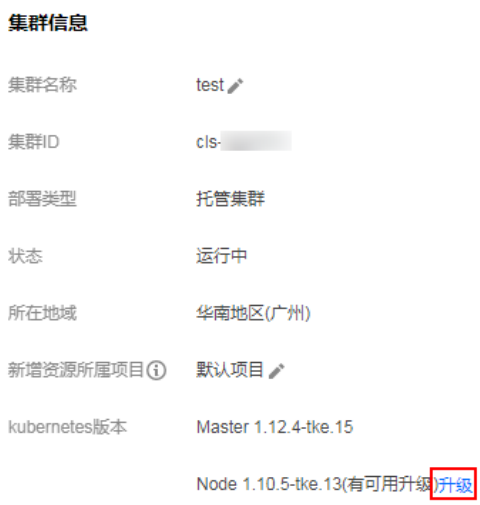
步骤描述如下：

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

原地滚动升级操作步骤

1. 登录容器服务控制台 ，选择左侧导航栏中的 **集群**。
2. 在 “集群管理” 页面，选择需进行 Node Kubernetes 升级的集群 ID，进入集群详情页。
3. 在集群详情页面，选择左侧**基本信息**。

4. 在集群“基本信息”页面的集群信息模块，单击 Node Kubernetes 版本右侧的**升级**。如下图所示：



5. 在**升级须知**中，选择升级方式为“原地滚动升级”，仔细阅读升级须知。勾选**我已阅读并同意上述技术条款**，并单击**下一步**。如下图所示：

6. 在“节点选择”步骤中，选择本批次需要升级的节点，单击**下一步**。
7. 在“确认”步骤中，确认信息并单击**完成**即可开始升级。
8. 查看节点升级进度， 直至所有节点升级完成。

集群启用 IPVS

最近更新时间：2022-06-09 11:19:22

操作场景

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

注意事项

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

操作步骤

1. 登录 容器服务控制台 。
2. 参考 创建集群，在 “创建集群” 页面中，将 “Kubernetes版本” 设置为高于1.10的 Kubernetes 版本，并单击**高级设置**，开启 “ipvs 支持” 。如下图所示：

3. 按照页面提示逐步操作，完成集群的创建。

集群启用 GPU 调度

最近更新时间：2022-06-09 11:19:09

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，您可以使用腾讯云容器服务支持 GPU 功能，通过该功能可以帮助您快速使用 GPU 容器。
启用 GPU 调度有以下两种方式：

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

前提条件

已登录 容器服务控制台 。

注意事项

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

操作步骤

在集群中添加 GPU 节点

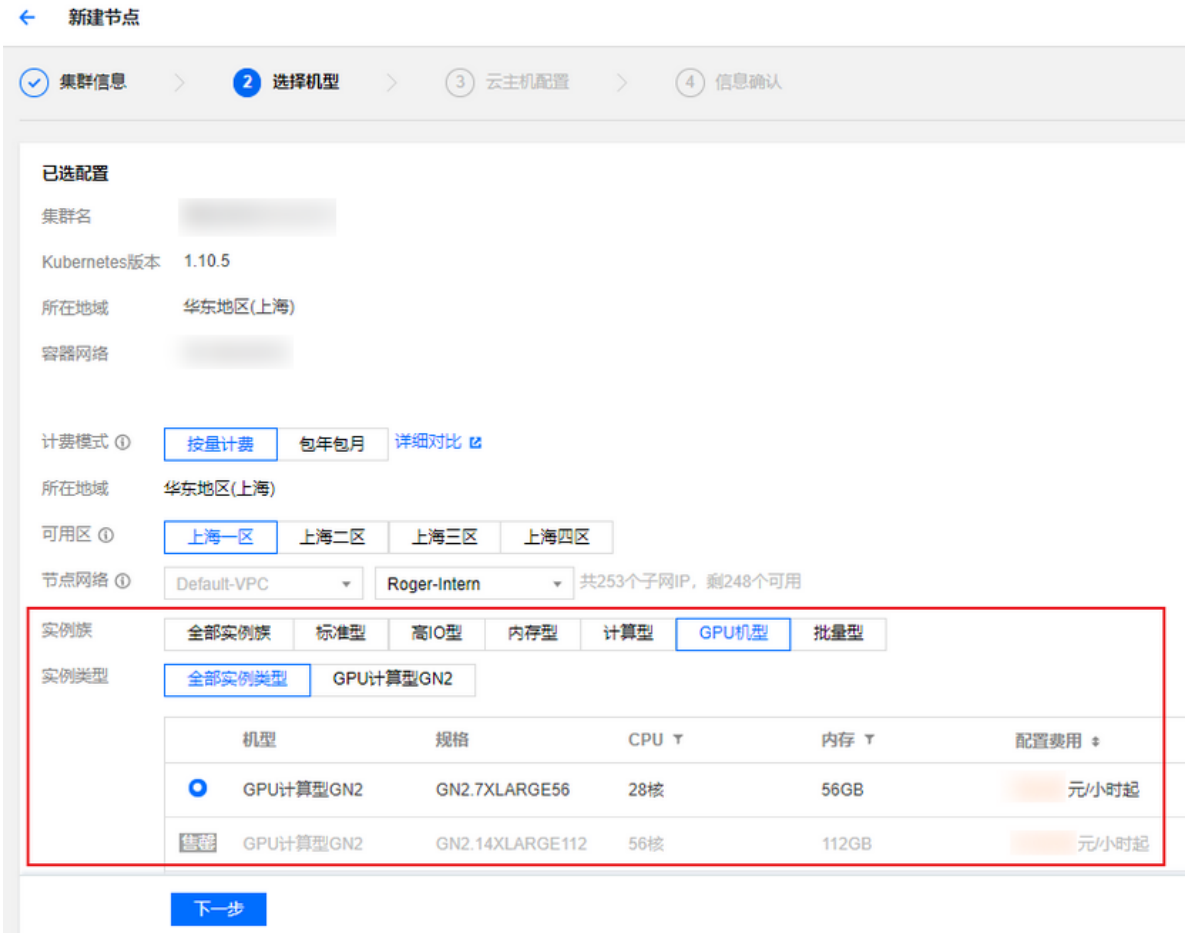
添加 GPU 节点有以下两种方法：

- 新建 GPU 云服务器
- 添加已有 GPU 云服务器

新建 GPU 云服务器

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

3. 在 “选择机型” 页面，将 “实例族” 设置为 “**GPU机型**”，并选择 GPU 计算型的实例类型。如下图所示：



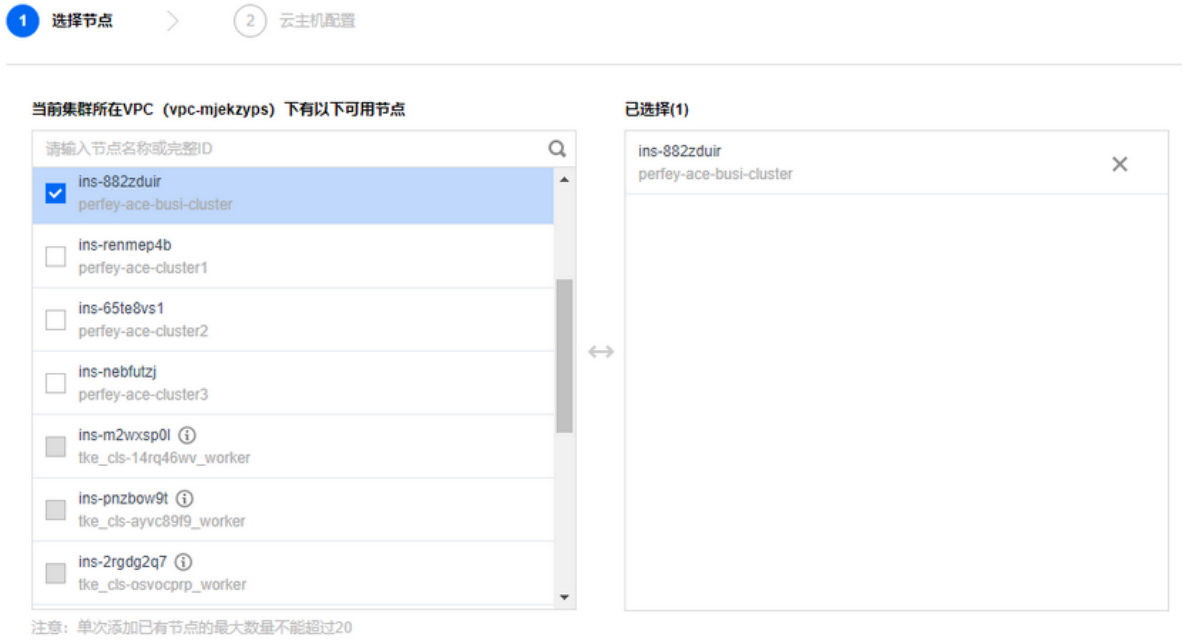
4. 按照页面提示逐步操作，完成创建。

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

添加已有 GPU 云服务器

1. 在左侧导航栏中，单击 **集群**，进入 “集群管理” 页面。
2. 在需要添加已有 GPU 云服务器的集群行中，单击**添加已有节点**。

3. 在 “选择节点” 页面，勾选已有的 GPU 节点，单击**下一步**。如下图所示：



4. 按照页面提示逐步操作，完成添加。

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

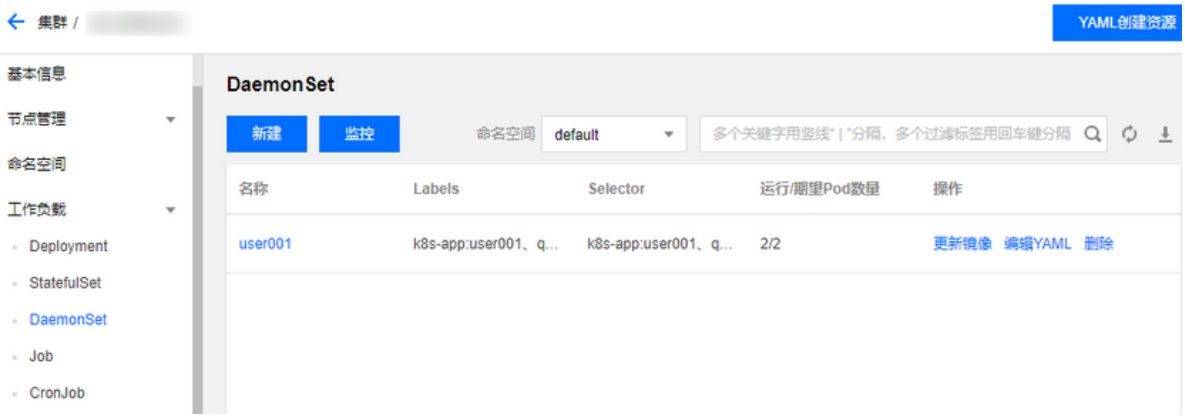
创建 GPU 服务的容器

创建 GPU 服务的容器有以下两种方法：

- 通过控制台方式创建
- 通过应用或 Kubectl 命令创建

通过控制台方式创建

1. 在左侧导航栏中，单击 **集群**，进入 “集群管理” 页面。
2. 单击需要创建工作负载的集群 ID/名称，进入待创建工作负载的集群管理页面。
3. 在 “工作负载” 下，任意选择工作负载类型，进入对应的信息页面。例如，选择**工作负载** > **DaemonSet**，进入 DaemonSet 信息页面。如下图所示：



4. 单击**新建**，进入 “新建Workload” 页面。

5. 根据页面信息，设置工作负载名、命名空间等信息。并在 “GPU限制” 中，设置 GPU 限制的数量。如下图所示：

6. 单击**创建Workload**，完成创建。

通过应用或 Kubectl 命令创建

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

自定义 Kubernetes 组件启动参数

最近更新时间：2022-04-21 16:18:54

操作场景

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

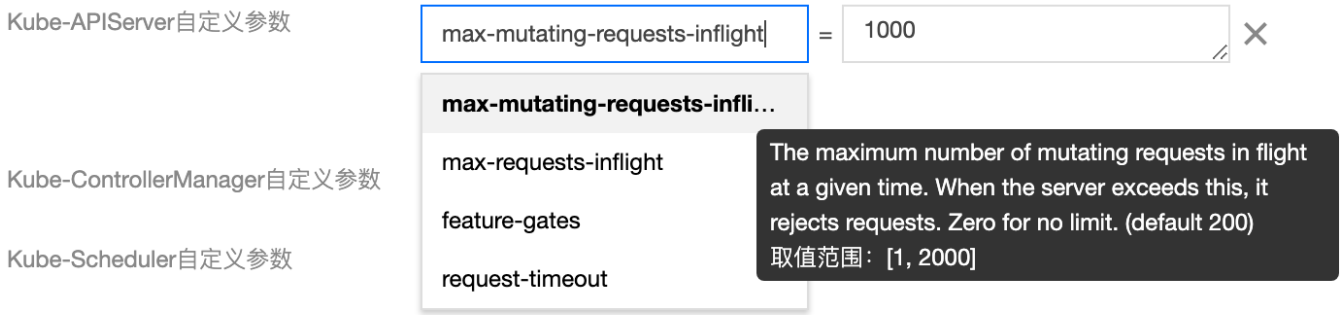
注意事项

- 使用自定义 Kubernetes 组件启动参数功能需 在线咨询 进行申请。
- 自定义 Kubernetes 组件启动参数功能属于租户、集群及可设置自定义参数维度开关，您在提交工单时需提供账号 ID、集群 ID、需要设置的组件和组件的参数。
- 升级 Kubernetes 集群版本，由于 Kubernetes 跨版本后启动参数可能存在不兼容的情况，大版本升级不会保留您原集群版本的自定义 Kubernetes 组件参数，您需要重新设置自定义的 Kubernetes 的组件参数。

操作说明

创建集群设置自定义 Kubernetes 组件参数

1. 登录 腾讯云容器服务控制台 ，单击左侧导航栏中的**集群**。
2. 在“集群管理”页面，单击集群列表上方的**新建**。
3. 在“创建集群”页面，选择**高级设置** > **设置kubernetes自定义组件参数**。如下图所示：



设置节点的自定义 Kubelet 参数

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

集群升级设置自定义 Kubernetes 组件参数

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

镜像

镜像概述

最近更新时间：2022-06-22 16:18:36

概述

本文档介绍腾讯云容器服务 TKE 支持的镜像类型，对应的使用场景 以及使用须知。TKE 支持以下三种类型的镜像，镜像详细说明可参见 镜像类型说明。

- **公共镜像**：公共镜像是由腾讯云官方提供的镜像，包含基础操作系统和腾讯云提供的初始化组件，所有用户均可使用。
- **自定义镜像**：由用户通过镜像制作功能制作，或通过镜像导入功能导入的镜像。仅创建者与共享者可以使用。自定义镜像属于非标环境，腾讯云不提供官方支持以及持续维护。
- **市场镜像**：针对特定使用场景 提供的镜像，例如 qGPU 共享场景。所有用户均可使用，除操作系统外还集成了某些特定应用程序。

注意事项

- 操作系统有两个级别：**集群级别**和**节点池级别**。
  - 在集群内进行新增节点、添加已有节点、节点升级操作时，均会使用集群级别设置的操作系统。
  - 在节点池内部进行添加已有节点、节点扩容操作时，会使用节点池级别设置的操作系统。
- 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**。

TKE 支持的公共镜像列表

TKE 为您提供以下**公共镜像**，请根据实际情况进行选择。

> 说明：
> 若 TKE 后期计划调整镜像逻辑，会提前**至少一周**通过站内信、短信、邮件的方式进行通知，请您放心使用。
> 镜像逻辑变化不会对您之前使用旧版本镜像创建的存量节点产生任何影响。为了达到更好的使用效果，建议您使用新版本基础镜像。

镜像 ID	Os Name	控制台操作系统展示名	OS 类型	发布状态	备注
img-eb30mz89	tlinux3.1x86_64	TencentOS Server 3.1 (TK4)	Tencent OS Server	全量发布	推荐使用 Tencent OS Server 最新发行版 内核版本：5.4.119 不支持自动安装 GPU 驱动，您需要自行安装 GPU 驱动
img-hdt9xxkt	tlinux2.4x86_64	TencentOS Server 2.4 曾用名：Tencent linux release 2.4 (Final)	Tlinux	全量发布	内核版本：4.14.105
img-22trbn9x	ubuntu20.04x86_64	Ubuntu Server 20.04.1 LTS 64bit	Ubuntu	内测中，请 提交工单 进行申请	Ubuntu 20.04.1 公版内核
img-pi0ii46r	ubuntu18.04.1x86_64	Ubuntu 18.04 LTS 64bit	Ubuntu	全量发布	Ubuntu 18.04.1 公版内核
img-25szkc8t	centos8.0x86_64	CentOS 8.0	CentOS	内测中，请 提交工单 进行申请	Centos 8.0 公版内核
img-9qabwvbn	centos7.6.0_x64	CentOS 7.6	CentOS	全量发布	Centos 7.6 公版内核

自定义镜像说明

最近更新时间：2022-01-17 17:47:25

操作场景

本文档介绍如何使用腾讯云容器服务（TKE）提供的基础镜像进行自定义镜像制作。

> 说明：
> 基础镜像包括 TKE 支持的**公共镜像**和**市场镜像**，支持的镜像列表请参考 镜像概述。

使用须知

- 目前仅支持同类型的操作系统镜像的制作。例如，使用 CentOS 基础镜像制作 CentOS 类的自定义镜像。
- 如果您使用自定义镜像功能，请使用 TKE 提供的基础镜像来制作自定义镜像。
- 若 TKE 后期计划调整镜像逻辑，会提前**至少一周**通过站内信、短信、邮件的方式进行通知，请您放心使用。镜像逻辑变化可能会造成用原有自定义镜像新建节点失败，您需要重新制作自定义镜像。如集群有使用节点池，需调整节点池的镜像配置。
- 如需使用自定义镜像功能，请 提交工单 申请。

注意事项

制作自定义镜像前，请务必仔细阅读以下注意事项。自定义镜像属于非标环境，腾讯云不提供官方支持以及持续维护。

- 请勿随意修改/etc/fstab。
- 制作镜像之后请及时清理 /var/lib/cloud/instances/${instance-id} 目录。
- 如果您在自定义镜像中预装了运行时组件，节点初始化无法正常进行，会直接报错。

操作步骤

本文以使用基础镜像创建云服务器（CVM）为例。

1. 创建 CVM

1. 登录 云服务器控制台，选择**新建**进入云服务器购买页面。
2. 在“镜像”中选择容器服务基础镜像，这里以公共镜像 TencentOS Server 3.1 为例。如下图所示：

3. 其他选项设置请参考 新建 CVM 实例。

2. 创建自定义镜像

1. 请参考 使用标准登录方式登录 Linux 实例（推荐）登录 CVM。
2. 执行以下命令，新建 test.txt 文件。

```
vi test.txt
```

3. 按 **i** 进入编辑模式，并写入以下内容。

```
this is customer cvm images test
```

4. 按 **Esc** 并输入 **:wq** 退出并保存。
5. 请参考 创建自定义镜像 完成创建。

3. 使用自定义镜像

自定义镜像制作完成后，即可使用该镜像创建 TKE 集群。

在“创建集群”页面的“镜像提供方”中选择**自定义镜像**，“操作系统”则选择已创建的自定义镜像。如下图所示：

其他选项设置请参考 创建集群。

4. 验证自定义镜像

1. 登录 TKE 控制台，选择左侧导航栏中的**集群**。
2. 选择使用自定义镜像创建的集群 ID，进入集群详情页。
3. 选择左侧导航栏中的**节点管理** > **节点**，记录需要登录验证的节点 ID。
4. 登录 云服务器控制台，在搜索框中输入记录的节点 ID 并单击🔍，即可看到已创建的节点。如下图所示：

5. 请参考 使用标准登录方式登录 Linux 实例（推荐），登录节点。
6. 执行以下命令，验证自定义镜像。

```
cat test.txt
```

返回结果如下，则表示该节点已使用自定义镜像。

```
[root@VM_0_118_centos ~]# cat test.txt
this is customer cvm images test
[root@VM_0_118_centos ~]#
```

使用总结

- **制作自定义镜像必须使用 TKE 提供的基础镜像，没有满足此要求的自定义的镜像在 TKE 控制台上不会显示。**
- **若在自定义镜像中对** /etc/resolv.conf **文件设置文件保护（**chattr +i /etc/resolv.conf**），则会导致** clount-init **失败。由于 TKE 依赖于** cloud-init **成功状态，最终会导致节点加入集群失败。**
- **由于** rc.local **和** container_cluster_agent **无法保证执行顺序，会导致用户** start_init.sh **脚本拷贝的数据丢失。不建议您把** start_init.sh **放在** rc.local **中执行，建议在** user-data **中执行。**
- **曾经制作过镜像的节点上如果保留了** /var/lib/cloud **目录，那么** /var/lib/cloud/instances/ins-1cgoe1y9/sem **目录下的** config_scripts_user **文件会影响** cloud-init **服务无法正常执行，会导致该节点加入 TKE 集群内时修改的节点主机名无法生效。**
- **在自定义镜像中添加个人 yum 源时，若放置在不合适的目录下（例如** /etc/yum.repo.d/**），则会引起** container-cluster-agent **在执行** yum install **操作时报错，从而跳过该步骤，导致 agent 安装 yum 源失败。**

节点管理
节点概述

最近更新时间：2022-01-17 15:08:50

简介

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

节点相关操作

- 新增节点
- 移出节点
- 驱逐或封锁节点
- 设置节点的启动脚本
- 使用 GPU 节点
- 设置节点 Label

节点生命周期

最近更新时间：2022-01-19 11:12:57

节点生命周期状态说明

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

节点资源预留说明

最近更新时间：2022-07-01 14:40:37

TKE 需要占用节点一定的资源来运行相关组件（例如：kubelet、kube-proxy、Runtime 等），因此会造成**节点资源总数与集群中可分配资源数存在差异**。本文介绍 TKE 集群中的节点资源预留策略和注意事项，以便在部署应用时合理设置 Pod 的请求资源量和限制资源量。

节点可分配资源计算策略

计算公式

ALLOCATABLE = CAPACITY − RESERVED − EVICTION − THRESHOLD

节点 CPU 预留规则

节点 CPU	预留规则	说明
1c <= CPU <= 4c	固定预留 0.1c	–
4c < CPU <= 64c	4c 以下预留 0.1c，超过 4c 部分预留 2.5%	例如：CPU = 32c 预留资源 = 0.1 + (32 − 4) * 2.5% = 0.8c
64c < CPU <= 128c	4c 以下预留 0.1c，4c～64c 预留 2.5%，超过 64c 部分预留 1.25%	例如：CPU = 96c 预留资源 = 0.1 + (64 − 4) * 2.5% + (96 − 64) * 1.25%= 1.2c
CPU > 128c	4c 以下预留 0.1c，4c～64c 预留 2.5%，64c～128c 预留 1.25%，超过 128c 部分预留 0.5%	例如：CPU = 196c 预留资源 = 0.1 + (64 − 4) * 2.5% + (96 − 64) * 1.25% + (196 − 128) * 0.5%= 1.54c

节点内存预留规则

节点内存	预留规则	说明
1G <= 内存 <= 4G	固定预留 25%	例如：内存 = 2G 预留资源 = 2 * 25% = 500MB
4G < 内存 <= 8G	4G 以下预留 25%，超过 4G 部分预留 20%	例如：内存 = 8G 预留资源 = 4 * 25% + (8 − 4) * 20% = 1843MB
8G < 内存 <= 16G	4G 以下预留 25%，4G～8G 预留 20%，超过 8G 部分预留 10%	例如：内存 = 12G 预留资源 = 4 * 25% + (8 − 4) * 20% + (16 − 8) * 10%= 2252MB
16G < 内存 <= 128G	4G 以下预留 25%，4G～8G 预留 20%，8G～16G 预留 10%，超过 16G 部分预留 6%	例如：内存 = 32G 预留资源 = 4 * 25% + (8 − 4) * 20% + (16 − 8) * 10% + (32 − 16) * 6% = 3645MB
内存 > 128G	4G 以下预留 25%，4G～8G 预留 20%，8G～16G 预留 10%，16G～128G 预留 6%，超过 128G 部分预留 2%	例如：内存 = 320G 预留资源 = 4 * 25% + (8 − 4) * 20% + (16 − 8) * 10% + (32 − 16) * 6% + (320 − 128) * 2% = 13475MB

> 说明：
> 用户可以通过自定义 kubelet 参数的方式来修改 kube-reserved 以达到修改节点预留资源的目的，建议给节点组件预留充足的 CPU 和内存资源来保证节点的稳定性。

查看节点可分配资源

检查集群中可用的节点可分配资源，请执行以下命令，并将 NODE_NAME 替换为您要检查的节点的名称。输出结果包含 Capacity 和 Allocatable 字段，并提供了针对 CPU、内存和临时存储的测量结果。

```
kubectl describe node NODE_NAME | grep Allocatable -B 7 -A 6
```

注意事项

- 该预留策略对新增节点自动生效，无需手动配置。
- 为保证业务稳定性，该预留策略不会对已有节点自动生效。因为该资源预留的计算方式可能会造成节点的可分配资源变少，对于资源水位较高的节点，可能会触发节点驱逐。
- 若您希望对已有节点应用该资源预留策略，您可以通过 容器服务控制台 将该节点在不直接销毁的情况下移出集群，然后添加已有节点，新添加的节点会默认执行该资源预留策略。

新增节点

最近更新时间：2022-06-09 11:16:46

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 容器服务控制台 。

操作步骤

新建节点

1. 在左侧导航栏中，单击 **集群**，进入 “集群管理” 页面。
2. 单击需要创建云服务器的集群 ID，进入该集群详情页。
3. 选择页面左侧**节点管理** > **节点**，进入节点列表页面，单击**新建节点**。
4. 在 “新建节点” 页面，根据实际需求配置相关参数。如下图所示：

主要参数信息如下：

- **计费模式**：提供**按量计费**和**包年包月**两种计费模式。详情请参见 计费模式。
- **可用区**：该参数仅用来筛选可用区下可用的子网列表。
- **集群网络**：选择为本次新建节点分配 IP 的子网，单次创建节点操作只支持单子网。
- **机型配置**：单击**请选择机型**，在弹出的“机型配置”窗口中参考以下信息按需选择：
  - **机型**：支持通过 CPU 核数、内存大小及实例类型进行筛选。详情请参见 实例规格 和 快速入门 Linux 云服务器。
  - **系统盘**：存储控制、调度云服务器运行的系统集合。支持查看所选机型的可选系统盘类型，请参考 云硬盘类型 并根据实际需求进行选择。
  - **数据盘**：用于存储所有的用户数据。
- **实例名称**：控制台显示的云服务器 CVM 实例名称，该属性受主机名命名模式限制。提供以下两种命名方式：
  - **自动命名**：主机名为自动命名模式，支持批量连续命名或指定模式串命名，最多输入60个字符。默认自动生成实例名，格式为 tke_集群id_worker。

- **手动命名**：主机名为手动命名模式，实例名称与主机名相同，无需重新配置。
- **登录方式**：提供以下三种登录方式，请根据实际情况进行选择。
  - **立即关联密钥**：密钥对是通过算法生成的一对参数，是一种比常规密码更安全的登录云服务器的方式。详情请参见 SSH 密钥。
    - **SSH密钥**：该配置项仅在选择**立即关联密钥**登录方式时出现，在下拉框中选用已有密钥即可。若需新建，请参考 创建 SSH 密钥。
  - **自动生成密码**：自动生成的密码将通过 站内信 发送给您。
  - **设置密码**：请根据提示设置对应密码。
- **安全组**：默认为创建集群时所设置的安全组，可根据实际需要进行更换或添加。
- **数量**：创建实例数量，请根据实际需求进行设置。

5. （可选）单击“新建节点”页面中的**更多设置**，查看或配置更多信息。如下图所示：

- **CAM角色**：可为本批次创建的所有节点绑定相同的 CAM 角色，赋予节点该角色绑定的授权策略。详情请参见 管理实例角色。
- **容器目录**：勾选即可设置容器和镜像存储目录，建议存储到数据盘。例如 /var/lib/docker。
- **安全加固**：默认免费开通 DDoS 防护、WAF 和云镜主机防护，详情请参见 T-Sec 主机安全官网页。
- **云监控**：默认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。详情请参见 云监控 CM 官网主页。
- **封锁初始节点**：勾选“开启封锁”后，将不接受新的 Pod 调度到该节点，需要手动取消封锁的节点，或在自定义数据中执行 取消封锁命令，请按需设置。
- **Label**：单击**新增Label**，即可进行 Label 自定义设置。可用于后续根据 Label 筛选、管理节点。
- **自定义数据**：指定自定义数据来配置节点，即当节点启动后运行配置的脚本。需确保脚本的可重入及重试逻辑，脚本及其生成的日志文件可在节点的 /usr/local/qcloud/tke/userscript 路径查看。

添加已有节点

> ⚠ 注意：
>
> - 当前仅支持添加同一 VPC 下的云服务器。
> - 请勿添加公网网关 CVM 加入集群，该类型 CVM 重装加入集群后产生 DNS 异常，会导致该节点不可用。

1. 在左侧导航栏中，单击 **集群**，进入 “集群管理” 页面。
2. 单击需要添加已有节点的集群ID，进入该集群详情页。

3. 选择**节点管理** > **节点**，单击**添加已有节点**。如下图所示：

4. 在 “选择节点” 页面，勾选需要添加的节点，单击**下一步**。
5. 在 “云服务器配置” 页面，配置需要添加到集群的云服务器。

   主要参数信息如下：
   - 数据盘挂载：格式化挂载相关设置：需要填写设备名称，格式化系统以及挂载点
     - 不勾选：不设置数据盘挂载选项，可手动或者使用脚本挂载。
     - 勾选：需要填写以下参数：设备名称，格式化系统（可以选择不格式化），挂载点。

       如果您想将 /dev/vdb 这块设备格式化成 ext4，并挂载到 /var/lib/docker 目录下，可以这样设置：

       设备名称：/dev/vdb, 格式化系统：ext4，挂载点：/var/lib/docker

     > ⚠
     > - 提前备份重要数据，如果已经自行格式化盘，则无需选择格式化系统，只需填写挂载点。
     > - 您填写的格式化挂载设置会对本批次添加节点全部生效，请确认填写的设备名称，例如 /dev/vdb 符合您的预期（如果您对 CBS 做了热插拔等操作，设备名称可能会变化）。
     > - 如果您对盘做了分区 /LVM，在设备名称处填写分区名 /LVM 名，配置对应的格式化挂载参数即可。
     > - 如果您填写了错误的设备名称，系统会报错并终止节点初始化流程。
     > - 如果您填写的挂载点不存在，系统会为您创建对应目录，不会报错。

   - 容器目录：设置容器和镜像存储目录，建议存储到数据盘。
   - 操作系统：操作系统为集群级别，您可以前往集群详情页进行修改，修改后新增或重装的节点将使用新的操作系统。
   - 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，是比常规密码更安全的登录云服务器的方式，具体详情可参阅 SSH 密钥。
     - 自动生成密码：自动生成的密码，该密码将通过站内信发送给您。
   - 安全组：用于设置云服务器 CVM 的网络访问控制，请根据实际需求进行选择。您还可以单击**新建安全组**，放通其他端口。
6. 单击**完成**。

移出节点

最近更新时间：2022-04-26 10:25:02

操作场景

本文档指导您移出集群下的节点。

注意事项

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

操作步骤

1. 登录容器服务控制台 ，选择左侧导航栏中的 **集群**。
2. 在“集群管理”列表页面，单击需要移出节点的集群 ID/名称，进入该集群详情页。
3. 选择左侧导航栏中的**节点管理** > **节点**，进入“节点列表”页面。
4. 在节点列表中，选择需要移出节点的节点行，单击**移出**。
5. 在弹出的 “您确定要移出以下节点么？” 窗口中，单击**确定**，即可完成移出。

驱逐或封锁节点

最近更新时间：2022-04-18 14:13:36

操作场景

本文档指导您如何驱逐或封锁节点。

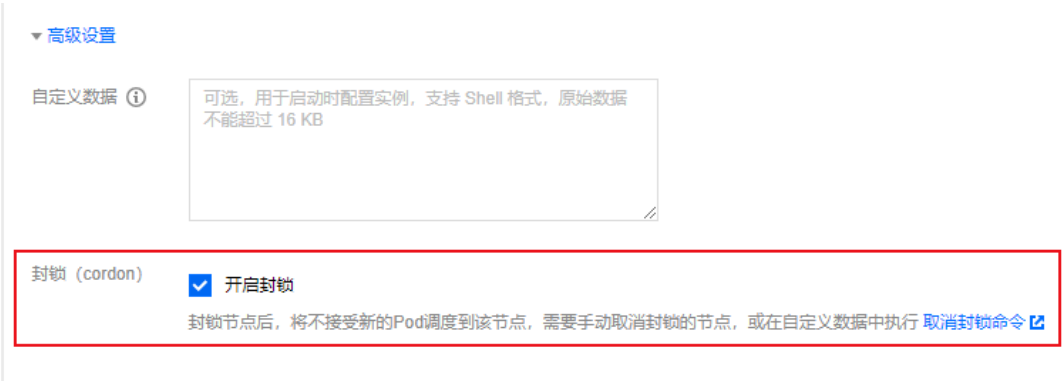
操作步骤

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。封锁节点后，如果节点之前已被 CLB 绑定作为后端目标节点，节点将从目标节点列表中移除。封锁节点有以下两种方法：

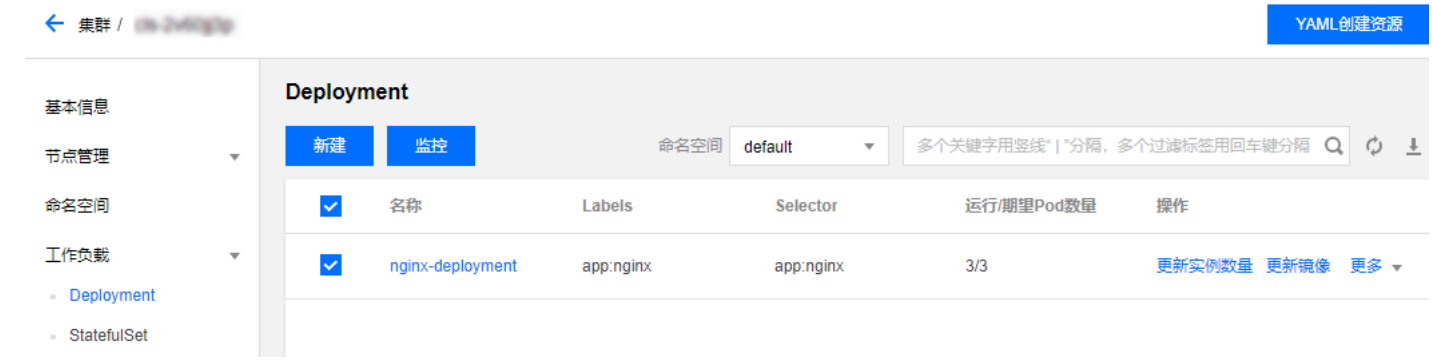
方法一

新增节点 时，在 “云服务器配置” 页面，单击**高级设置**，勾选 “开启封锁” 。



方法二

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击 **集群**，进入集群管理页面。
3. 单击需要封锁节点的集群 ID/名称，进入该集群的管理页面。如下图所示：



4. 在左侧导航栏中，选择 “节点管理” > “节点”，进入 “节点列表” 页面。
5. 在节点列表中，选择需要封锁的节点行，单击**封锁**。如下图所示：

6. 在弹出的对话框中，单击**确定**，即可完成封锁。

取消封锁节点

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

方法一

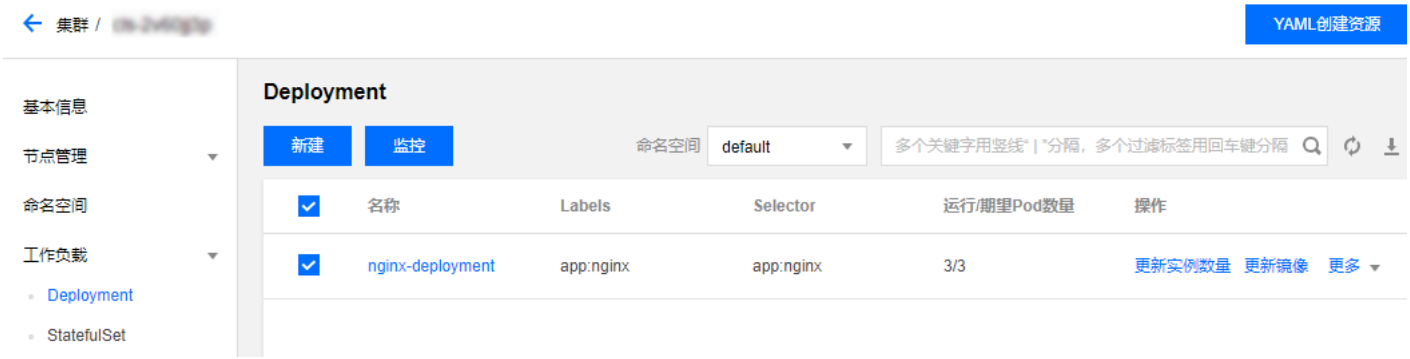
通过执行脚本的方式新增节点时，您可以在该脚本中添加取消封锁节点的命令，即可取消封锁。其示例如下：

```
#!/bin/sh
# your initialization script
echo "hello world!"
# If you set unschedulable when you create a node,
# after executing your initialization script,
# use the following command to make the node schedulable.
node=`ps -ef|grep kubelet|grep -oE 'hostname-override=\S+'|cut -d"=" -f2`
#echo ${node}
kubectl uncordon ${node} --kubeconfig=/root/.kube/config
```

kubectl uncordon 命令即表示取消封锁节点。

方法二

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击 **集群**，进入集群管理页面。
3. 单击需要取消封锁节点的集群 ID/名称，进入该集群的管理页面。如下图所示：



4. 在左侧导航栏中，选择 “节点管理” > “节点”，进入“节点列表” 页面。
5. 在节点列表中，选择需要取消封锁的节点行，单击**取消封锁**。如下图所示：

6. 在弹出的对话框中，单击**确定**，即可完成取消封锁。

驱逐节点

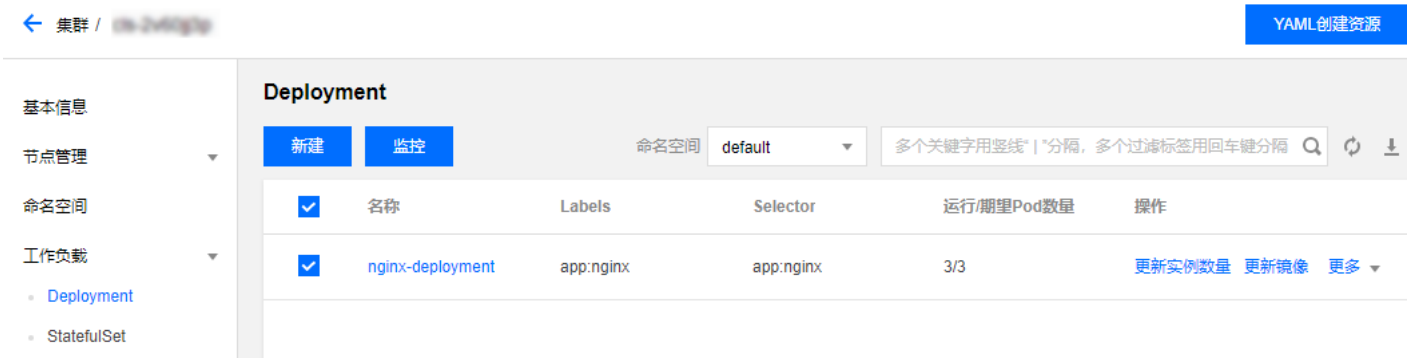
概述

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

> ⚠ 注意：
> 本地存储的 Pod 被驱逐后数据将丢失，请谨慎操作。

操作方法

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击 **集群**，进入集群管理页面。
3. 单击需要驱逐节点的集群 ID/名称，进入该集群的管理页面。如下图所示：



4. 在左侧导航栏中，选择 “节点管理” > “节点”，进入“节点列表” 页面。
5. 在需要驱逐节点的节点行中，单击**更多 > 驱逐**。如下图所示：

6. 在弹出的对话框中，单击**确定**，即可完成驱逐。

设置节点的启动脚本

最近更新时间：2022-06-09 11:23:29

操作场景

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

使用限制

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

- 创建集群 时，在 “云服务器配置” 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 “云服务器配置” 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

添加已有节点 时，在 “云服务器配置” 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

创建伸缩组 时，在 “启动配置” 页面，单击**高级设置**，填写自定义数据，启动脚本。如下图所示：

使用 GPU 节点

最近更新时间：2022-04-18 14:13:30

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，您可以使用腾讯云容器服务支持 GPU 功能，通过该功能可以帮助您快速使用 GPU 容器。
创建 GPU 云服务器有以下多种方式：

- 新建 GPU 云服务器
- 添加已有 GPU 云服务器
- 新建GPU节点池

使用限制

- 添加的节点需要选择 GPU 机型，可根据需求选择自动安装 GPU 驱动，详情可参见 GPU驱动。
- TKE 仅在集群 kubernetes 版本大于1.8.*时支持使用 GPU 调度。
- 默认情况下，容器之间不共享 GPU，每个容器可以请求一个或多个 GPU。无法请求 GPU 的一小部分。
- 当前独立集群的Master节点暂不支持设置为 GPU 机型。

操作步骤

新建 GPU 云服务器

具体操作请参考 新增节点。创建 GPU 机器过程中，请特别关注以下 GPU 的特殊参数：

机型

在 “选择机型” 页面，将 “Node机型” 中的 “机型” 设置为 GPU 机型。

GPU驱动、CUDA版本、cuDNN版本

设置机型后, 可以根据需求选择 GPU 驱动的版本、CUDA 版本、cuDNN 版本。如下图所示：

- 勾选“后台自动安装GPU驱动”，将在系统启动时进行自动安装，预计耗时15-25分钟。
- 支持的驱动版本由 OS 以及 GPU 机型共同决定，详情可参见 GPU 后装驱动版本列表。
- 如果您未勾选“后台自动安装GPU驱动”，为了保证 GPU 机型的正常使用，针对某些低版本 OS，将会为您默认安装 GPU 驱动，完整的默认驱动版本信息可参考下表：

OS名称	默认安装驱动版本
CentOS 7.6、Ubuntu 18、Tencent Linux2.4	450

OS名称	默认安装驱动版本
Centos 7.2 (不推荐)	384.111
Ubuntu 16 (不推荐)	410.79

MIG

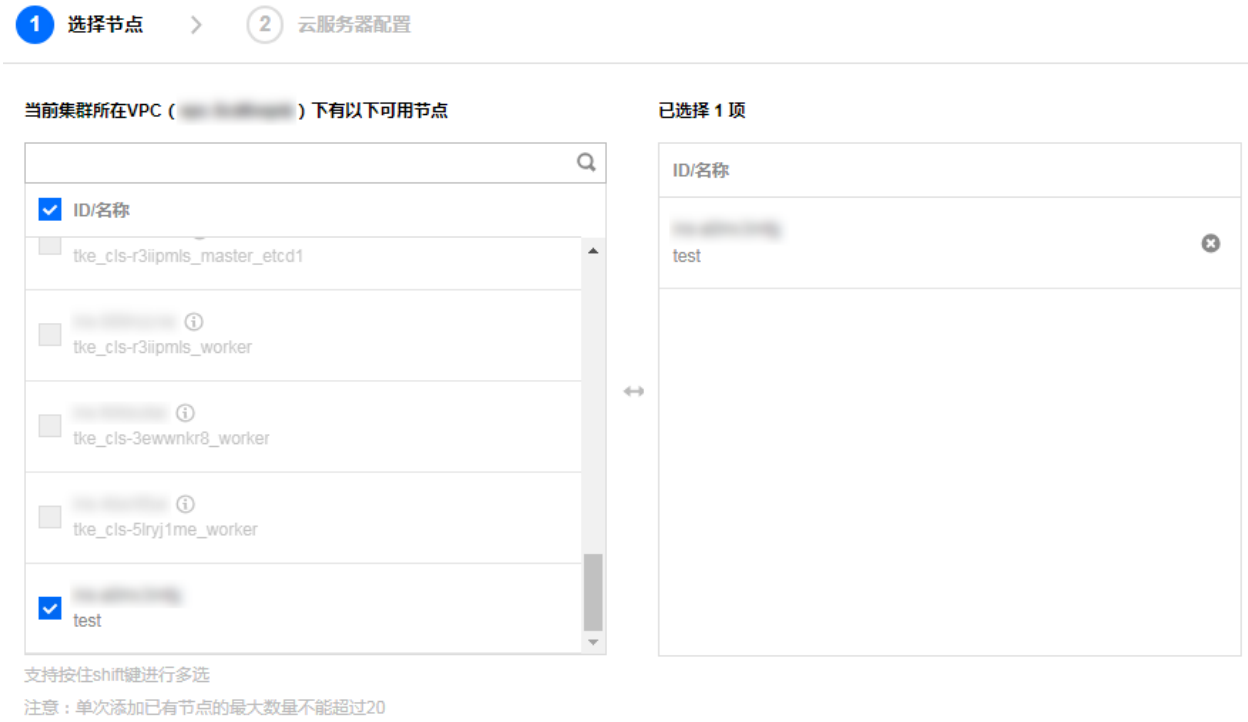
开启 MIG（Multi-Instance GPU）特性后，一颗 A100 GPU 将被划分为七个独立的 GPU 实例，帮助您在多个作业并行的场景下提高 GPU 利用率，详情可参见 NVIDIA 官网指南。

⚠ 使用 MIG 功能，必须满足如下限制：
- GPU 机型为 GT4。
- 在控制台上勾选了 “后台自动安装GPU驱动” 并且配置了 GPU 版本，CUDA 版本和 cuDNN 版本。

添加已有 GPU 云服务器

具体操作请参考 添加已有节点。添加过程中，请注意以下两点：

- 在 “选择节点” 页面，勾选已有的 GPU 节点。如下图所示：



- 按需配置自动安装 GPU 驱动、MIG 等参数。

设置节点 Label

最近更新时间：2022-06-09 11:14:45

操作场景

本文档指导您设置节点 Label。

使用限制

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

操作步骤

控制台设置节点Label

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击 **集群**，进入集群管理页面。
3. 选择需要设置节点 Label 的集群 ID/名称，进入集群详情。
4. 在左侧导航栏中，选择**节点管理 > 节点**，进入“节点列表” 页面。
5. 选择需要设置 Label 的节点行，单击**更多 > 编辑标签**。
6. 在弹出的 “编辑 Label” 窗口中，编辑 Label，单击**确定**。如下图所示：

Kubectl设置节点Label

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

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

3. 执行以下命令，查看节点 Label。

```
kubectl get nodes --show-labels
```

返回类似如下信息：

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

节点池管理
节点池概述

最近更新时间：2022-06-09 11:34:21

简介

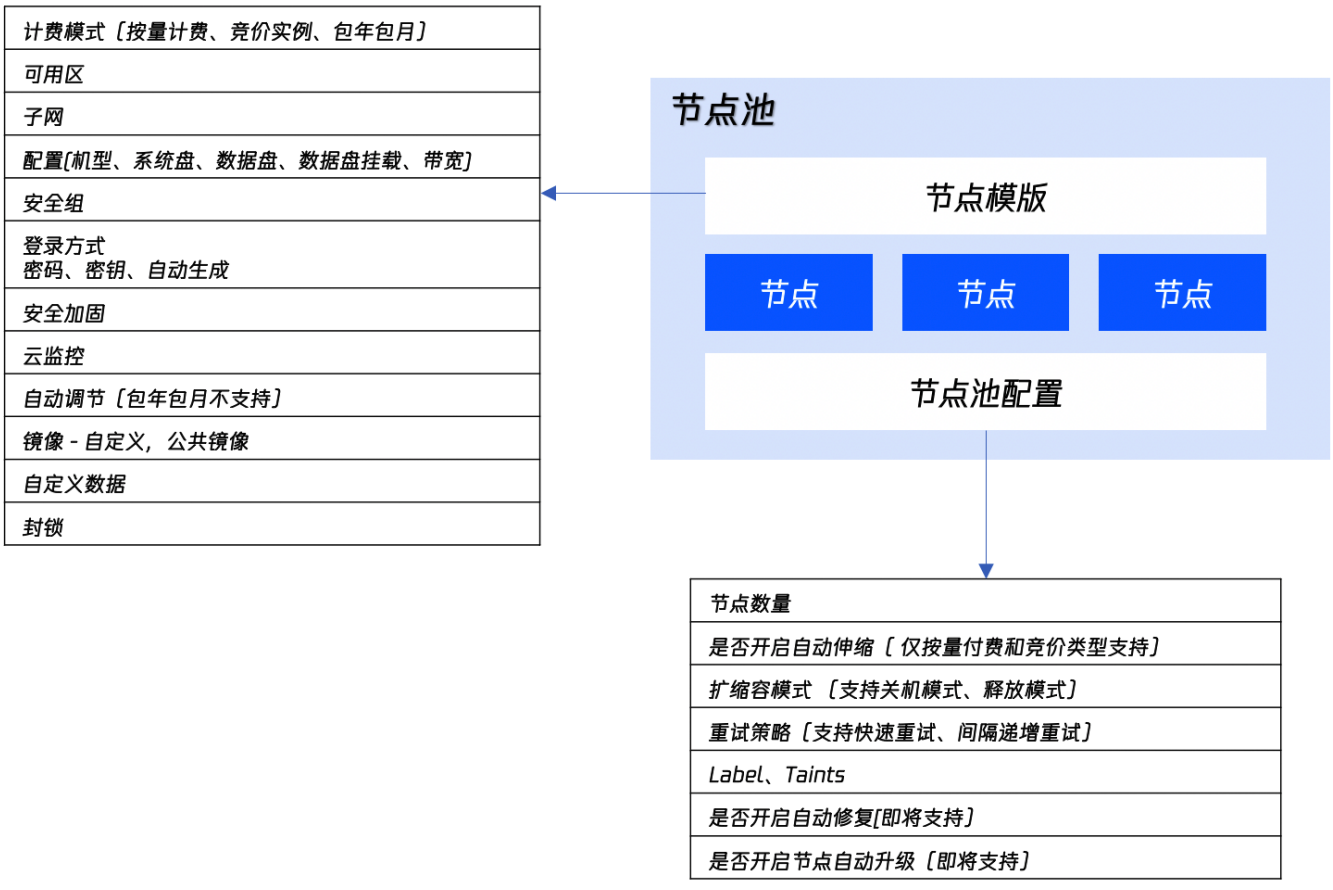
> 说明：
> 节点池功能目前已全量发布。如果您的集群中已创建伸缩组，现在仍可以继续使用。但节点池全量后容器服务 TKE 不会对旧版伸缩组进行新功能迭代，您可以使用 TKE 提供的 从伸缩组创建节点池 API 或 通过容器服务控制台 将伸缩组转换成节点池。
> 同时，除了对存量伸缩组的操作，不建议您在伸缩组入口下进行新建等操作。请通过节点池入口来完成相关操作（节点池已覆盖伸缩组的全部功能）。

为帮助您高效管理 Kubernetes 集群内节点，腾讯云容器服务 TKE 引入节点池概念。借助节点池基本功能，您可以方便快捷地创建、管理和销毁节点，以及实现节点的动态扩缩容：

- 当集群中出现因资源不足而无法调度的实例（Pod）时，自动触发扩容，为您减少人力成本。
- 当满足节点空闲等缩容条件时，自动触发缩容，为您节约资源成本。

产品架构

节点池整体架构图如下所示：



通常情况下，节点池内的节点均具有如下相同属性：

- 节点操作系统。
- 计费类型（目前支持按量计费、竞价实例和包年包月）。
- CPU/内存/GPU。
- 节点 Kubernetes 组件启动参数。
- 节点自定义启动脚本。
- 节点 Kubernetes Label 和 Taint 设置。

此外，TKE 将同时围绕节点池扩展以下功能：

- 支持用 CRD 管理节点池。
- 节点池级别每节点的 Pod 数上限。
- 节点池级别自动修复与自动升级。

应用场景

当业务需要使用大规模集群时，推荐您使用节点池进行节点管理，以提高大规模集群易用性。下表介绍了多种大规模集群管理场景，并分别展示节点池在每种场景下发挥的作用：

场景	作用
集群存在较多异构节点（机型配置不同）	通过节点池可规范节点分组管理。
集群需要频繁扩缩容节点	通过节点池可提高运维效率，降低人力成本。
集群内应用程序调度规则复杂	通过节点池标签可快速指定业务调度规则。
集群内节点日常维护	通过节点池可便捷操作 Kubernetes 版本升级、Docker 版本升级。

相关概念

TKE 的弹性伸缩实现是基于腾讯云弹性伸缩（AutoScaling）以及 Kubernetes 社区的 cluster-autoscaler 实现的。相关概念介绍：

- CA：cluster-autoscaler，社区开源组件，主要负责集群的弹性扩缩容。
- AS：AutoScaling，腾讯云弹性伸缩服务。
- ASG：AutoScaling Group，具体某个节点池（节点池依赖弹性伸缩服务提供的伸缩组，一个节点池对应一个伸缩组，您只需关心节点池）。
- ASA：AS activity，某次伸缩活动。
- ASC：AS config，AS 启动配置，即节点模板。

节点池内节点种类

为了满足不同场景下的需求，节点池内的节点可以分为两个类型。

> 无特殊场景不推荐您使用添加已有节点功能，例如您没有新建节点的权限仅能通过添加已有节点来扩容集群，添加已有节点部分参数可能会与您定义的节点的模板不一致，将无法参与弹性伸缩。

节点类型	节点来源	是否支持弹性伸缩	从节点池移除方式	节点数目是否受**调整数量**影响
伸缩组内节点	弹性扩容或手动调整数量	是	弹性缩容或手动调整数量	是
伸缩组外节点	用户手动加入节点池	否	用户手动移除	否

节点池弹性伸缩原理

在您使用节点池弹性伸缩功能前，请阅读以下原理说明。

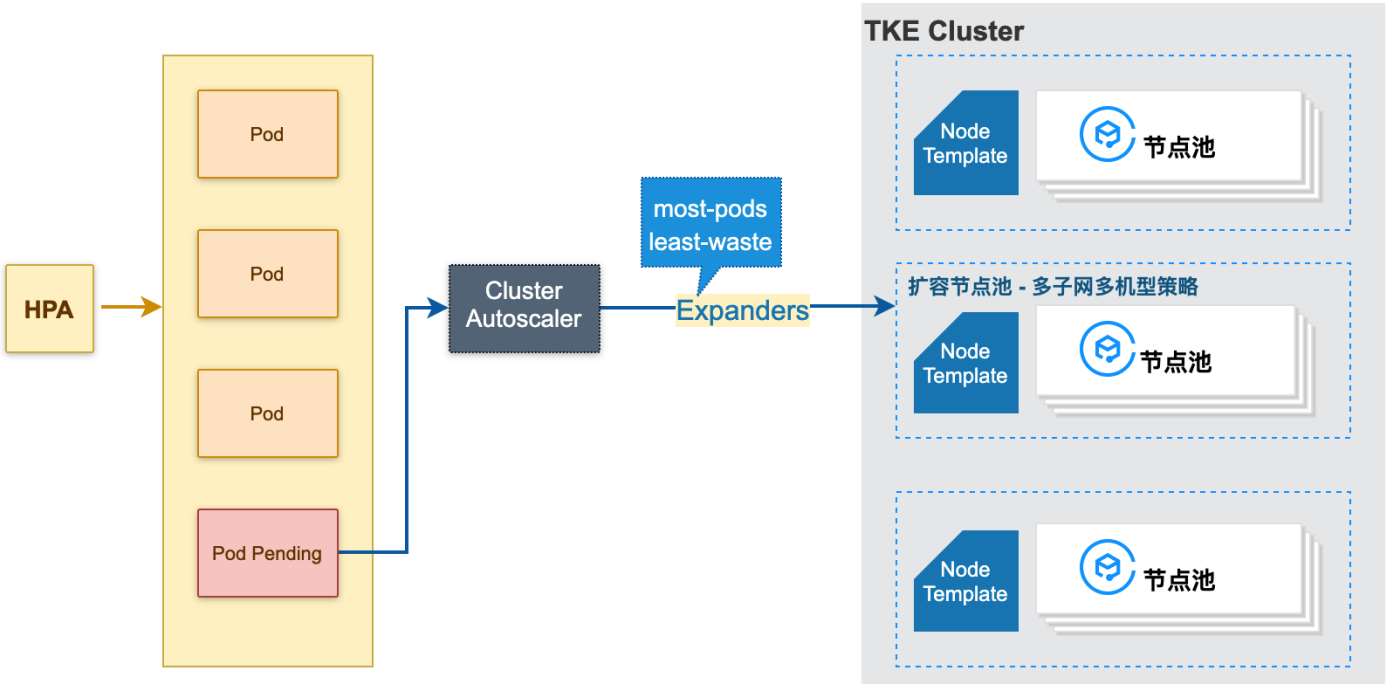
节点池弹性扩容原理

1. 当集群中资源不足时（集群的计算/存储/网络等资源满足不了Pod 的request /亲和性规则），CA（Cluster Autoscaler）会监测到因无法调度而 Pending 的 Pod 。
2. CA 根据每个节点池的节点模板进行调度判断，挑选合适的节点模板。
3. 若有多个模板合适，即有多个可扩的节点池备选，CA 会调用 expanders 从多个模板挑选最优模板并对对应节点池进行扩容。
4. 对指定节点池进行扩容（根据多子网多机型策略），并且提供两种重试策略（可在创建节点池设置），在扩容失败时根据您设定的重试策略进行重试。

> 说明：
> 对特定节点池扩容时，会根据您创建节点池设置的子网以及后续设置的多机型配置来进行扩容。一般情况下会**先保证多机型的策略，后保证多可用区/子网的策略**。

例如您配置了多机型 A、B，多子网1、2、3，会按照 A1、A2、A3、B1、B2、B3 进行尝试，如果A1售罄，会尝试 A2，而不是 B1。

节点池弹性扩容原理如下图所示：



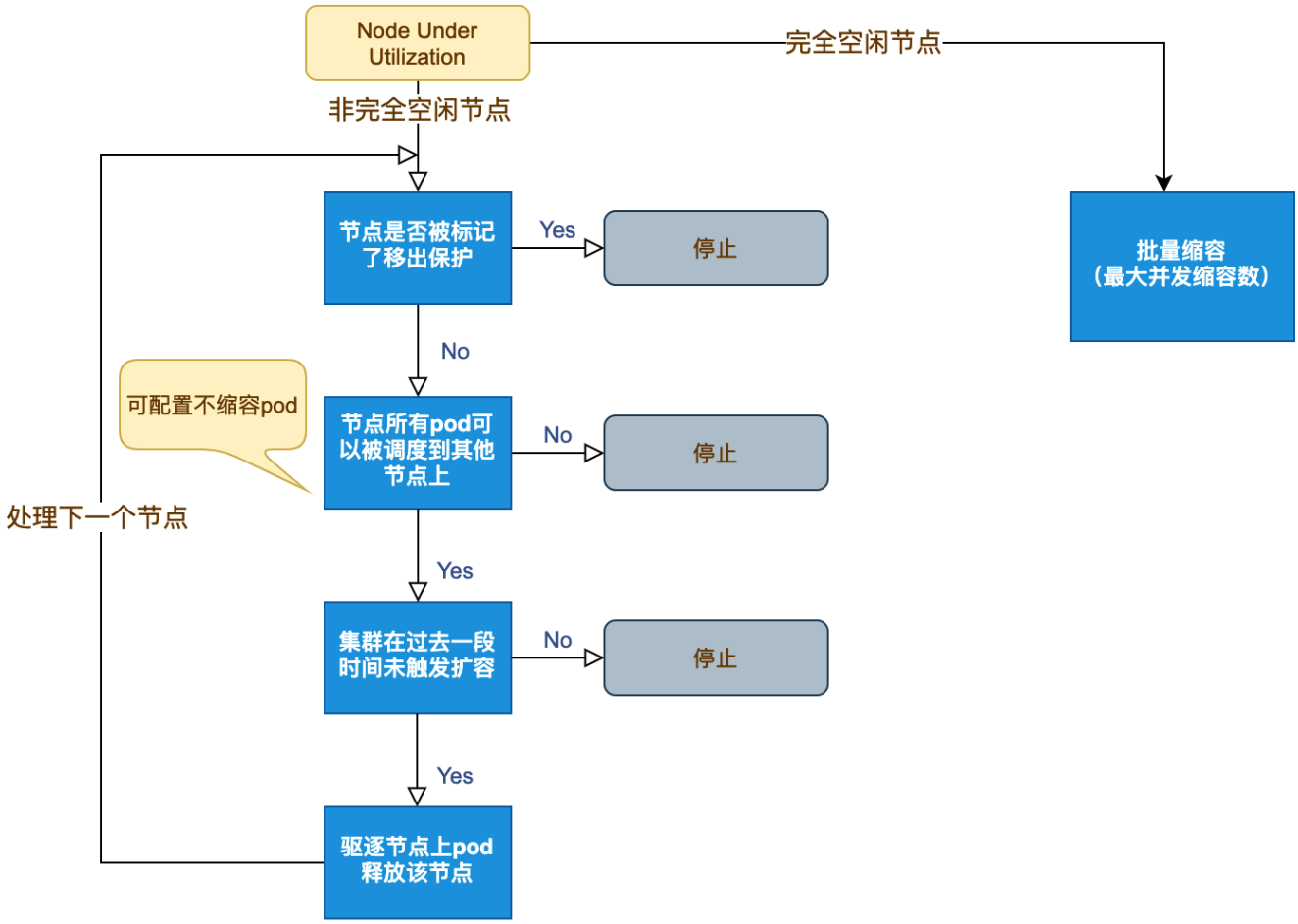
节点池弹性缩容原理

1. CA（Cluster Autoscaler）监测到利用率（取 CPU 利用率和 MEM 利用率的最大值）低于设定的节点。计算利用率时，可以设置 Daemonset 类型不计入 Pod 占用资源。
2. CA 判断集群的状态是否可以触发缩容，需要满足如下要求：
   - 节点空闲时长要求（默认10分钟）。
   - 集群扩容缓冲时间要求（默认10分钟）。
3. CA 判断该节点是否符合缩容条件。您可以按需设置以下**不缩容条件**（满足条件的节点不会被 CA 缩容）：
   - 含有本地存储的节点。
   - 含有 Kube-system namespace 下非 DaemonSet 管理的 Pod 的节点。

   > 说明：
   > 上述不缩容条件在集群维度生效，若您需要更细粒度的保护节点免于缩容，可以使用**缩容保护**功能。

4. CA 驱逐节点上的 Pod 后释放/关机节点（不会处理包年包月节点）。
   - 完全空闲节点可并发缩容（可设置最大并发缩容数）。
   - 非完全空闲节点逐个缩容。

节点池弹性缩容原理如下图所示：



功能点及注意事项

功能点	功能说明	注意事项
创建节点池	新增节点池	• 单个集群不建议超过20个节点池。 • 计费模式为节点池维度属性，包年包月节点池详情可参见创建包年包月节点池说明。请勿将节点池内按量计费节点转成包年包月节点，建议您新建包年包月节点池。
删除节点池	• 删除节点池时可选择是否销毁节点池内节点。 • 无论是否销毁节点，节点都不会保留在集群内。	删除节点池时选择销毁节点，节点将不会保留，后续如需使用新节点可重新创建。
节点池开启弹性伸缩	开启弹性伸缩后，节点池内节点数量将随集群负载情况自动调整。	请勿在伸缩组控制台开启和关闭弹性伸缩。
节点池关闭弹性伸缩	关闭弹性伸缩后，节点池内节点数量不随集群负载情况自动调整。	
调整节点池节点数量	• 支持直接调整节点池内节点数量。 • 若减小节点数量，将按节点移出策略（默认移出最老节点）从伸缩组内缩容节点。请注意：该缩容动作由伸缩组执行，TKE 无法感知具体缩容节点，无提前驱逐/封锁动作。	• 开启弹性伸缩后，不建议手动调整节点池大小。 • 请勿在伸缩组控制台直接调整伸缩组期望实例数。 • 无特殊情况，请勿手动缩容节点池，请使用弹性缩容：弹性缩容时会首先将节点标记为不可调度，随后驱逐或者删除节点上所有 Pod 后再释放节点。
调整节点池配置	可修改节点池名称、操作系统、伸缩组节点数量范围、Kubernetes label 及 Taint。	修改 Label 和 Taint 会对节点池内节点全部生效，可能会引起 Pod 重新调度，请谨慎变更。

功能点	功能说明	注意事项

添加已有节点	• 可添加不属于集群的实例到节点池。要求如下：
	  ◦ 实例与集群属于同一私有网络。
	  ◦ 实例未被其他集群使用且实例与节点池配置相同机型、相同计费模式。
	• 可添加集群内不属于任何节点池的节点，要求节点实例与节点池配置相同机型、相同计费模式。
		无特殊情况时，不建议添加已有节点，推荐直接新建节点池。

移出节点池内节点	支持移出节点池内任意节点，移出时节点可选择是否保留到集群。	请勿在伸缩组控制台往伸缩组内加入节点，可能会导致数据不一致的严重后果。
原伸缩组转换节点池	• 支持存量伸缩组切换为节点池。转化后，节点池完全继承原伸缩组的功能，该伸缩组信息将不再展示。 • 集群内存量所有伸缩组切换完成后，不再提供伸缩组入口。	操作不可逆，请熟悉节点池功能后再进行切换。

相关操作

您可以登录 容器服务控制台 并参考以下文档， 进行对应节点池操作：

- 创建节点池
- 查看节点池
- 调整节点池
- 删除节点池

创建节点池

最近更新时间：2022-06-09 11:34:10

操作场景

本文介绍如何通过容器服务控制台 在集群中创建节点池，并提供了节点池相关操作，例如查看、管理及删除节点池。

前提条件

- 已了解 节点池基本概念。
- 已 创建集群。

说明事项

包年包月节点池

包年包月节点池里的节点为预付费机器，因此在弹性伸缩能力会有一定程度上的限制，说明如下：

- 包年包月节点池支持弹性扩容。
- 包年包月节点池不支持弹性缩容（弹性缩容具备一定的随机性，推荐您通过**手动移出**节点管理预付费机器）。
- 将包年包月节点池内节点移出节点池，可以选择是否保留在集群内，容器服务 TKE **不会为您销毁包年包月节点**，请将节点移出集群后自行前往云服务器控制台销毁，详情可参见 包年包月实例退费说明。

存量伸缩组转换为节点池

容器服务支持集群下原伸缩组转换为节点池。当集群下已创建伸缩组时，可使用存量伸缩组创建节点池。步骤如下：

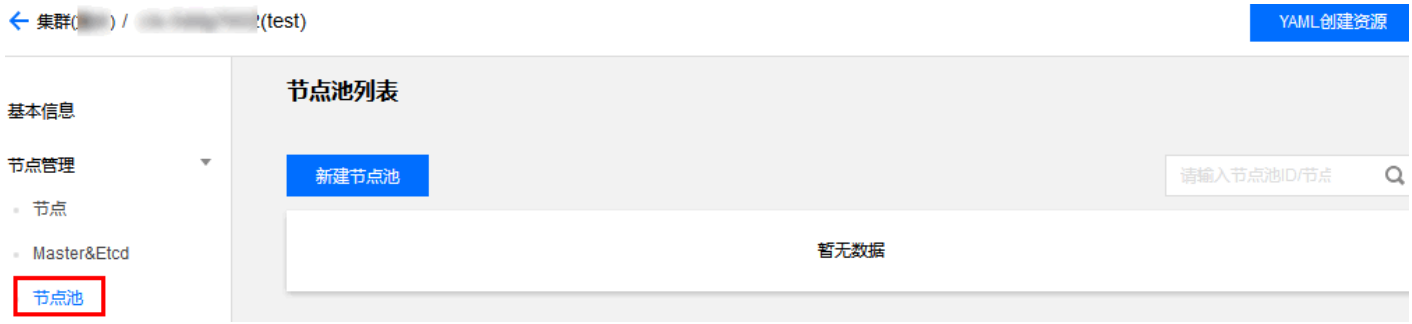
1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “Deployment” 页面。
3. 选择左侧菜单栏中的**节点管理** > **伸缩组**，进入“伸缩组列表”页面。如下图所示：

4. 选择伸缩组所在行右侧的**更多** > **创建节点池**，在弹出窗口中单击**确定**即可。

> 说明：
> 创建完成后，即可参考 查看节点池 查看相关信息。原**伸缩组**项将不可再次查看。

操作步骤

1. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “Deployment” 页面。
2. 选择左侧菜单栏中的**节点管理** > **节点池**，进入“节点池列表”页面。如下图所示：



3. 单击**新建节点池**，进入“新建节点池”页面，参考以下提示进行设置。如下图所示：

- **节点池名称**：自定义，可根据业务需求等信息进行命名，方便后续资源管理。
- **操作系统**：根据实际需求进行选择。该操作系统节点池维度生效，支持更改。更改后新操作系统只对节点池内增量节点生效，不会影响存量节点。
- **计费模式**：提供**按量计费**、**竞价计费**、**包年包月**三种计费模式，请根据实际需求进行选择。详情请参见 计费模式对比。

> ⚠ 注意：
> **包年包月类型的节点池不支持弹性伸缩，可通过手动调整节点池下节点数目来扩容。**

- **支持网络**：系统将为集群内主机分配在节点网络地址范围内的 IP 地址。

> ⚠ 注意：

该选项为集群维度设置项，故不支持修改。

- **机型配置**：单击**请选择机型**，在弹出的“机型配置”窗口中参考以下信息按需选择：
  - **可用区**：启动配置里不包含可用区信息，该选项仅用于过滤所选可用区下可用实例类型。
  - **机型**：支持通过 CPU 核数、内存大小及实例类型进行筛选。详情请参见 实例规格 和 快速入门 Linux 云服务器。
  - **系统盘**：存储控制、调度云服务器运行的系统集合。支持查看所选机型的可选系统盘类型，请参考 云硬盘类型 并根据实际需求进行选择。
  - **数据盘**：用于存储所有的用户数据。请根据以下指引进行设置。每种机型所对应的数据盘设置不尽相同，请参考以下表格进行设置：

机型	数据盘设置
标准型、内存型、计算型、GPU 机型	默认不勾选。若勾选，请根据实际情况进行云硬盘设置及格式化设置。
高 IO 型、大数据型	默认勾选且不可更改，支持对默认购买的本地盘进行自定义格式化设置。
批量型	默认勾选且支持取消勾选，勾选时仅支持购买默认本地盘，支持对默认本地盘进行自定义格式化设置。

  - **添加数据盘（可选）**：单击**添加数据盘**，并参考上表进行设置。
    - **公网宽带**：默认勾选**分配免费公网IP**，系统将免费分配公网 IP。支持按使用流量、按带宽计费两种模式，请参考 公网计费模式 根据实际情况进行选择，并进行网速自定义设置。
- **登录方式**：提供以下三种登录方式，请根据实际情况进行选择。
  - **立即关联密钥**：密钥对是通过算法生成的一对参数，是一种比常规密码更安全的登录云服务器的方式。详情请参见 SSH 密钥。
  - **SSH密钥**：该配置项仅在选择**立即关联密钥**登录方式时出现，在下拉框中选用已有密钥即可。若需新建，请参考 创建 SSH 密钥。
  - **自动生成密码**：自动生成的密码将通过 站内信 发送给您。
  - **设置密码**：请根据提示设置对应密码。
- **安全组**：默认为创建集群时所设置的安全组，可根据实际需要进行更换或添加。
- **数量**：对应期望实例数量，请根据实际需求进行设置。

> 注意：
> 若节点池已开启自动伸缩， 该数量将会随集群负载自动调整。

- **节点数量范围**：节点数量将在设定的节点范围内自动调节，不会超出该设定范围。
- **支持子网**：请根据实际需求选择合适的可用子网。

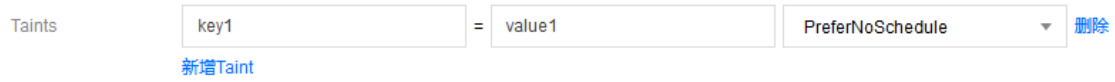
> 说明：
> 节点池默认的多子网扩容策略如下：当您配置了多个子网，节点池扩容时（手动扩容及弹性扩容）将按照子网列表的顺序，作为优先级来尝试创建节点，如果优先级最高的子网可以创建成功，则总在该子网创建。

4.（可选）单击**更多设置**，查看或配置更多信息。如下图所示：

- **CAM角色**：可为节点池的所有节点绑定相同的 CAM 角色，从而赋予节点该角色绑定的授权策略。详情请参见 管理实例角色。
- **容器目录**：勾选即可设置容器和镜像存储目录，建议存储到数据盘。例如 /var/lib/docker。
  - **安全加固**：默认免费开通 DDoS 防护、WAF 和云镜主机防护，详情请参见 T-Sec 主机安全官网页。
  - **云监控**：默认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，详情请参见 云监控 CM 官网主页。
  - **弹性伸缩**：默认勾选**开启**。
  - **封锁初始节点**：勾选**开启封锁**后，将不接受新的 Pod 调度到该节点，需要手动取消封锁的节点，或在自定义数据中执行 取消封锁命令，请按需设置。
  - **Label**：单击**新增Label**，即可进行 Label 自定义设置。该节点池下所创建的节点均将自动增加此处设置的 Label，可用于后续根据 Label 筛选、管理节点。
  - **Taints**：节点属性，通常与 Tolerations 配合使用。此处可为节点池下的所有节点设置 Taints，确保不符合条件的 Pod 不能够调度到这些节点上，且这些节点上已存在的不符合条件的 Pod 也将会被驱逐。

> 说明：
> Taints 内容一般由 key 、value 及 effect 三个元素组成。其中 effect 可取值通常包含以下三种：
>
> - **PreferNoSchedule**：非强制性条件，尽量避免将 Pod 调度到设置了其不能容忍的 taint 的节点上。
> - **NoSchedule**：当节点上存在 taint 时，没有对应容忍的 Pod 一定不能被调度。
> - **NoExecute**：当节点上存在 taint 时，对于没有对应容忍的 Pod，不仅不会被调度到该节点上，该节点上已存在的 Pod 也会被驱逐。

以设置 Taints key1=value1:PreferNoSchedule 为例，控制台配置如下图所示：



- **重试策略**：提供以下两种策略，请根据实际需求进行选择。
  - **快速重试**：立即重试，在较短时间内快速重试，连续失败超过一定次数（5次）后不再重试。
  - **间隔递增重试**：间隔递增重试，随着连续失败次数的增加，重试间隔逐渐增大，重试间隔从秒级到1天不等。
- **扩缩容模式**：提供以下两种扩缩容模式，请根据实际需求进行选择。
  - **释放模式**：缩容时自动释放 Cluster AutoScaler 判断的空余节点， 扩容时自动创建新的节点加入到伸缩组。
  - **关机模式**： 扩容时优先对已关机的节点执行开机操作，节点数依旧不满足要求时再创建新的节点。缩容时将关机空余节点，若节点支持关机不收费则将不收取机型的费用，详情请参见 按量计费实例关机不收费说明 ，其余节点关机会继续收取费用。
- **自定义数据**：指定自定义数据来配置节点，即当节点启动后运行配置的脚本。需确保脚本的可重入及重试逻辑，脚本及其生成的日志文件可在节点的 /usr/local/qcloud/tke/userscript 路径查看。

5. 单击**创建节点池**即可创建节点池。

相关操作

节点池创建完成之后，您可参考以下操作指引进行后续的节点池管理：

- 查看节点池
- 调整节点池
- 删除节点池

查看节点池

最近更新时间：2022-06-09 11:33:56

操作场景

本文介绍如何通过容器服务控制台 查看集群中已创建的节点池，并获取节点池的详细信息，以便后续对节点池进行管理。

前提条件

集群下已 创建节点池。

操作步骤

查看节点池列表页

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “Deployment” 页面。
3. 选择左侧菜单栏中的**节点管理** > **节点池**，进入“节点池列表”页面。即可查看节点池全局配置及已创建的节点池。如下图所示：

节点池信息及配置如下：

- **全局配置**：包含该集群下所有节点池的共同配置项。可单击该模块右上角**编辑**进行修改，详情请参见 调整节点池全局配置。
  - **自动缩容**：本例此处已关闭。正常开启时，集群中节点空闲资源较多时将触发缩容。详情请参见 集群自动扩缩容说明。
  - **扩容算法**：本例此处默认为“随机”，表示节点池将随机选择一个伸缩组进行扩容。容器服务还支持以下两种扩容算法，您可根据实际需求进行更改：
    - **most-pods**：选择能调度更多 Pod 的伸缩组进行扩容。
    - **least-waste**：选择 Pod 调度后资源剩余更少的伸缩组进行扩容。
  - **集群规模上限**：展示当前集群规模信息。对已有节点池进行数量调整或再次新建节点池时，请注意参考此处规模限制，合理设置节点池的节点数量。
- **节点池名片页**：全局配置下方即为节点池排列区域，每个节点池以卡片的形式进行展示，主要包含以下信息：

> 说明：
> 当节点池较多时，可在该区域右上角的搜索框中输入节点池 ID 或节点池名称进行筛选。

- **节点池 ID（节点池名称）**：本例为 np-***(test)，单击此 ID 可进入该节点池详情页，查看更多节点池相关信息。
- **节点池状态**：本例为“正常”，表示该节点池处于正常状态。
- **节点池操作**：包含**编辑**、**调整数量**、**更多**等，详情请参见 调整节点池配置。
- **该节点池下可用节点数/节点总数**：本例为“1可用/共1台”。
- **机型**：展示该节点池下所有节点的机型。
- **计费模式**：展示该节点池下所有节点的计费模式，本例为“按量计费”，表示按照实例的实际使用量进行收费。更多计费详情请参见 计费模式 。
- **弹性伸缩**：本例为“已启用”。

查看单个节点池

1. 在“节点池名片页”中，单击目标节点池 ID。如下图所示：

2. 进入该节点池详情页，即可查看节点池更多基本信息及节点信息。如下所示：

相关操作

您可参考以下文档，了解更多节点池具备的功能：

- 创建节点池
- 调整节点池
- 删除节点池

调整节点池

最近更新时间：2022-06-09 11:33:45

操作场景

本文介绍如何通过容器服务控制台 调整节点池配置。包含调整节点池全局配置、节点池配置、节点池下节点数量及启用或停用弹性伸缩、为节点设置缩容保护操作。

前提条件

- 已创建可用节点池。详情请参见 创建节点池。
- 已进入节点池列表。详情请参见 查看节点池。

操作步骤

调整节点池全局配置

1. 在“节点池列表”页面，单击“全局配置”模块右上角的**编辑**。如下图所示：

2. 在弹出的“设置集群伸缩全局配置”窗口中，参考以下信息进行设置。如下图所示：

主要参数信息如下：

- **自动缩容**：默认不勾选。开启自动缩容时，集群中节点空闲资源较多时将触发缩容。详情请参见 集群自动扩缩容说明。
- **缩容配置**：该配置项仅在开启自动缩容时显示，请根据实际需求进行设置。
  - **最大并发缩容数**：该数值表示为可以同时进行缩容的节点数，此处默认为“10”，可按需自定义设置。

> ⚠ 注意：

> 此处只缩容完全空闲的空节点。如果节点上存在 Pod，则每次缩容最多一个节点。

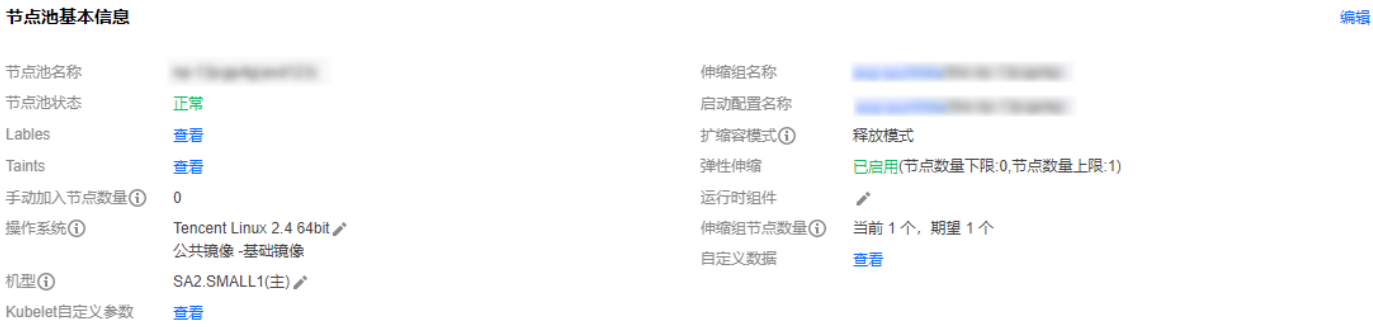
- **Pod 占用资源/可分配资源小于的值**：可设置 **Pod 占用资源/可分配资源**在占比小于设定值时开始判断缩容条件。占比值范围需确保在0 – 80之间。
- **节点连续空闲**：可自定义设置节点连续空闲时间超过几分钟之后会被缩容。
- **集群扩容**：可自定义设置集群首次判断扩容条件的时间点。
- **不缩容节点**：请根据实际需求勾选以下配置项，确保不缩容以下特定类型的节点。
  - 含有本地存储 Pod 的节点。
  - 含有 kube-system namespace 下非 DaemonSet 管理的 Pod 的节点。
- **扩容算法**：集群扩容时所依赖的算法准则，提供以下三种选择：
  - **随机**：有多个节点池时，随机选择一个节点池进行扩容。
  - **most-pods**：有多个节点池时，选择能调度更多 Pod 的节点池进行扩容。
  - **least-waste**：有多个节点池时，选择 Pod 调度后资源剩余更少的节点池进行扩容。

3. 单击**确定**，即可设置成功。

调整节点池配置

调整节点池操作系统、备选机型、容器运行时

1. 在“节点池列表”页面，单击节点池 ID，进入节点池详情页。
2. 在节点池基本信息页，可对节点池属性进行更改。如下图所示：



操作系统

单击“操作系统”右侧的 ，即可更改节点池的操作系统。
更改操作系统仅决定节点池内新增或者重装升级节点的操作系统，不影响正在运行节点的操作系统。如下图所示：

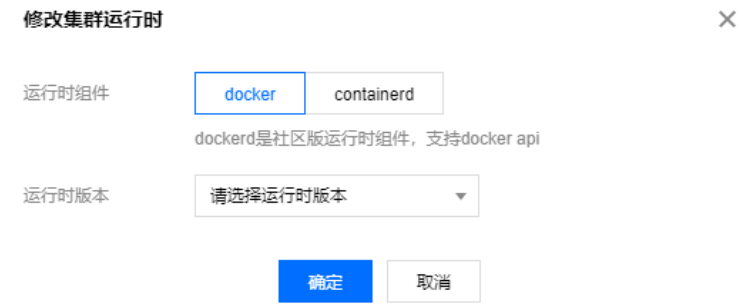
机型

单击“机型”右侧的，即更改节点池的备选机型（主机型不可更改）。设置备选机型可有效降低由于主机型售罄导致扩容失败的风险。如下图所示：

- 备选机型顺序对应该机型的优先级顺序，请根据需要设置机型顺序，您可以通过弹窗最下方展示的机型顺序进行确认。
- 备选机型必须与主机型规格（CPU、内存、CPU 架构）相同。
- 同一节点池最多只可选择10种机型（包含主机型），请按需自行规划。

运行时组件

单击“运行时组件”右侧的，即可更改节点池的运行时组件以及版本，详情请参见 如何选择运行时组件。如下图所示：



调整节点数量范围、Label、Taints

1. 单击目标节点池名片页右上角的**编辑**。如下图所示：

2. 在弹出的“调整节点池配置”页面，参考以下信息进行设置。如下图所示：

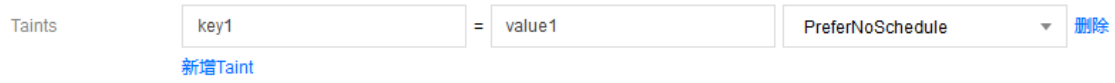
- **节点池名称**：自定义。可根据业务需求等信息进行命名，方便后续资源管理。
- **弹性伸缩**：根据实际需求进行勾选。
- **节点数量范围**：节点数量将在设定的节点范围内自动调节，不会超出该设定范围。

> **注意：**
> 该数量范围的设置，将影响 调整节点池下节点数量 操作。例如，当前节点池的节点数量已达到该范围最大值时，节点数量将不再支持上调。

- **Label**：该节点池下所创建的节点将自动加上此处设置的 Label，方便后续根据 Label 筛选、管理节点。单击**新增Label**，即可进行 Label 自定义设置。
- **Taints**：节点属性，通常与 Tolerations 配合使用。此处可为节点池下的所有节点设置 Taints，确保不符合条件的 Pod 不能够调度到这些节点上，且这些节点上已存在不符合条件的 Pod 也将会被驱逐。

> **说明：**
> Taints 内容一般由 key 、value 及 effect 三个元素组成。其中 effect 可取值通常包含以下三种：
>
> - **PreferNoSchedule**：非强制性条件，尽量避免将 Pod 调度到设置了其不能容忍的 taint 的节点上。
> - **NoSchedule**：当节点上存在 taint 时，没有对应容忍的 Pod 一定不能被调度。
> - **NoExecute**：当节点上存在 taint 时，对于没有对应容忍的 Pod，不仅不会被调度到该节点上，该节点上已存在的 Pod 也会被驱逐。

以设置 Taints key1=value1:PreferNoSchedule 为例，控制台配置如下图所示：



3. 单击**确定**并等待更新完成即可。

调整节点池下节点数量

1. 单击目标节点池名片页右侧的**调整数量**。如下图所示：

2. 在弹出的“调整数量”页面，按需调整节点数量，该数量必须落在设置的节点池数量范围内。如下图所示：

> 说明：
> 节点池已开启弹性伸缩时，该数量将会随着集群工作负载自动调整，可能会存在最终实际的节点数量与数量调整时所设置的值不一致的问题。

3. 单击**确定**等待数量调整完成即可。

启用或停用弹性伸缩

> 说明：
> 执行启用/停用弹性伸缩操作时，仅建议在容器服务侧节点池处进行，以确保该状态能够同步至 Cluster-autoscaler。

1. 单击目标节点池名片页右上角的**更多**。如下图所示：

2. 结合实际情况选择**启用弹性伸缩**或者**停用弹性伸缩**，并在弹出的窗口中单击**确认**即可。

相关操作

您可参考以下文档，了解更多节点池功能及操作：

- 创建节点池
- 查看节点池
- 删除节点池

删除节点池

最近更新时间：2022-06-09 11:37:23

操作场景

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

前提条件

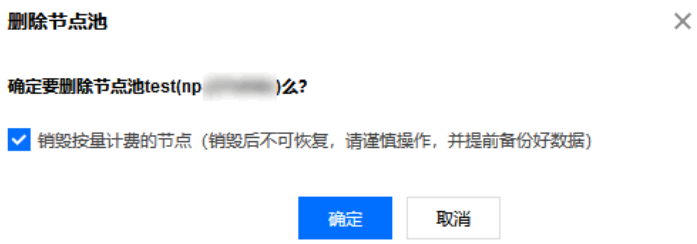
- 已创建可用节点池。详情请参见 创建节点池。
- 已进入“节点池列表”页面。详情请参见 查看节点池。

操作步骤

1. 选择目标节点池名片页右上角的**更多** > **删除**。如下图所示：

2. 在弹出的“删除节点池”窗口中，按需设置是否保留节点。如下图所示：

> ⚠ 注意：
>
> - 默认勾选销毁按量计费的节点，可根据实际需求取消勾选。
> - 按量计费的节点销毁后不可恢复，请谨慎操作，并提前备份好数据。



3. 单击**确认**，等待删除成功即可。

相关操作

您可参考以下文档，了解节点池更多功能及操作：

- 创建节点池
- 查看节点池
- 调整节点池

查看节点池伸缩记录

最近更新时间：2022-02-16 14:58:53

操作场景

本文介绍如何查看节点池的伸缩记录，适用于以下场景：

- 您可以通过伸缩活动了解自己业务的流量变化，更有效的按需配置节点池。
- 您可以通过节点池内节点扩缩容活动来了解自己的花费来源，进行更高效的成本管理。
- 您可以了解扩缩容活动失败的原因（例如扩容时由于地域资源售罄导致扩容失败），进行风险管理。
- 您可以查看两个层级的伸缩记录：全局伸缩记录 和 特定节点池伸缩记录。

> 说明：
>
> - 在存在多个节点池的情况下，CA（Cluster Autoscaler）负责选择合适的节点池进行扩缩容，全局伸缩记录可以从 CA 的 Event 得到。
> - 如果您只关心特定节点池的伸缩记录，不关心 CA 的行为，可进入节点池详情页查看该节点池的扩缩容活动记录。

前提条件

- 已创建可用节点池。详情请参见 创建节点池。
- 已进入“节点池列表”页面。详情请参见 查看节点池。

操作步骤

查看全局伸缩记录

社区开源组件 CA 会把任何一次扩缩活动的相关信息，以 Kubernetes event 的形式存储到特定的 Pod 或者 Node 下，但存在 Kubernetes events 资源默认后端只存储1小时的限制。若您想对节点池的扩缩记录进行查询及复盘，建议您开启集群的事件持久化功能，对 Kubernetes Events 进行持久存储。

开启事件持久化

> 说明：
>
> 该步骤为新版事件持久化设置步骤，旧版事件持久化设置步骤请参见 事件存储。

1. 登录 腾讯云容器服务控制台。
2. 选择左侧导航栏中的**运维功能管理**，单击目标集群所在行右侧的**设置**。

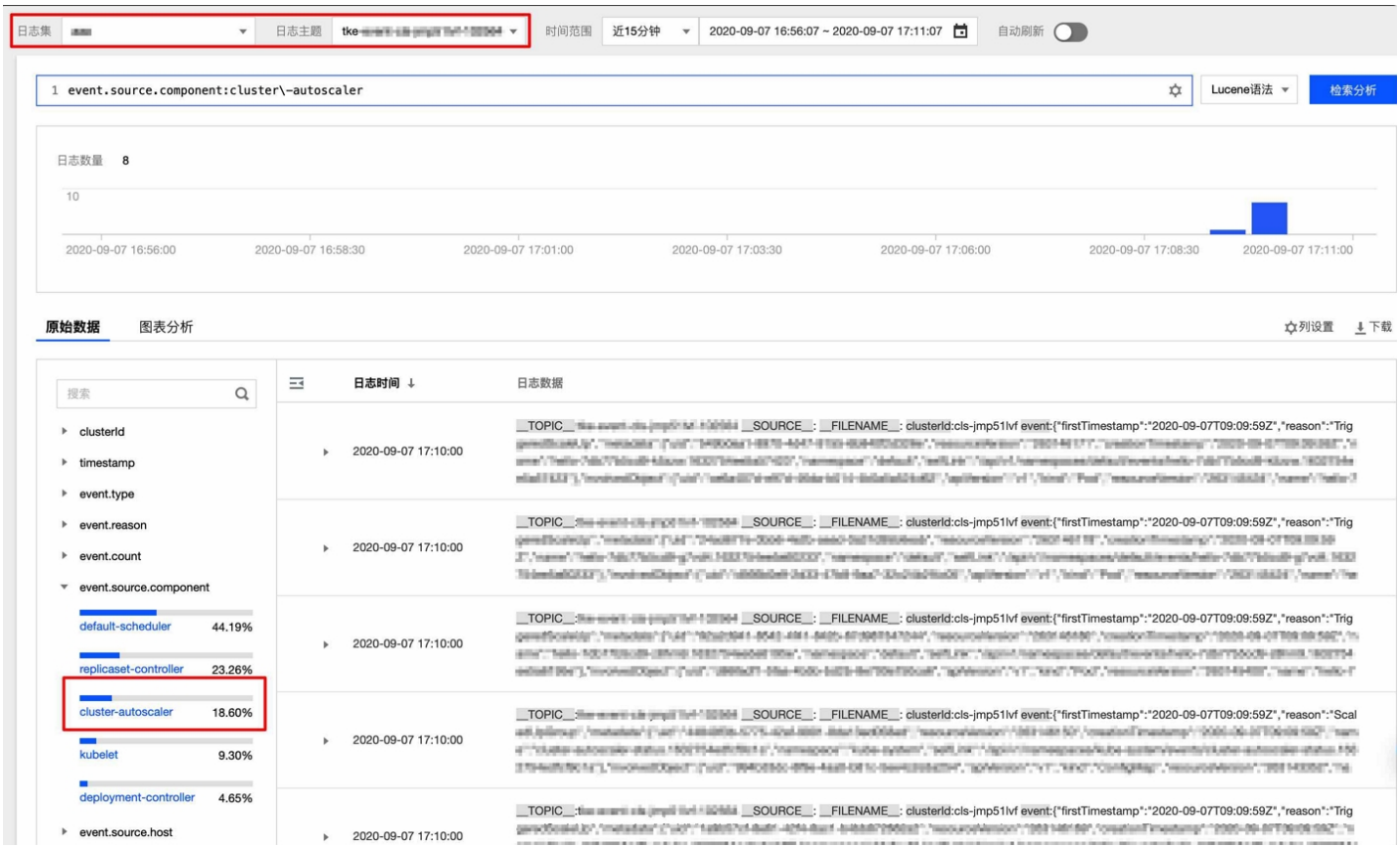
3. 在弹出的“设置功能”窗口中，选择“事件存储”功能右侧的**编辑**，勾选“开启事件存储”，并创建或者选择已有的日志主题。如下图所示：

4. 单击**确定**即可开启事件持久化功能。

查看事件持久化

1. 登录 腾讯云日志服务控制台。
2. 选择左侧导航栏中的**检索分析**，进入“检索分析”管理页面。
3. 在“检索分析”页面上方选择地域，选择希望查看事件持久化的日志集和日志主题。

4. 勾选**event.source.component:cluster-autoscaler**，单击**检索分析**。如下图所示：



5. 在右侧的**版面设置**可配置数据列，对关注的列进行可视化。

检索指引

您可参考以下文档，查看更具体的扩缩容活动列表：

- CLS 检索语法
- CA FAQ
- CA 扩缩容 Event 的 Reason 字段可能有如下取值：TriggeredScaleUp、NotTriggerScaleUp、ScaledUpGroup、FailedToScaleUpGroup、ScaleDown、ScaleDownFailed、ScaleDownEmpty。详情请参见 字段详细介绍。

查看特定节点池伸缩记录

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “Deployment” 页面。
3. 选择左侧菜单栏中的**节点管理 > 节点池**，进入“节点池列表”页面。
4. 在“节点池名片页”中，单击目标节点池 ID。如下图所示：

5. 进入该节点池详情页，选择顶端**伸缩记录**页签，即可查看伸缩记录。如下图所示：

伸缩记录展示字段如下：

- **活动ID**：伸缩活动ID。
- **状态**：伸缩活动的状态。
- **描述**：此次伸缩活动的描述，显示扩容机器数/缩容机器数。
- **活动起因**：触发此次伸缩活动的原因，例如“因匹配期望实例数”。
- **失败原因**：如果伸缩活动**状态**为失败，该栏会显示伸缩活动的失败原因。
- **开始时间**：伸缩活动开始的时间，精确到秒。
- **结束时间**：伸缩活动结束的时间，精确到秒。

相关操作

您可参考以下文档，了解节点池更多功能及操作：

- 创建节点池
- 查看节点池
- 调整节点池

超级节点管理
超级节点概述

最近更新时间：2022-06-17 16:58:13

什么是超级节点?

超级节点是腾讯云全新升级的节点产品形态，向用户提供可用区级别的、支持自定义规格的节点能力，使用超级节点类似于使用一台超大规格的 CVM，资源管理和资源扩缩容都更简单，超级节点支持**包年包月**和**按量计费**两种计费模式。

- 包年包月模式下，由用户自定义节点规格，用户可包年包月购买自定义总规格的节点算力来实现预付费，包年包月模式的节点限制用户调度规模在 1C 至 8C 间、CPU 内存比值小于 1:4 的 Pod。包年包月模式适用于固定算力的在线常驻业务，包年包月的超级节点相较于普通节点单核价格更低，用户可将符合规则的 Pod 全部迁移至超级节点，来降低固定资源的单核成本。
- 按量计费模式下，用户无需指定节点规格，可弹性使用节点资源，使用后按节点内实际使用的 CPU 内存按量计费。按量计费模式适用于弹性算力场景，用户可添加按量计费的超级节点，在业务高峰时使用弹性资源，进一步降低集群资源成本。

部署在超级节点上的 Pod 具备云服务器一致的安全隔离性，具备与部署在集群既有普通节点上的 Pod 一致的网络隔离性和网络联通性。

> 说明：
> 超级节点包年包月模式现火热内测中，可 提交工单 申请购买，还可参与 618新购活动，享受一折购买包年包月超级节点代金券福利，首月体验超值划算。

产品优势

超级节点相较于常规节点有如下产品优势：

成本节省

购买包年包月的超级节点，相较于购买普通节点单核价格更低。

- CPU、内存定价更低，详情请参考 超级节点价格说明。
- 包年包月的超级节点规格上限更大，使 Pod 可调度域更广，避免了购买包年包月普通节点时边界资源的浪费。

按量计费的超级节点由于具备秒级弹性的优势及按需使用的产品形态，使其在弹性业务场景的成本节省方面也具有很大的优势。

- 按需使用，减少集群的资源的 buffer。按量计费的超级节点是真正的按需使用，避免了碎片资源的产生，提升整体集群的资源利用率，通过减少资源 buffer，降低成本。
- 减少弹性资源的计费时长，节省成本。由于按量计费的超级节点是秒级扩容，瞬时缩容，因此会大幅降低在扩缩容过程中产生的计费成本。

管理简单

从在一个可用区需运维和管理多个节点变为一个可用区仅需管理 1 个节点，资源管理更容易。
包年包月超级节点支持随时升配，支持部分降配，像管理一个超规格的 CVM 节点一样，对节点进行升降配即可轻松实现常驻资源的扩缩容。弹性资源可通过添加按量计费的超级节点，弹性能力相较于常规节点池及伸缩组更快更高效，在更好满足弹性诉求的基础上，节省用户成本。

节点管理对比：

场景	常规节点	超级节点
购买	需详细规划不同节点类型、规格、数量，分多次购买	仅需统计可用区下资源总规格并购买 1 个自定义规格的超级节点。
资源扩容	需要新添加节点或批量调整单个节点的规格	• 长期扩容：升级超级节点配置，调整总规格大小。 • 短期扩容：使用按量计费超级节点，弹性部分默认调度至按量计费的超级节点。
资源缩容	需要退还节点并承担损失	• 长期缩容：降级超级节点配置，调整总规格大小（单月支持降配 1 次），支持降配 3 个超级节点。 • 短期缩容：使用按量计费超级节点后无需处理。
节点管理	需管理多个节点	只需按可用区维度管理单个节点。

弹性更快更高效

相比节点池及伸缩组，按量计费的超级节点的扩容、缩容流程简化了购买、初始化、退还服务器的流程，大幅提升了弹性的速度，尽可能降低在扩容流程中可能出现的失败，使得弹性更加的高效。

- 对于扩容，按量计费超级节点将 4~6 分钟的常规扩容流程缩短至秒级，扩容流程更高效。
- 对于缩容，按量计费超级节点的缩容流程短规避了 CA 流程、封锁流程及 Pod 驱逐流程，完全做到了无损缩容，瞬时缩容。

计费模式

超级节点支持按量计费和包年包月两种计费模式。

实例计费模式	包年包月	按量计费
付款方式	预付费	购买时 冻结费用，每小时结算
计费单位	元/月	元/秒
单价	单价较低	单价较高
最少使用时长	至少使用一个月	按秒计费，按小时结算，随时购买随时释放
配置调整	支持升降配置	无规格限制
使用场景	适用于算力需求量长期稳定的成熟在线业务	适用于算力需求量瞬间大幅波动的场景

说明：

- 添加包年包月的超级节点，将按照节点总规格、CPU 内存单价及购买时长计算总价格进行预付费。
- 添加按量计费的超级节点，本身不收取任何费用，将根据实际调度到超级节点上的 Pod 资源来计费。后台根据工作负载申请的 CPU、GPU、内存数值以及工作负载的运行时间来核算具体费用，用户无需提前支付费用。

地域和可用区

用户可在以下可用区内使用包年包月的超级节点：

中国

地域	可用区
华南地区（广州） ap-guangzhou	广州三区 ap-guangzhou-3
	广州四区 ap-guangzhou-4
	广州六区 ap-guangzhou-6
	广州七区 ap-guangzhou-7
华东地区（上海） ap-shanghai	上海二区 ap-shanghai-2
	上海三区 ap-shanghai-3
	上海四区 ap-shanghai-4
	上海五区 ap-shanghai-5
华东地区（南京）	南京一区

ap-nanjing	ap-nanjing-1
	南京二区 ap-nanjing-2
	南京三区 ap-nanjing-3
华北地区（北京） ap-beijing	北京三区 ap-beijing-3
	北京四区 ap-beijing-4
	北京五区 ap-beijing-5
	北京六区 ap-beijing-6
	北京七区 ap-beijing-7

Kubernetes 版本

- 按量计费超级节点支持 1.16 及以上版本集群。
- 包年包月超级节点当前仅支持 1.20 最高版本集群，请确保集群已升级至最高小版本 1.20-tke.20。

超级节点可调度 Pod 说明

添加了超级节点的集群，依据不同的计费模式，支持调度不同的规格的 Pod。

包年包月的超级节点

- 支持调度 1C～8C 标准规格的 Pod（若为非标准规格，则自动向上转换成标准规格），规格请参考 超级节点可调度 Pod 说明。
- 支持调度 CPU 内存比小于 1:4 的 Pod。

按量计费的超级节点

- 支持调度 0.25C~16C 标准规格的 Pod。
- 支持调度 CPU 内存比小于等于 1:8 的 Pod。
- 支持调度 GPU Pod。

详情请参考 超级节点可调度 Pod 说明。

超级节点调度说明

包年包月的超级节点与包年包月的 TKE 常规节点平权调度。

添加了按量计费的超级节点的 TKE 集群，在包年包月的节点资源不足时，会自动调度到按量计费的超级节点上，若节点资源充足，会优先缩容按量计费超级节点上的 Pod。另外，也支持手动将 Pod 调度至超级节点。

详情请参考 调度至超级节点。

应用场景

超级节点适用于全业务场景，包年包月模式适用于 Pod 规格在 1C 到 8C 的所有在线常驻业务，按量计费模式适用于弹性场景。

在线长驻业务使用超级节点

优势：低成本、易管理

相较于包年包月的普通节点，超级节点单核价格便宜约 20%。
对于在线长驻的服务，用户所需算力固定，若当前 Pod 规格大部分在 8C 以下，可使用超级节点的包年包月模式，基于 Pod 总规格按需购买包年包月的超级节点，将 1C~8C Pod 全部调度至超级节点，降低集群资源的单核价格，实现降本诉求。

弹性业务使用按量计费的超级节点

优势：低成本、高弹性

弹性业务场景下，用户使用按量计费的超级节点，可实现快速秒级扩容，轻松应对突发流量，通过减少预留的资源 buffer 降低成本。
高弹性：快速秒级扩容，轻松应对突发流量，业务流量下降后自动销毁 Pod，无损缩容。
低成本：减少集群预留 buffer，将集群的节点维护在资源利用率更高、使用和预留更合理的水平，节省成本。

超级节点价格说明

最近更新时间：2022-06-21 17:50:05

计费模式

超级节点提供两种类型的计费模式：按量计费和包年包月，分别适用于不同场景下的用户需求。

计费模式	包年包月	按量计费
付款方式	预付费	购买时 冻结费用，每小时结算
计费单位	元/月	元/秒
单价	单价较低	单价较高
最少使用时长	至少使用一个月	按秒计费，按小时结算，随时购买随时释放
最小购买规格	50C100G	无限制
使用场景	适用于算力需求量长期稳定的成熟在线业务	适用于算力需求量瞬间大幅波动的场景

计费项

按量计费模式超级节点资源类型分为 CPU 和 GPU 两类，包年包月模式暂不支持 GPU 类型。

资源类型	CPU 容器	GPU 容器
计费项	CPU、内存	GPU、CPU、内存

产品定价

包年包月：CPU 容器资源单位时间价格

计量	价格（核/元/月）
CPU	36
内存	18

按量计费：CPU 容器资源单位时间价格

Intel 按量计费

计量	价格（秒）	价格（小时）
CPU	每核 0.00003334 元	每核 0.12 元
内存	每 GiB 0.00001389 元	每 GiB 0.05 元

星星海 AMD 按量计费

基于腾讯云自研星星海服务器，提供可靠、安全、稳定的高性能。详情请参见 云服务器标准型 SA2 介绍。

计量	价格（秒）	价格（小时）
CPU	每核 0.00001528 元	每核 0.055 元
内存	每 GiB 0.00000889 元	每 GiB 0.032 元

按量计费：GPU 容器资源单位时间价格

Tesla V100-NVLINK-32G

计量	价格（秒）	价格（小时）	备注
GPU	每卡 0.003193 元	每卡 11.5 元	同样适用于 vGPU，vGPU 以虚拟化系数（0.25、0.5）乘以此价格
CPU	每核 0.00005778 元	每核 0.208 元	-
内存	每 GiB 0.00003389 元	每 GiB 0.122 元	-

NVIDIA T4

计量	价格（秒）	价格（小时）	备注
GPU	每卡 0.001447 元	每卡 5.21 元	同样适用于 vGPU，vGPU 以虚拟化系数（0.25、0.5）乘以此价格
CPU	每核 0.00002411 元	每核 0.0868 元	-
内存	每 GiB 0.00002411 元	每 GiB 0.0868 元	-

计费示例

包年包月

示例 1

包年包月购买总规格为 200C400G 的超级节点 1 个月，总费用为 36 * 200 + 18 * 400 = 14400，该节点可累计调度算力总和为 200C400G 的符合包年包月超级节点规则的 Pod 资源。

按量计费

示例 1

某 Deployment 的 Pod 资源规格为 2 核 4GB，指定 intel，调度到广州六区，Pod 数固定为 2。假设该 Deployment 从启动到终止，共耗时 5 分钟，即要计算 300 秒的费用。

则该 Deployment 的运行费用为 2 ×（2 × 0.00003334 + 4 × 0.00001389）× 300 = 0.073344 元，匹配优惠折扣后，最终计费为 0.073344 × 0.675 = 0.00495072 元

示例 2

某 CronJob 需要每次启动 10 个 Pod，每个 Pod 资源规格为 4 核 8GB，指定 intel，调度到上海五区，运行 10 分钟后结束。假设该 CronJob 每天执行 2 次，使用弹性容器服务托管该任务。

则该任务每天收费为 2 × 10 ×（4 × 0.00003334 + 8 × 0.00001389）× 600 = 2.93376 元，匹配优惠折扣后，最终计费为 2.93376 × 0.75 = 2.20032 元

超级节点可调度 Pod 说明

最近更新时间：2022-06-15 18:07:25

计费方式

调度到超级节点上的 Pod 支持预付费、后付费（按量计费、竞价）的两种计费模式。

支持超级节点的 Kubernetes 版本

- 按量计费超级节点支持 1.16 及以上版本集群。
- 包年包月超级节点当前仅支持1.20最高版本集群，请确保集群已升级至最高小版本1.20-tke.20。

超级节点上可调度的 Pod 规格

超级节点支持的 Pod 的规格配置是容器运行时可用资源和使用服务计费的依据，请务必了解超级节点 Pod 的资源规格配置。不同计费模式的超级节点支持的可调度 Pod 规格不同。

包年包月模式

- 支持调度 1C～8C 标准规格的 Pod。
- 支持调度 CPU 内存比小于 1:4 的 Pod。

节点支持规格列表：

> 注意：
> 若为非标准规格，则自动向上转换成标准规格。

CPU/核	内存区间/GiB	内存区间粒度/GiB
1	1 – 4	1
2	2 – 8	1
4	8 – 16	1
8	16 – 32	1

按量计费模式

- 支持调度 0.25C～16C 标准规格的 Pod（若为非标准规格，则自动向上转换成标准规格）。
- 支持调度 CPU 内存比小于等于 1:8 的 Pod。

节点支持规格列表：

> 注意：
> 若为非标准规格，则自动向上转换成标准规格。

CPU/核	内存区间/GiB	内存区间粒度/GiB
0.25	0.5、1、2	-
0.5	1、2、3、4	-
1	1 – 8	1
2	4 – 16	1

CPU/核	内存区间/GiB	内存区间粒度/GiB
4	8 – 32	1
8	16 – 32	1
12	24 – 48	1
16	32 – 64	1

超级节点配置说明

Pod 临时存储

每个调度到超级节点上的 Pod，创建时会分配 20GiB 的临时镜像存储。

> 注意：
>
> - 临时镜像存储将于 Pod 生命周期结束时删除，请勿用于存储重要数据。
> - 由于需存储镜像，实际可用空间小于 20GiB。
> - 重要数据、超大文件等推荐挂载 Volume 持久化存储。

Pod 网络

调度到超级节点上的 Pod 采用的是与云服务器、云数据库等云产品平级的 VPC 网络，每个 Pod 都会占用一个 VPC 子网 IP。
Pod 与 Pod、Pod 与其他同 VPC 云产品间可直接通过 VPC 网络通信，没有性能损耗。

Pod 隔离性

调度到超级节点上的 Pod 拥有与云服务器完全一致的安全隔离性。Pod 在腾讯云底层物理服务器上调度创建，创建时会通过虚拟化技术保证 Pod 间的资源隔离。

其他 Pod 特殊配置

调度到超级节点上的 Pod 可以通过在 yaml 中定义 template annotation 的方式，实现为 Pod 绑定安全组、分配资源、分配 EIP 等能力。配置方法见下表：

> 注意：
>
> - 如果不指定安全组，则 Pod 会默认绑定节点池指定的安全组。请确保安全组的网络策略不影响该 Pod 正常工作，例如，Pod 启用 80 端口提供服务，请放通入方向 80 端口的访问。
> - 如需分配 CPU 资源，则必须同时填写 cpu 和 mem 2 个 annotation，且数值必须符合 资源规格 中的 CPU 规格。
> - 如需通过 annotation 指定的方式分配 GPU 资源，则必须同时填写gpu-type 及 gpu-count 2 个 annotation，且数值必须符合 资源规格 中的 GPU 规格。

Annotation Key	Annotation Value 及描述	是否必填
eks.tke.cloud.tencent.com/security-group-id	工作负载默认绑定的安全组，请填写 安全组 ID：可填写多个，以,分割。例如 sg-id1,sg-id2。网络策略按安全组顺序生效。	否。如不填写，则默认绑定节点池指定的安全组。如填写，请确保同地域已存在该安全组 ID。
eks.tke.cloud.tencent.com/cpu	Pod 所需的 CPU 核数，请参考 资源规格 填写。默认单位为核，无需再次注明。	否。如填写，请确保为支持的规格，且需完整填写 cpu 和 mem 两个参数。
eks.tke.cloud.tencent.com/mem	Pod 所需的内存数量，请参考 资源规格 填写，需注明单位。例如，512Mi、0.5Gi、1Gi。	否。如填写，请确保为支持的规格，且需完整填写 cpu 和 mem 两个参数。

Annotation Key	Annotation Value 及描述	是否必填
eks.tke.cloud.tencent.com/cpu-type	Pod 所需的 CPU 资源型号，目前支持型号如下：intelamd 具体型号，如 S4、S3 各型号支持的具体配置请参考 资源规格。	否。如果不填写则默认不强制指定 CPU 类型，会根据 指定资源规格方法 尽量匹配最合适的规格，若匹配到的规格 Intel 和 amd 均支持，则优先选择 Intel。
eks.tke.cloud.tencent.com/gpu-type	Pod 所需的 GPU 资源型号，目前支持型号如下：V1001/4 *T41/2*T4T4 支持优先级顺序写法，如 “T4,V100” 表示优先创建 T4 资源 Pod，如果所选地域可用区 T4 资源不足，则会创建 V100 资源 Pod。各型号支持的具体配置请参考 资源规格。	如需 GPU，则此项为必填项。填写时，请确保为支持的 GPU 型号，否则会报错。
eks.tke.cloud.tencent.com/gpu-count	Pod 所需的 GPU 数量，请参考 资源规格 填写，默认单位为卡，无需再次注明。	否。如填写，请确保为支持的规格。
eks.tke.cloud.tencent.com/retain-ip	Pod 固定 IP，value 填写 "true" 开启此特性，开启特性的 Pod，当 Pod 被销毁后，默认会保留这个 Pod 的 IP 24 小时。24 小时内 Pod 重建，还能使用该 IP。24 小时以后，该 IP 有可能被其他 Pod 抢占。**仅对 statefulset、rawpod 生效。**	否
eks.tke.cloud.tencent.com/retain-ip-hours	修改 Pod 固定 IP 的默认时长，value 填写数值，单位是小时。默认是 24 小时，最大可支持保留一年。**仅对 statefulset、rawpod 生效。**	否
eks.tke.cloud.tencent.com/eip-attributes	表明该 Workload 的 Pod 需要关联 EIP，值为 "" 时表明采用 EIP 默认配置创建。"" 内可填写 EIP 云 API 参数 json，实现自定义配置。例如 annotation 的值为 '{"InternetMaxBandwidthOut":2}' 即为使用 2M 的带宽。注意，非带宽上移的账号无法使用。	否
eks.tke.cloud.tencent.com/eip-claim-delete-policy	Pod 删除后，EIP 是否自动回收，“Never” 不回收，默认回收。该参数只有在指定 eks.tke.cloud.tencent.com/eip-attributes 时才生效。注意，非带宽上移的账号无法使用。	否
eks.tke.cloud.tencent.com/eip-id-list	如果工作负载为 StatefulSet，也可以使用指定已有 EIP 的方式，可指定多个，如"eip-xx1,eip-xx2"。请注意，StatefulSet pod 的数量必须小于等于此 annotation 中指定 EIP Id 的数量，否则分配不到 EIP 的 pod 会处于 Pending 状态。注意，非带宽上移的账号无法使用。	否

示例请参考 Annotation 说明。

默认配额

购买包年包月的超级节点时，将按照总规格分配默认配额，开通按量计费的超级节点，默认每个集群仅可将** 100 个 Pod** 调度到超级节点上。若您需要超过以上配额的资源，可填写提升配额申请，由腾讯云对您的实际需求进行评估，评估通过之后将为您提升配额。

申请提升配额操作指引

1. 请 提交工单 ，选择**人工支持**或者**其他问题** > **立即创建**，进入创建工单信息填写页面。
2. 在问题描述中填写“期望提升集群超级节点 Pod 配额”，注明目标地区及目标配额，并按照页面提示填写您可用的手机号等信息。
3. 填写完成后，单击**在线咨询**即可。

Pod 限制说明

Workload 限制

DaemonSet 类型工作负载的 Pod 不会调度到超级节点上。

Service 限制

采用 GlobalRouter 网络模式 的集群 service 如果开启了 externaltrafficpolicy = local，流量不会转发到调度到超级节点上的 Pod。

Volume 限制

挂载 hostPath 类型数据卷的 Pod 不会调度到超级节点上。

其他限制

- 没有任何服务器节点的空集群暂时无法正常使用超级节点功能。
- 开启了 固定 IP 的 Pod 暂不支持调度到超级节点上。
- 指定了 hostPort 的 Pod 不会调度到超级节点上。
- 指定了 hostIP 配置的 Pod 默认会把 Pod IP 作为 hostIP。
- 如果开启了反亲和性特性，同工作负载 Pod 仅会在超级节点上创建一个。
- 如果容器日志存储在指定的节点文件中，也是通过节点文件进行的日志采集，则无法采集超级节点上的 Pod 日志。

调度 Pod 至超级节点

最近更新时间：2022-05-19 20:57:21

本篇文章主要介绍在容器服务 TKE 集群中，如何调度 Pod 至超级节点，主要有两种调度方式：

- 自动调度
- 手动调度

自动扩容

- 若集群配置了包年包月超级节点，在 Pod 符合超级节点调度规则的前提下，包年包月的超级节点与包年包月的常规节点平权调度。
- 若集群配置了按量计费超级节点，则当业务高峰且已有包年包月节点资源不足时，会自动调度 Pod 至超级节点，无需购买服务器；业务恢复平稳，自动释放在超级节点中的 Pod 资源，也无需再进行退还机器操作。
- 如果集群同时开启了 Cluster Autoscaler 和按量计费超级节点，则会尽量优先将 Pod 调度到按量计费的超级节点上，而非触发集群节点扩容。如果受调度限制影响，Pod 无法调度到超级节点上，则会依然正常触发集群节点扩容。而服务器节点资源充足时，会优先缩容按量计费超级节点上的 Pod。

手动调度

支持用户手动将 Pod 调度至超级节点，默认按量计费的超级节点会自动添加 Taints 以降低调度优先级，如需手动调度 Pod 到超级节点或指定超级节点调度，通常需要为 Pod 添加对应的 Tolerations。但并非所有的 Pod 均可以调度到超级节点上，详情请参见 超级节点调度说明。为方便使用，您可以在 Pod Spec 中指定 nodeselector 。示例如下：

```
spec:
nodeSelector:
node.kubernetes.io/instance-type: eklet
```

容器服务 TKE 的管控组件会判断该 Pod 是否可以调度到超级节点，若不支持则不会调度到超级节点。

包年包月的超级节点当前仅支持调度指定规格和指定 CPU 内存比的 Pod，若不符合规则，则调度不会成功。

超级节点 Annotation 说明

最近更新时间：2022-06-23 17:19:48

通过在 YAML 文件中定义 Annotation（注解）的方式，可以实现超级节点丰富的自定义能力。您可以从 Annotation 说明 中了解通过注解可以对超级节点进行的常见配置操作。

采集超级节点上的 Pod 日志

最近更新时间：2022-05-19 20:59:09

本文主要介绍 TKE 集群中调度至超级节点的 Pod 如何采集日志，包括：

- 采集日志至 CLS
- 采集日志至 Kafka

采集日志至 CLS

服务角色授权

在采集超级节点上的 Pod 日志至 CLS 之前，需要进行服务角色授权，以保证将日志正常上传到 CLS。

操作步骤如下：

1. 登录**访问管理控制台** > **角色**。
2. 在角色页面单击**新建角色**。
3. 在“选择角色载体”中，选择**腾讯云产品服务** > **容器服务(tke)** > **容器服务-EKS日志采集**，并单击**下一步**。如下图所示：

4. 确认角色策略，单击**下一步**。
5. 审阅角色策略，单击**完成**，即可完成为该账号配置该角色。

配置日志采集

服务角色授权完成后，需要开启 TKE 日志采集功能，并配置相应的日志采集规则。例如，指定工作负载采集和指定 pod labels 采集。详情可参见 通过控制台使用 CRD 配置日志采集。

采集日志至 Kafka

若需要采集超级节点上的 Pod 的日志至自建 Kafka 或者 CKafka，需要您自行配置 CRD，定义采集源及消费端，CRD 配置完成后，Pod 自带的采集器会依照规则进行日志采集。

CRD 具体配置如下所示：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig ## 默认值
metadata:
name: test ## CRD资源名，在集群内唯一
spec:
kafkaDetail:
brokers: xxxxxx # 必填，broker地址，一般是域名:端口，多个地址以“,”分隔
topic: xxxxxx # 必填，topicID
messageKey: # 选填，指定pod字段作为key上传到指定分区
valueFrom:
fieldRef:
fieldPath: metadata.name
```

```
timestampKey: #时间戳的key，默认是@timestamp
timestampFormat: #时间戳的格式，默认是double
inputDetail:
type: container_stdout ## 采集日志的类型，包括container_stdout（容器标准输出）、container_file（容器文件）

containerStdout: ## 容器标准输出
namespace: default ## 采集容器的kubernetes命名空间，如果不指定，代表所有命名空间
allContainers: false ## 是否采集指定命名空间中的所有容器的标准输出
container: xxx ## 采集日志的容器名，此处可填空
includeLabels: ## 采集包含指定label的Pod
k8s-app: xxx ## 只采pod标签中配置"k8s-app=xxx"的pod产生的日志，与workloads、allContainers=true不能同时指定
workloads: ## 要采集的容器的Pod所属的kubernetes workload
- namespace: prod ## workload的命名空间
name: sample-app ## workload的名字
kind: deployment ## workload类型，支持deployment、daemonset、statefulset、job、cronjob
container: xxx ## 要采集的容器名，如果填空，代表workload Pod中的所有容器

containerFile: ## 容器内文件
namespace: default ## 采集容器的kubernetes命名空间，必须指定一个命名空间
container: xxx ## 采集日志的容器名，此处可填*
includeLabels: ## 采集包含指定label的Pod
k8s-app: xxx ## 只采pod标签中配置"k8s-app=xxx"的pod产生的日志，与workload不能同时指定
workload: ## 要采集的容器的Pod所属的kubernetes workload
name: sample-app ## workload的名字
kind: deployment ## workload类型，支持deployment、daemonset、statefulset、job、cronjob
logPath: /opt/logs ## 日志文件夹，不支持通配符
filePattern: app_*.log ## 日志文件名，支持通配符 * 和 ？，* 表示匹配多个任意字符，？表示匹配单个任意字符
```

超级节点常见问题

最近更新时间：2022-05-19 20:59:18

第三方节点管理
第三方节点概述

最近更新时间：2022-01-24 15:55:24

简介

第三方节点是针对混合云场景提供的混合集群功能，允许用户将非腾讯云的主机，添加到容器服务 TKE 集群，由用户提供计算资源，容器服务 TKE 负责集群生命周期管理。

使用场景

上云资源利旧

用户期望将更多的业务迁移到 TKE 公有云集群，但在 IDC 有存量主机资源，期望将 IDC 主机资源添加到 TKE 公有云集群，确保在上云过程中存量主机资源得到有效利用，随着 IDC 主机资源的逐渐淘汰，逐步将业务全量迁移到云上。

集群免运维

用户业务在 IDC，但期望由 TKE 公有云来管理集群创建、升级、监控等集群生命周期管理问题，同时获得与公有云一致的 API 与用户体验。

注意事项

- 第三方节点特性目前处于内测阶段，如果您想要使用第三方节点功能，请通过 在线咨询 联系我们。
- IDC 节点的操作系统必须使用 TencentOS Server 3.1 。
- 第三方节点特性仅支持在版本为**1.18及以上** TKE 集群中使用。
- 第三方节点特性仅支持在网络插件为 **VPC-CNI-单网卡多IP模式** 或者 **Cilium-Overlay** 的 TKE 集群开启。
- 为了保障第三方节点的稳定性，第三方节点仅支持内网连接。

相关概念

节点池

为帮助您高效管理 Kubernetes 集群内节点，腾讯云容器服务引入节点池概念，节点池利用节点模板，来管理一组同质节点。借助节点池基本功能，您可以方便快捷地创建、管理和销毁节点，以及实现节点的动态扩缩容。详情请参见 节点池概述。

相关操作

您可以登录 容器服务控制台 并参考以下文档， 进行对应第三方节点操作：

- 开启第三方节点功能
- 创建第三方节点池
- 新建第三方节点
- 编辑第三方节点池
- 删除第三方节点池

管理第三方节点池

最近更新时间：2022-01-24 15:55:08

操作场景

本文介绍如何管理集群中的第三方节点池，包括如何调整第三方节点池和调整节点池中的第三方节点。

前提条件

- 已 创建集群。
- 集群的网络模型为 VPC-CNI-单网卡多IP模式 或者 Cilium-Overlay。
- 集群 Kubernetes 版本为 1.18 及以上版本。

操作步骤

开启第三方节点功能

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “基本信息” 页面。
3. 在**节点和网络信息**中，单击“支持导入第三方节点”右侧的 。
4. 在“开启第三方节点功能”中根据实际需求配置相关参数。如下图所示：

   - **子网**：节点访问 kube-apiserver 需要通过您的 VPC 弹性网卡，因此需要您提供 VPC 子网，容器服务 TKE 会自动在选定的子网内创建代理弹性网卡。
   - **容器网络插件**：针对于云下第三方节点的网络插件，目前支持 Cilium VXLan 和 Cilium BGP 两种类型。云上和云下必须是同一类型的网络插件，全是 Underlay 或者全是 Overlay。如云上网络插件是 Cilium-Overlay，则这里默认是 Cilium-Overlay；如云上是 VPC-CNI-单网卡多IP模式（Underlay），则这里默认是 Cilium BGP。
   - **容器网络**：系统将为第三方节点上运行的 Pod 分配该容器网络地址范围内的 IP 地址，仅当云下的网络插件是 Cilium-BGP 需要填写。
5. 单击**确认开启**。

创建第三方节点池

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “Deployment” 页面。
3. 选择左侧菜单栏中的**节点管理** > **节点池**，进入“节点池列表”页面。

4. 单击**新建节点池**，进入“新建节点池”页面，参考以下提示进行设置。如下图所示：

- **节点池名称**：自定义，可根据业务需求等信息进行命名，方便后续资源管理。
- **节点池类型**：目前支持**云服务器**和**虚拟节点**和**第三方节点**三种类型。当开启第三方节点功能后，选择**第三方节点**类型。
- **容器目录**：勾选即可设置容器和镜像存储目录，建议存储到数据盘。例如 /var/lib/docker。
- **运行时组件**：容器运行时组件，当前支持**docker**和**containerd**。
- **运行时版本**：容器运行时组件的版本。
- **封锁初始节点**：勾选**开启封锁**后，将不接受新的 Pod 调度到该节点，需要手动取消封锁的节点，或在自定义数据中执行 取消封锁命令，请按需设置。
- **Label**：单击**新增Label**，即可进行 Label 自定义设置。该节点池下所创建的节点均将自动增加此处设置的 Label，可用于后续根据 Label 筛选、管理第三方节点。

> 说明
> 必填 Label 说明（**仅填写 value，请勿更改 key**）：
> - tke.cloud.tencent.com/location
> - 如果容器网络模式设置为 **Cilium BGP** 模式，还需填写如下两组 Label：
>   - infra.tce.io/as 为节点所属交换机的 BGP AS 号，请根据业务环境设置。
>   - infra.tce.io/switch-ip 为节点所属交换机的交换机 IP，请根据业务环境配置。

- **Taints**：节点属性，通常与 Tolerations 配合使用。此处可为节点池下的所有节点设置 Taints，确保不符合条件的 Pod 不能够调度到这些节点上，且这些节点上已存在的不符合条件的 Pod 也将会被驱逐。
  Taints 内容一般由 key、value 及 effect 三个元素组成。其中 effect 可取值通常包含以下三种：
  - **PreferNoSchedule**：非强制性条件，尽量避免将 Pod 调度到设置了其不能容忍的 taint 的节点上。

- **NoSchedule**：当节点上存在 taint 时，没有对应容忍的 Pod 一定不能被调度。
- **NoExecute**：当节点上存在 taint 时，对于没有对应容忍的 Pod，不仅不会被调度到该节点上，该节点上已存在的 Pod 也会被驱逐。以设置 Taints key1=value1:PreferNoSchedule 为例，控制台配置如下图所示：

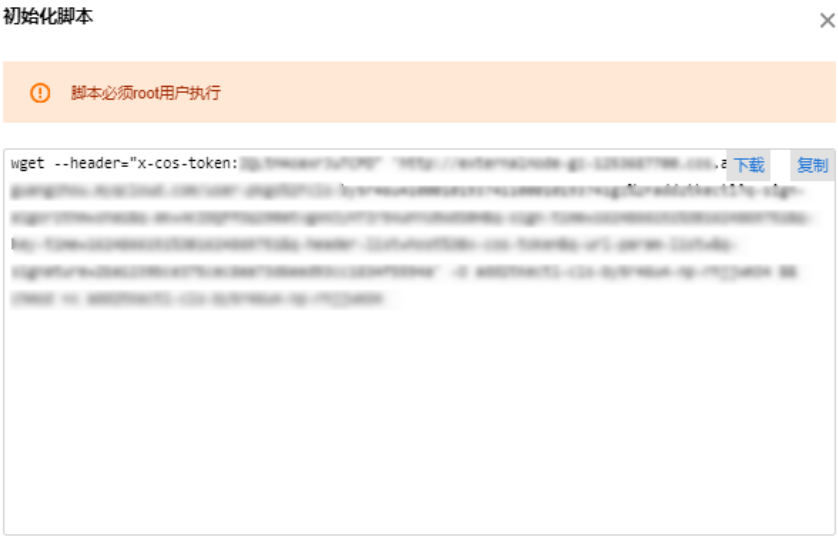
- **自定义数据**：指定自定义数据来配置节点，即当节点启动后运行配置的脚本。需确保脚本的可重入及重试逻辑，脚本及其生成的日志文件可在节点的 /usr/local/qcloud/tke/userscript 路径查看。

5. 单击**创建节点池**即可创建第三方节点池。

新建第三方节点

成功创建第三方节点池后，此时节点池内还没有节点，请参考以下步骤导入节点：

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “Deployment” 页面。
3. 选择左侧菜单栏中的**节点管理** > **节点池**，进入“节点池列表”页面。
4. 在“节点池名片页”中，单击目标节点池 ID。
5. 进入该节点池详情页，单击**新建节点**，获取导入节点的脚本。
6. 在“初始化脚本”弹窗中，复制或下载脚本，如下图所示：



7. 在您的机器上执行脚本。

> ⚠ 注意
> 脚本下载链接1小时后过期。因为脚本通过 COS 下载，所以需要确保 IDC 节点能够通过内网/外网访问 COS。

8. 执行如下命令，完成节点添加：

```
./add2tkectl-cls-m57oxxxp-np-xxxx install
```

编辑第三方节点池

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “Deployment” 页面。
3. 选择左侧菜单栏中的**节点管理** > **节点池**，进入“节点池列表”页面。
4. 在“节点池名片页”中，选择计划修改的类型为**第三方节点**的节点池，单击**编辑**。
5. 在“调整第三方节点池配置”弹窗中，支持修改节点池名称、Labels、Taints 等信息。如下图所示：

> ⚠ 注意：

Labels、Taints 相关的修改会在节点池中的所有第三方节点上生效，如果有特殊调度策略请务必谨慎操作。

删除第三方节点池

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在“集群管理”列表页面，选择目标集群 ID，进入该集群 “Deployment” 页面。
3. 选择左侧菜单栏中的**节点管理** > **节点池**，进入“节点池列表”页面。
4. 选择计划删除节点池名片页右上角的**删除**。
5. 在“删除第三方节点池”弹窗中，单击**确认**删除节点池。

⚠ 注意

- 删除第三方节点池后，池内所有第三方节点仅会被移出集群，不会清除节点上运行的 Pod。
- 删除节点后为了保证安全，建议重装节点或者执行以下命令删除节点上 kube-apiserver 访问配置：

```
>rm - rf /etc/kubernetes $HOME/.kube
```

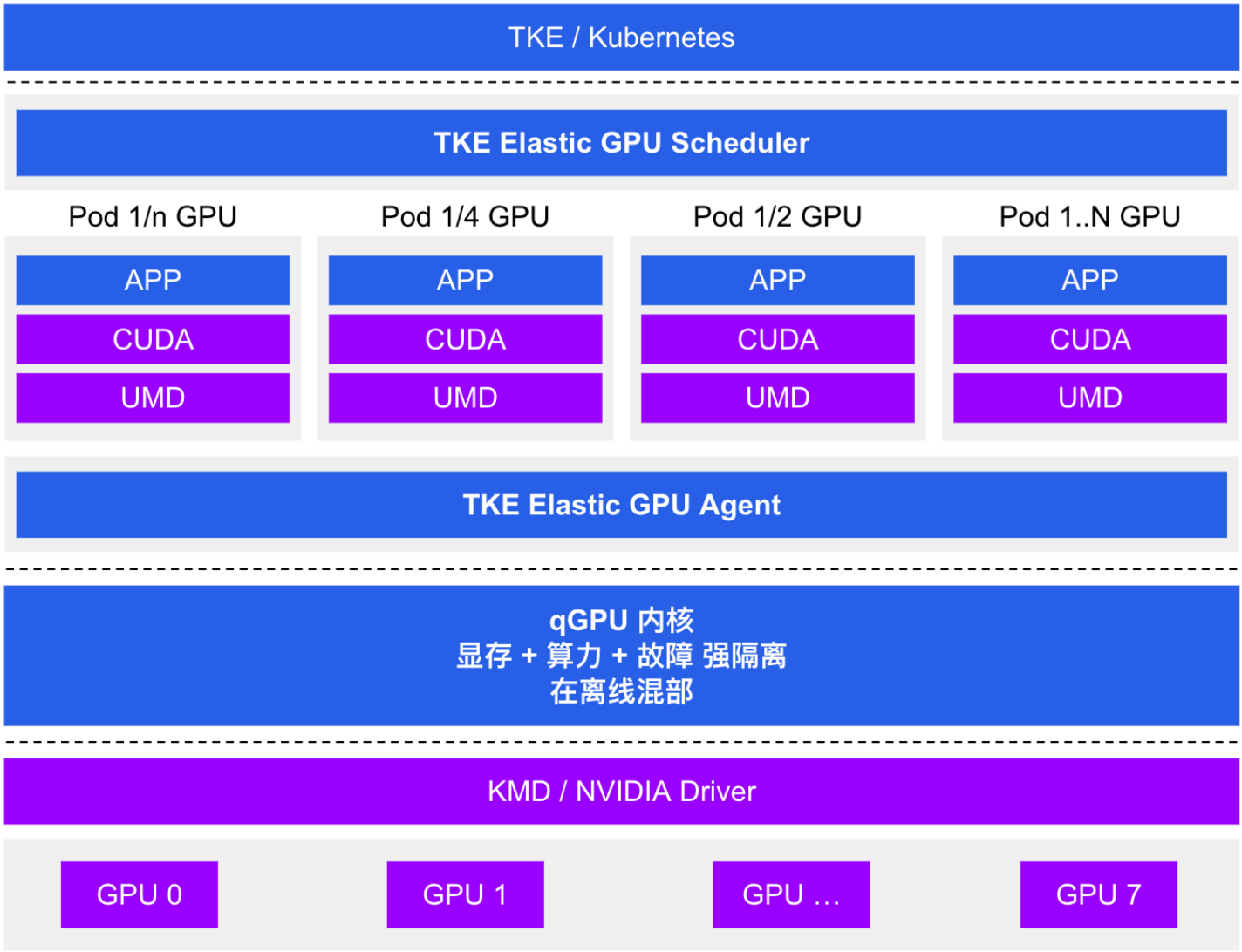
GPU 共享
qGPU 概述

最近更新时间：2022-06-07 11:02:03

qGPU 概述

qGPU 是腾讯云推出的 GPU 共享技术，支持在多个容器间共享 GPU 卡并提供容器间显存、算力强隔离的能力，从而在更小粒度的使用 GPU 卡的基础上，保证业务安全，达到提高 GPU 使用率、降低客户成本的目的。
qGPU 依托 TKE 对外开源的 Elastic GPU 框架，可实现对 GPU 算力与显存的细粒度调度，并支持多容器共享 GPU 与多容器跨 GPU 资源分配。同时依赖底层强大的 qGPU 隔离技术，可做到 GPU 显存和算力的强隔离，在通过共享使用 GPU 的同时，尽量保证业务性能与资源不受干扰。

方案框架图



qGPU 优势

最近更新时间：2022-06-07 11:02:12

qGPU 产品优势

- **灵活性：**精细配置 GPU 算力占比和显存大小。
- **强隔离：**支持显存和算力的严格隔离。
- **在离线：**支持业界唯一在离线混部能力，GPU 利用率压榨到极致。
- **覆盖度：**支持主流架构 Volta（如 V100 等）、Turing（如 T4 等）、Ampere（如 A100、A10 等)。
- **云原生：**支持标准 Kubernetes 和 NVIDIA Docker。
- **兼容性：**业务不重编、CUDA 库不替换、业务无感。
- **高性能：**GPU 设备底层虚拟化，高效收敛，吞吐接近0损耗。

使用 qGPU

最近更新时间：2022-06-07 11:02:15

使用须知

- **版本支持：** TKE 版本 ≥ v1.14.x
- **OS 支持：** 仅支持特定的 Tencent OS 3.1 镜像。
- **GPU 卡架构：** 支持 Volta（如 **V100**）、Turing（如 **T4**）、Ampere（如 **A100**、**A10**）。
- **驱动版本：** 默认预装 NVIDIA 驱动 450.102.04 / 470.82.01；支持 CUDA 11.4 及以下。为保证兼容性，强烈建议用户使用节点预安装 NVIDIA 驱动，无需在 POD 内部重复安装。
- **共享粒度：** 每个 qGPU 最小分配 1G 显存，精度单位是 1G。算力最小分配 5（代表一张卡的 5%），最大 100（代表一张卡），精度单位是 5（即 5、10、15、20 ... 100）。
- **整卡分配：** 开启了 qGPU 能力的节点可按照 tke.cloud.tencent.com/qgpu-core: 100 | 200 | ...（N * 100，N 是整卡个数）的方式分配整卡。建议通过 TKE 的节点池能力来区分 nvidia 分配方式或转换到 qGPU 使用方式。
- **个数限制：** 一个 GPU 上最多可创建 16 个 qGPU 设备。建议按照容器申请的显存大小确定单个 GPU 卡可共享部署的 qGPU 个数。
- **升级需知：** 如需升级 Kubernetes Master 版本，请注意：
  - 对于托管集群，无需重新设置本插件。
  - 对于独立集群，master 版本升级会重置 master 上所有组件的配置，从而影响到 qgpu-scheduler 插件作为 Scheduler Extender 的配置，因此 qGPU 插件需要卸载后再重新安装。

操作步骤

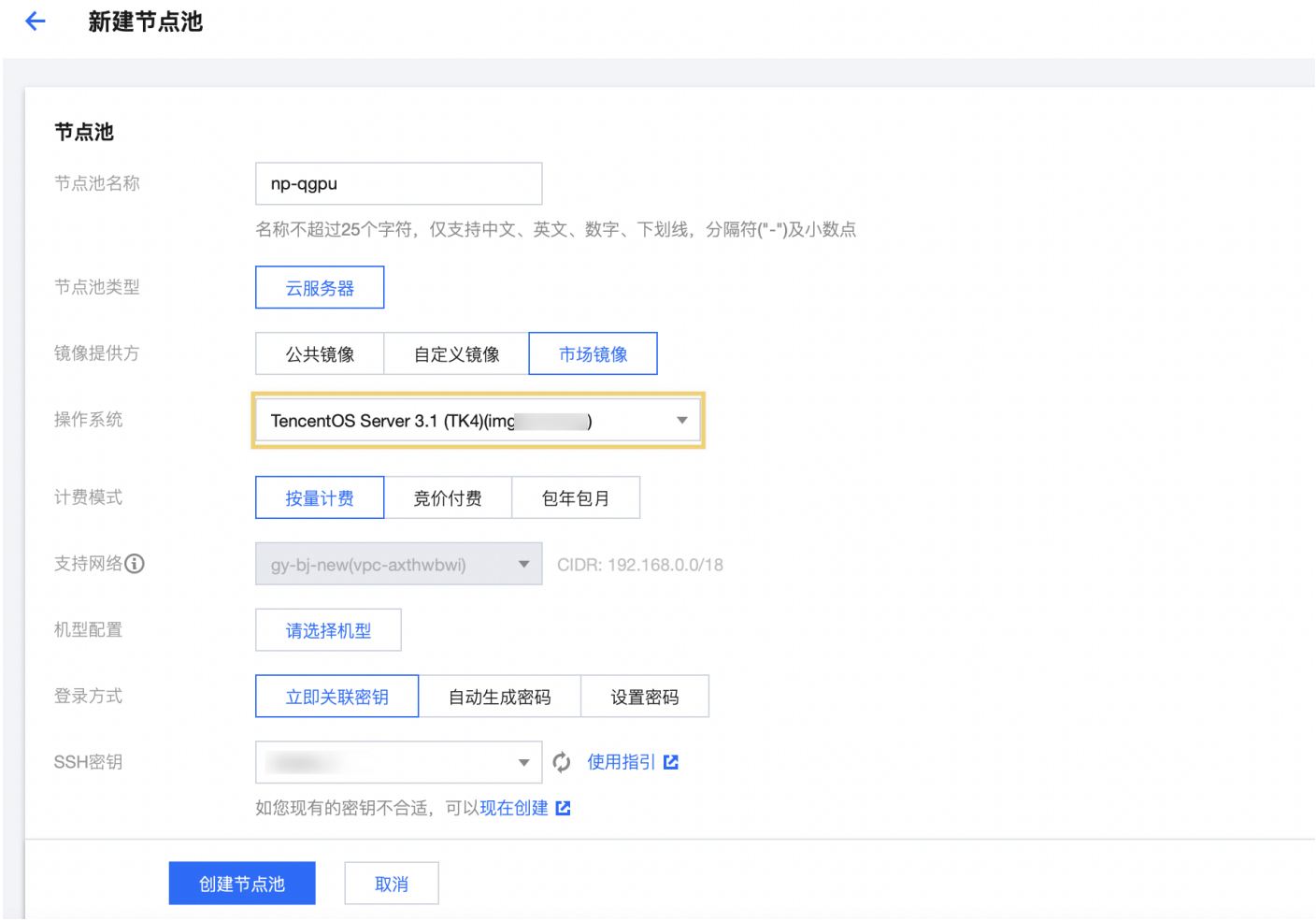
> 说明：
> 由于使用 qGPU 能力需要使用特定镜像以及设置相关 Label，因此强烈建议您使用 TKE 的节点池能力来对节点进行分组管理（节点池的节点具备统一的 Label 以及镜像属性），详情请参见 新建节点池 。

安装 qGPU

1. 登录 容器服务控制台 ，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入“组件列表”页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 QGPU（GPU隔离组件）。
5. 单击**参数配置**，可以设置 qgpu-scheduler 的调度策略。
   - **spread**：多个 Pod 会分散在不同节点、不同显卡上，优先选择资源剩余量较多的节点，适用于高可用场景，避免把同一个应用的副本放到同一个设备上。
   - **binpack**：多个 Pod 会优先使用同一个节点，适用于提高 GPU 利用率的场景。
6. 单击**完成**即可创建组件。安装成功后，需要为集群准备 GPU 资源。

准备 GPU 资源

1. 单击**新建节点池**，选中**qGPU 专用市场镜像**。如下图所示：



通过 qGPU 指定的镜像创建节点后，tke 后台会自动给节点添加 label qgpu-device-enable:"enable"，设置了该 label 后，DaemonSet qgpu-manager 会调度到对应节点上，并自动进行 qGPU 相关的设置。

2. 通过节点池的高级配置来设置 Label，从而指定 qGPU 隔离策略（填写 Label value 时，可填写全称或者缩写）：
   - **Label 键：**tke.cloud.tencent.com/qgpu-schedule-policy。

- Label 值：fixed-share（更多取值可参考下方 隔离策略说明 ）。

当前 qGPU 支持以下三种隔离策略：

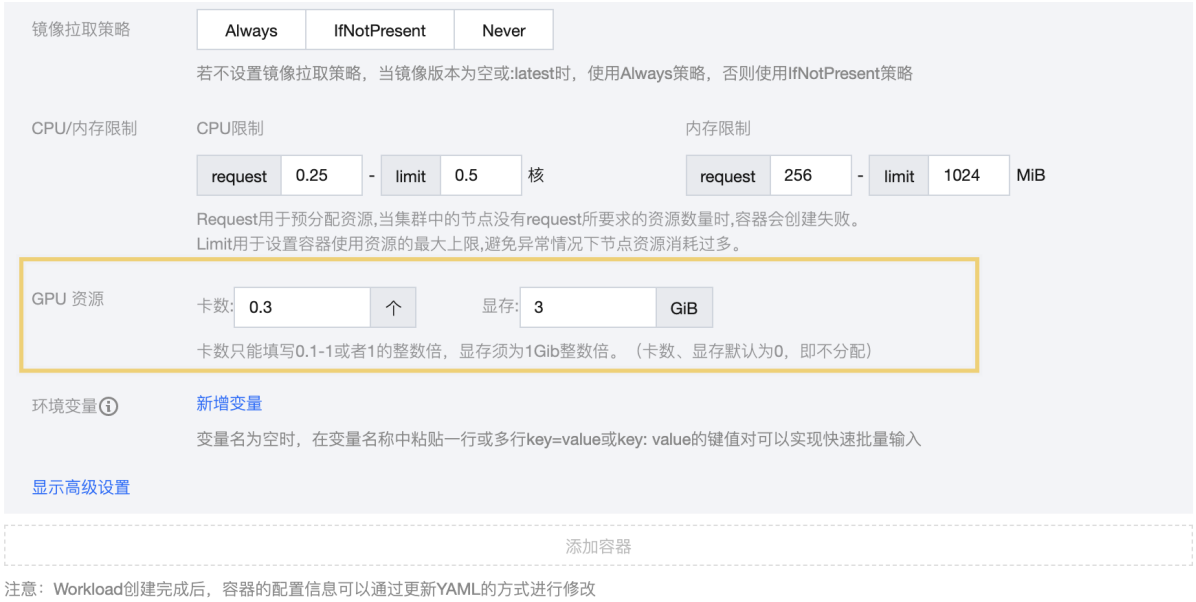
Label 值	缩写	英文名	中文名	含义
best-effort（默认值）	be	Best Effort	争抢模式	默认值。各个 Pods 不限制算力，只要卡上有剩余算力就可使用。如果一共启动 N 个 Pods，每个 Pod 负载都很重，则最终结果就是 1/N 的算力。
fixed-share	fs	Fixed Share	固定配额	每个 Pod 有固定的算力配额，无法超过固定配额，即使 GPU 还有空闲算力。
burst-share	bs	Guaranteed Share with Burst	保证配额加弹性能力	调度器保证每个 Pod 有保底的算力配额，但只要 GPU 还有空闲算力，就可被 Pod 使用。例如，当 GPU 有空闲算力时（没有分配给其他 Pod），Pod 可以使用超过它的配额的算力。注意，当它所占用的这部分空闲算力再次被分配出去时，Pod 会回退到它的算力配额。

3. 为应用分配 GPU 资源。通过给容器设置 qGPU 对应资源可以允许 Pod 使用 qGPU，您可以通过控制台或者 YAML 方式来设置：

> **说明：**
>
> - **如果应用需要使用整数卡资源，只需填写卡数，无需填写显存（自动使用分配的 GPU 卡上全部显存）。**
> - **如果应用需要使用小数卡资源（即和其他应用共享同一张卡），需要同时填写卡数和显存。**

通过控制台设置

在“新建 Workload 页面”，直接填写 GPU 相关资源，如下图所示：



通过 YAML 设置

通过 YAML 来设置相关 qGPU 资源：

```
spec:
containers:
resources:
limits:
tke.cloud.tencent.com/qgpu-memory: "5"
tke.cloud.tencent.com/qgpu-core: "30"
requests:
tke.cloud.tencent.com/qgpu-memory: "5"
tke.cloud.tencent.com/qgpu-core: "30"
```

其中：

- requests 和 limits 中和 qGPU 相关的资源值必须一致（根据 K8S 的规则，可以省略掉 requests 中对 qGPU 的设置，这种情况下 requests 会被自动设置为和 limits 相同的值）。
- tke.cloud.tencent.com/qgpu-memory 表示容器申请的显存（单位G），**整数分配，不支持小数。**
- tke.cloud.tencent.com/qgpu-core 代表容器申请的算力，每个 GPU 卡可以提供100%算力，**qgpu-core 的设置应该小于100**，设置值超过剩余算力比例值，则设置失败，设置后容器可以得到一张 GPU 卡 n% 的算力。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	请求资源	Namespace
qgpu-manager	DaemonSet	每 GPU 节点一个 Memory: 300M, CPU:0.2	kube-system
qgpu-manager	ClusterRole	-	-
qgpu-manager	ServiceAccount	-	kube-system
qgpu-manager	ClusterRoleBinding	-	kube-system
qgpu-scheduler	Deployment	单一副本 Memory: 800M, CPU:1	kube-system
qgpu-scheduler	ClusterRole	-	-

Kubernetes 对象名称	类型	请求资源	Namespace
qgpu-scheduler	ClusterRoleBinding	-	kube-system
qgpu-scheduler	ServiceAccount	-	kube-system
qgpu-scheduler	Service	-	kube-system

Kubernetes 对象管理

概述

最近更新时间：2022-04-18 14:14:44

对象管理说明

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

- 正在运行的应用程序
- 应用程序可用的资源
- 应用程序关联的策略等

您可以直接通过 TKE 控制台或者 Kubernetes API 使用 Kubernetes 的对象，例如 Kubectl。

对象分类

Kubernetes 常用对象主要分为以下类型：

对象分类		对象说明	对象管理操作
工作负载	Deployment	用于管理指定调度规则的 Pod。	Deployment 管理
	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。	StatefulSet 管理
	DaemonSet	确保所有或部分节点上运行 Pod，例如日志采集程序。	DaemonSet 管理
	Job	一个 Job 创建一个或多个 Pod，直至运行结束。	Job 管理
	CronJob	定时运行的 Job 任务。	CronJob 管理
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。	Service 管理
	Ingress	管理集群中 Services 的外部访问的 Kubernetes 对象。	Ingress 管理
配置	ConfigMap	用于保存配置信息。	ConfigMap 管理
	Secret	用于保存敏感信息，例如密码、令牌等。	Secret 管理
存储	Volume	可以存储容器访问相关的数据。	存储管理
	Persistent Volumes（PV）	Kubernetes 集群中配置的一块存储。	
	Persistent Volumes Claim（PVC）	请求存储的声明。如果把 PV 比作 Pod，那么 PVC 相当于工作负载。	
	StorageClass	用于描述存储的类型。 创建 PVC 时，通过 StorageClass 创建指定类型的存储，即存储的模板。	

Kubernetes 对象还包括 Namespaces、HPA、Resource Quotas等数十种，您可以根据业务需要使用不同的 Kubernetes 对象。 不同版本的 Kubernetes 可使用的对象也不相同，更多说明可登录 Kubernetes 官方网站 查询。

资源限制

TKE 使用 ResourceQuota/tke-default-quota 对所有**托管集群**进行以下资源限制，如果您需要更多的配额项数量，请 在线咨询 进行申请。

集群规模	限制总量（单位：个）	
	Pod	ConfigMap
节点数 ≤ 5	4000	3000
5 < 节点数 ≤ 20	8000	6000
节点数 > 20	暂无限制	暂无限制

Namespaces

最近更新时间：2022-01-17 15:11:47

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

使用方法

- 通过 容器服务控制台 使用：容器服务控制台提供 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 资源的使用约束。可以设置 Namespaces 资源的使用约束如下：

- 计算资源的配额，例如 CPU、内存。
- 存储资源的配额，例如请求存储的总存储。
- Kubernetes 对象的计数，例如 Deployment 个数配额。

不同的 Kubernetes 版本，ResourceQuota 支持的配额设置略有差异，更多详情可查看 Kubernetes ResourceQuota 官方文档。
ResourceQuota 的示例如下所示：

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

通过 NetWorkPolicy 设置 Namespaces 网络的访问控制

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

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

工作负载
Deployment 管理

最近更新时间：2022-04-18 14:14:38

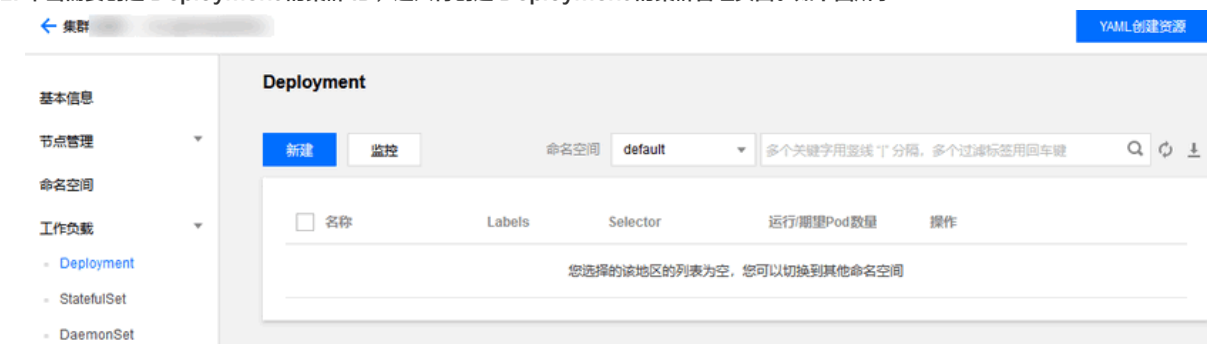
简介

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

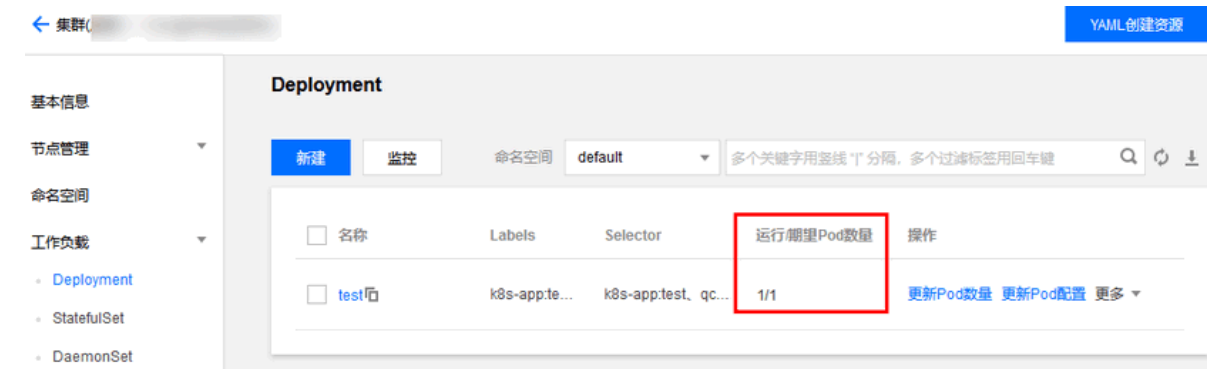
Deployment 控制台操作指引

创建 Deployment

1. 登录容器服务控制台，选择左侧导航栏中的 集群。
2. 单击需要创建 Deployment 的集群 ID，进入待创建 Deployment 的集群管理页面。如下图所示：



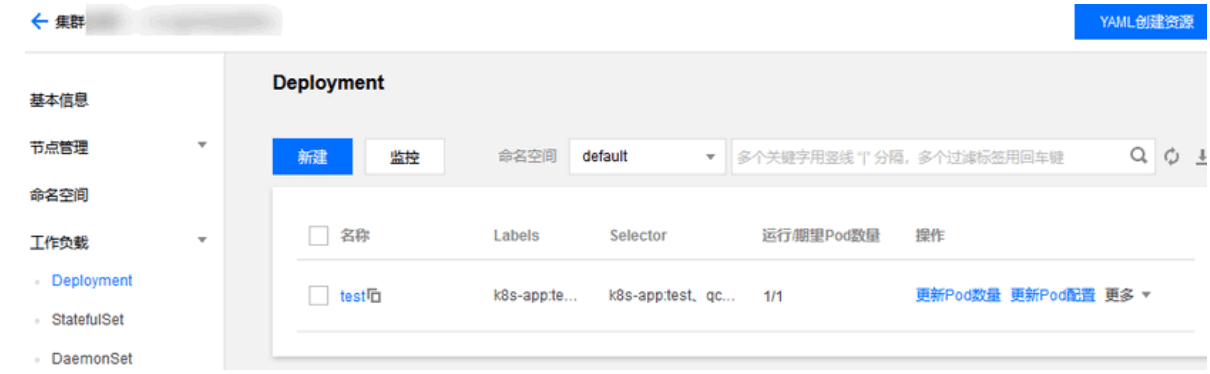
3. 单击**新建**，进入 “新建Workload” 页面。
   根据实际需求，设置 Deployment 参数。关键参数信息如下：
   - **工作负载名**：输入自定义名称。
   - **标签**：一个键 – 值对（Key-Value），用于对资源进行分类管理。
   - **命名空间**：根据实际需求进行选择。
   - **类型**：选择**Deployment（可扩展的部署 Pod）**。
   - **数据卷（选填）**：为容器提供存储，目前支持临时路径、主机路径、云硬盘数据卷、文件存储 NFS、配置文件、PVC，还需挂载到容器的指定路径中。
   - **实例内容器**：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（Tag）**：根据实际需求进行填写。
     - **镜像拉取策略**：提供以下3种策略，请按需选择。
       若不设置镜像拉取策略，当镜像版本为空或 latest 时，使用 Always 策略，否则使用 IfNotPresent 策略。
       - **Always**：总是从远程拉取该镜像。
       - **IfNotPresent**：默认使用本地镜像，若本地无该镜像则远程拉取该镜像。
       - **Never**：只使用本地镜像，若本地没有该镜像将报异常。
     - **CPU/内存限制**：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
     - **GPU 资源**：配置该工作负载使用的最少 GPU 资源。
     - **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**”和“**特权级**”等参数。
   - **镜像访问凭证**：容器镜像默认私有，在创建工作负载时，需选择实例对应的镜像访问凭证。
   - **实例数量**：根据实际需求选择调节方式，设置实例数量。
     - **手动调节**：设定实例数量，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。详情请参见 自动伸缩。
4. 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

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 单击需要更新 Deployment 的集群 ID，进入待更新 Deployment 的集群管理页面。如下图所示：

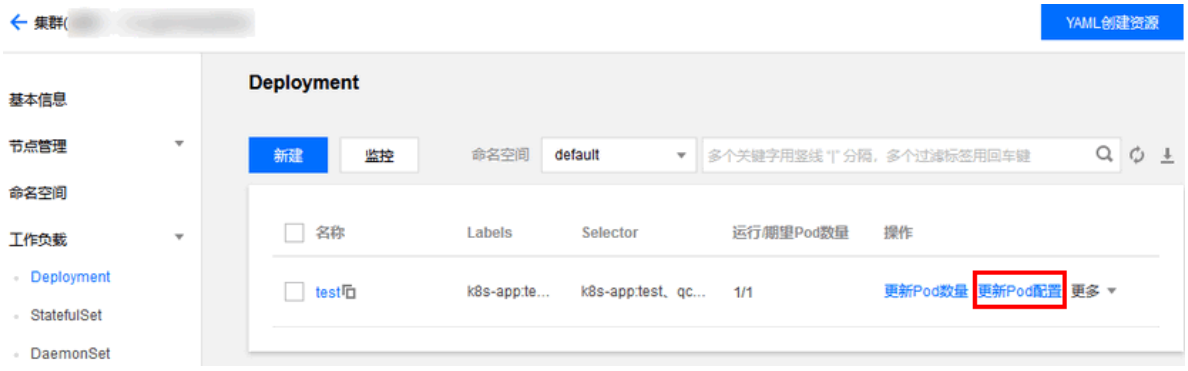


3. 在需要更新 YAML 的 Deployment 行中，单击**更多 > 编辑YAML**，进入更新 Deployment 页面。

4. 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 Pod 配置

1. 在集群管理页面，单击需要更新 Pod 配置的 Deployment 的集群 ID，进入待更新 Pod 配置的 Deployment 的集群管理页面。
2. 在需要更新 Pod 配置的 Deployment 行中，单击**更新Pod配置**。如下图所示：

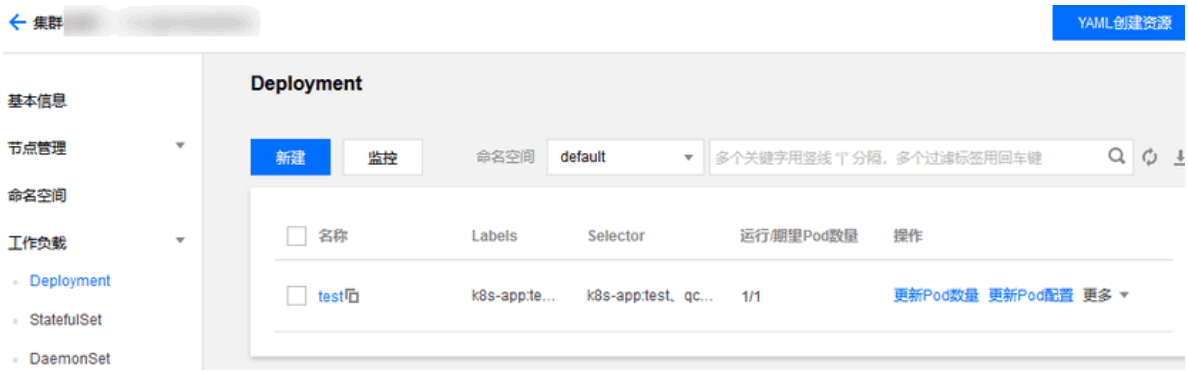


3. 在 “更新Pod配置” 页面，根据实际需求修改更新方式，设置参数。如下图所示：

4. 单击**完成**，即可更新 Pod 配置。

回滚 Deployment

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 单击需要回滚 Deployment 的集群 ID，进入待回滚 Deployment 的集群管理页面。如下图所示：



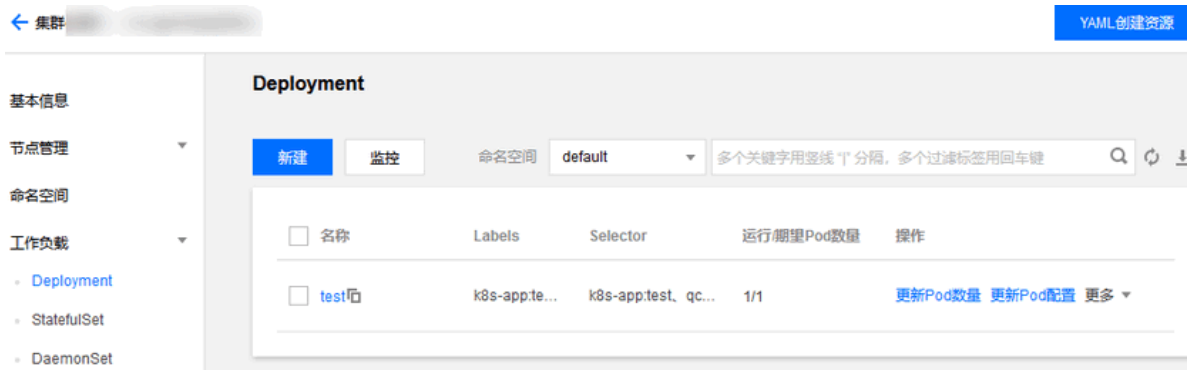
3. 单击需要回滚的 Deployment 名称，进入 Deployment 信息页面。

4. 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：

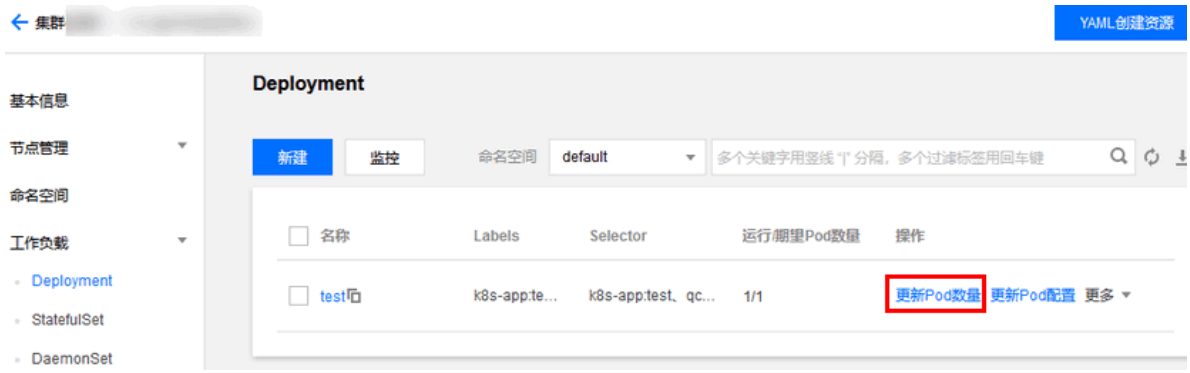
5. 在弹出的 “回滚资源” 提示框中，单击**确定**即可完成回滚。

调整 Pod 数量

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 单击需要调整 Pod 数量的 Deployment 的集群 ID，进入待调整 Pod 数量的 Deployment 的集群管理页面。如下图所示：



3. 在需要调整 Pod 数量的 Deployment 行中，单击**更新Pod数量**，进入更新 Pod 数量页面。如下图所示：



4. 根据实际需求调整 Pod 数量，单击**更新实例数目**即可完成调整。

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，可通过该参数设置腾讯云 TKE 的额外增强能力。
- **spec.replicas**：Deployment 管理的 Pod 数量。
- **spec.selector**：Deployment 管理 Seletor 选中的 Pod 的 Label。
- **spec.template**：Deployment 管理的 Pod 的详细模板配置。

更多参数详情可查看 Kubernetes Deployment 官方文档。

Kubectl 创建 Deployment

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

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

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

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

4. 执行以下命令，验证创建是否成功。

```
kubectl get deployments
```

返回类似以下信息，即表示创建成功。

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

Kubectl 更新 Deployment

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

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

方法一

执行以下命令，更新 Deployment。

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

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

方法二

执行以下命令，更新指定容器的镜像。

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

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

方法三

执行以下命令，滚动更新指定资源。

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

Kubectl 回滚 Deployment

1. 执行以下命令，查看 Deployment 的更新历史。

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

2. 执行以下命令，查看指定版本详情。

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

3. 执行以下命令，回滚到前一个版本。

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

如需指定回滚版本号，可执行以下命令。

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

Kubectl 调整 Pod 数量

手动更新 Pod 数量

执行以下命令，手动更新 Pod 数量。

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

自动更新 Pod 数量

前提条件

开启集群中的 HPA 功能。TKE 创建的集群默认开启 HPA 功能。

操作步骤

执行以下命令，设置 Deployment 的自动扩缩容。

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

Kubectl 删除 Deployment

执行以下命令，删除 Deployment。

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

StatefulSet 管理

最近更新时间：2022-04-18 14:14:25

简介

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

StatefulSet 控制台操作指引

创建 StatefulSet

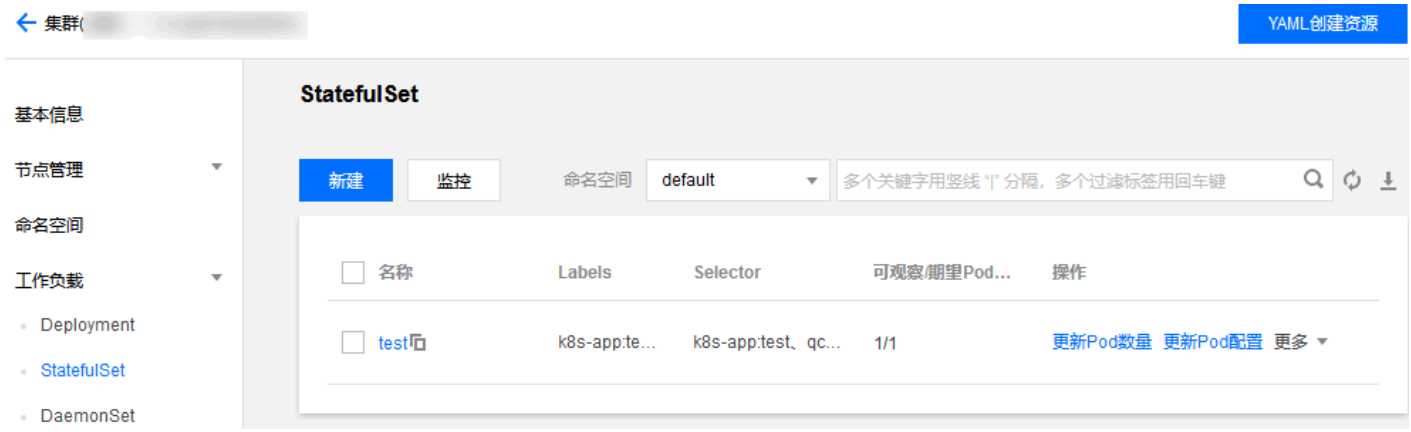
1. 登录容器服务控制台 ，选择左侧导航栏中的 **集群**。
2. 单击需要创建 StatefulSet 的集群 ID，进入待创建 StatefulSet 的集群管理页面。
3. 选择**工作负载** > **StatefulSet**，进入 StatefulSet 管理页面。如下图所示：

4. 单击**新建**，进入 “新建Workload” 页面。
   根据实际需求，设置 StatefulSet 参数。关键参数信息如下：
   - **工作负载名**：输入自定义名称。
   - **标签**：一个键 – 值对（Key-Value），用于对资源进行分类管理。
   - **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）**。
   - **数据卷（选填）**：为容器提供存储，目前支持临时路径、主机路径、云硬盘数据卷、文件存储 NFS、配置文件、PVC，还需挂载到容器的指定路径中。
   - **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（Tag）**：根据实际需求进行填写。
     - **镜像拉取策略**：提供以下3种策略，请按需选择。
       若不设置镜像拉取策略，当镜像版本为空或 latest 时，使用 Always 策略，否则使用 IfNotPresent 策略。
       - **Always**：总是从远程拉取该镜像。
       - **IfNotPresent**：默认使用本地镜像，若本地无该镜像则远程拉取该镜像。
       - **Never**：只使用本地镜像，若本地没有该镜像将报异常。
     - **CPU/内存限制**：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
     - **GPU 资源**：配置该工作负载使用的最少 GPU 资源。
     - **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**”和“**特权级**”等参数。
   - **镜像访问凭证**：容器镜像默认私有，在创建工作负载时，需选择实例对应的镜像访问凭证。
   - **实例数量**：根据实际需求选择调节方式，设置实例数量。
   - **节点调度策略**：可根据调度规则，将 Pod 调度到符合预期的 Label 的节点中。
   - **访问设置**：根据实际需求，设置 Service 参数。详情见 服务访问方式。
5. 单击**创建Workload**，完成创建。

更新 StatefulSet

更新 YAML

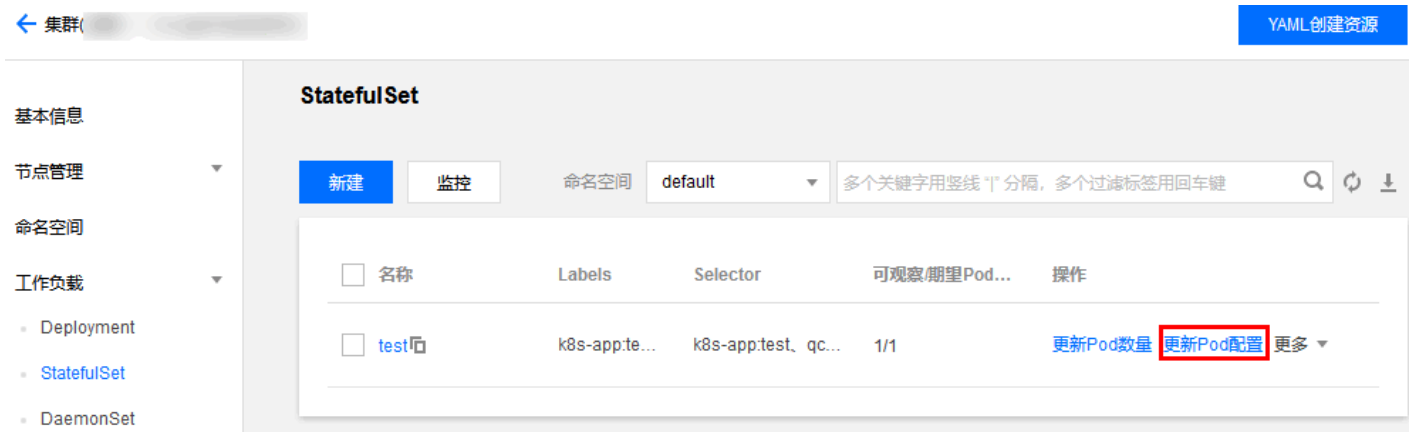
1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 单击需要更新 YAML 的集群 ID，进入待更新 YAML 的集群管理页面。
3. 选择**工作负载** > **StatefulSet**，进入 StatefulSet 信息页面。如下图所示：



4. 在需要更新 YAML 的 StatefulSet 行中，选择**更多** > **编辑YAML**，进入更新 StatefulSet 页面。
5. 在 “更新StatefulSet” 页面编辑 YAML，并单击**完成**即可更新 YAML。

更新 Pod 配置

1. 在集群管理页面，单击需要更新 Pod 配置的 StatefulSet 的集群 ID，进入待更新 Pod 配置的 StatefulSet 的集群管理页面。
2. 在需要更新 Pod 配置的 StatefulSet 行中，单击**更新Pod配置**。如下图所示：



3. 在 “更新Pod配置” 页面，根据实际需求修改更新方式，设置参数。如下图所示：

4. 单击**完成**，即可更新 Pod 配置。

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，可通过该参数设置腾讯云 TKE 的额外增强能力。
- **spec.template**：StatefulSet 管理的 Pod 的详细模板配置。
- **spec.volumeClaimTemplates**：提供创建 PVC&PV 的模板。

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

创建 StatefulSet

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

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

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

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

4. 执行以下命令，验证创建是否成功。

```
kubectl get StatefulSet
```

返回类似以下信息，即表示创建成功。

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

更新 StatefulSet

执行以下命令，查看 StatefulSet 的更新策略类型。

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

StatefulSet 有以下两种更新策略类型：

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

方法一

执行以下命令，更新 StatefulSet。

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

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

方法二

执行以下命令，更新指定容器的镜像。

```
kubectl patch statefulset <NAME> --type='json' -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/image", "value":"<new Image>"}]'
```

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

如果更新的 StatefulSet 是滚动更新方式的策略，可执行以下命令查看更新状态：

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

删除 StatefulSet

执行以下命令，删除 StatefulSet。

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

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

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

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

DaemonSet 管理

最近更新时间：2022-04-18 14:14:32

简介

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

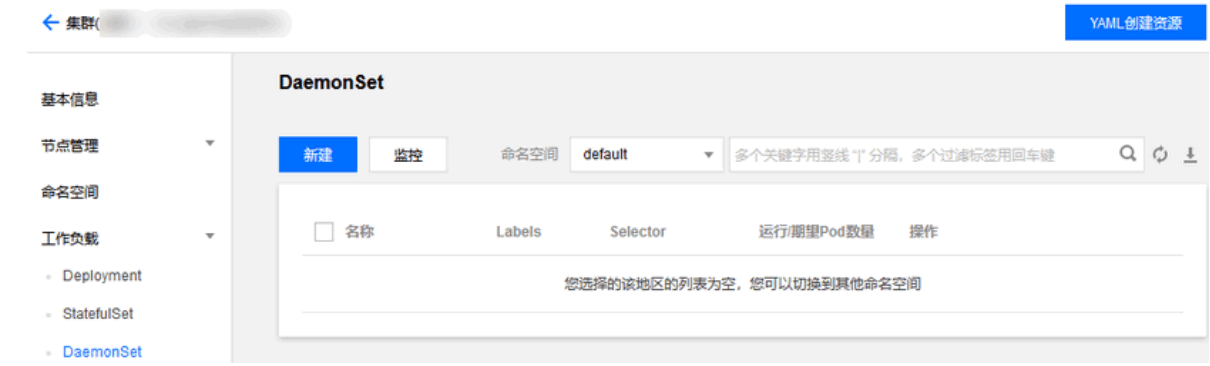
调度说明

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

DaemonSet 控制台操作指引

创建 DaemonSet

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 单击需要创建 DaemonSet 的集群 ID，进入待创建 DaemonSet 的集群管理页面。
3. 选择**工作负载** > **DaemonSet**，进入 DaemonSet 信息页面。如下图所示：

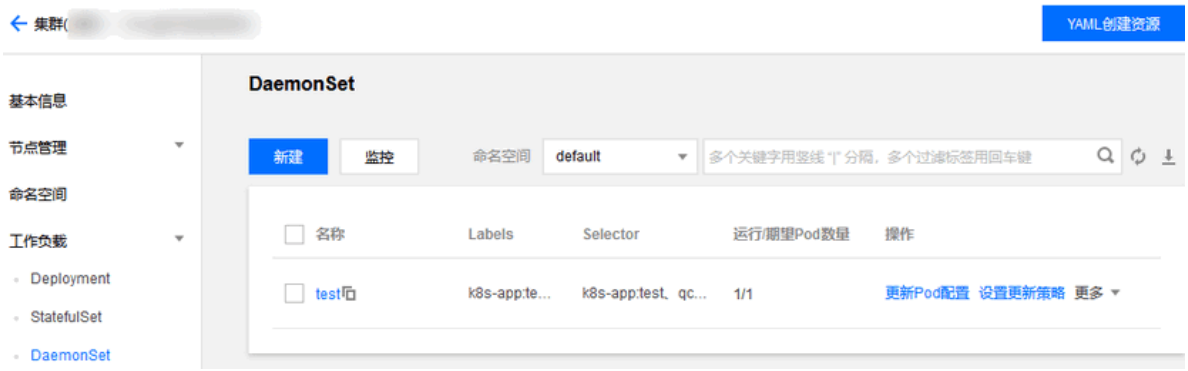


4. 单击**新建**，进入 “新建Workload” 页面。
   根据实际需求，设置 DaemonSet 参数。关键参数信息如下：
   - **工作负载名**：输入自定义名称。
   - **标签**：一个键 – 值对（Key-Value），用于对资源进行分类管理。
   - **命名空间**：根据实际需求进行选择。
   - **类型**：选择**DaemonSet（在每个主机上运行Pod）**。
   - **数据卷（选填）**：为容器提供存储，目前支持临时路径、主机路径、云硬盘数据卷、文件存储 NFS、配置文件、PVC，还需挂载到容器的指定路径中。
   - **实例内容器**：根据实际需求，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（Tag）**：根据实际需求进行填写。
     - **镜像拉取策略**：提供以下3种策略，请按需选择。
       若不设置镜像拉取策略，当镜像版本为空或 latest 时，使用 Always 策略，否则使用 IfNotPresent 策略。
       - **Always**：总是从远程拉取该镜像。
       - **IfNotPresent**：默认使用本地镜像，若本地无该镜像则远程拉取该镜像。
       - **Never**：只使用本地镜像，若本地没有该镜像将报异常。
     - **CPU/内存限制**：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
     - **GPU 资源**：配置该工作负载使用的最少 GPU 资源。
     - 高级设置：可设置 “**工作目录**”，“**运行命令**”，“**运行参数**”，“**容器健康检查**”，“**特权级**”等参数。
   - **镜像访问凭证**：容器镜像默认私有，在创建工作负载时，需选择实例对应的镜像访问凭证。
   - **节点调度策略**：可根据调度规则，将 Pod 调度到符合预期的 Label 的节点中。
5. 单击**创建Workload**，完成创建。

更新 DaemonSet

更新 YAML

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 单击需要更新 YAML 的集群 ID，进入待更新 YAML 的集群管理页面。
3. 选择**工作负载** > **DaemonSet**，进入 DaemonSet 信息页面。如下图所示：

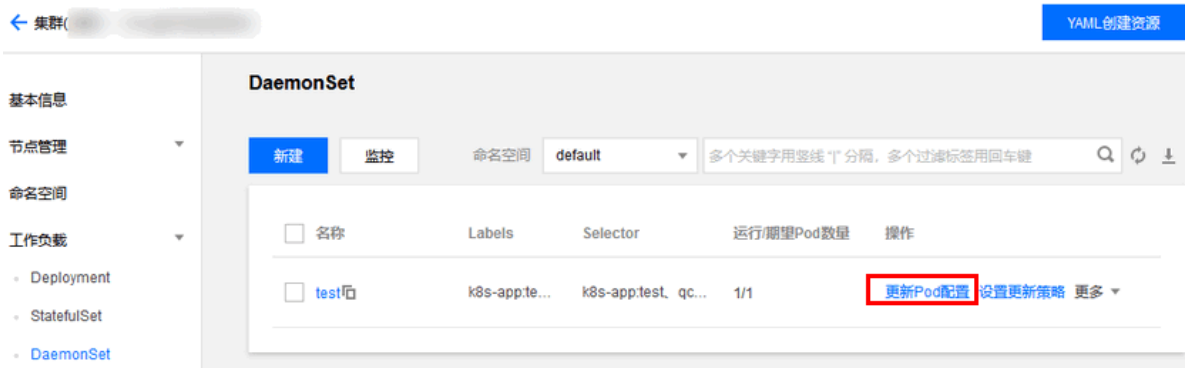


4. 在需要更新 YAML 的 DaemonSet 行中，选择**更多** > **编辑YAML**，进入更新 DaemonSet 页面。
5. 在 “更新DaemonSet” 页面编辑 YAML，单击**完成**即可更新 YAML。

更新 Pod 配置

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

1. 在集群管理页面，单击需要更新 Pod 配置的 DaemonSet 的集群 ID，进入待更新 Pod 配置的 DaemonSet 的集群管理页面。
2. 在需要更新 Pod 配置的 DaemonSet 行中，单击**更新Pod配置**。如下图所示：



3. 在 “更新Pod配置” 页面，根据实际需求修改更新方式，设置参数。如下图所示：

4. 单击**完成**，即可更新 Pod 配置。

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
```

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

- **kind**：标识 DaemonSet 资源类型。
- **metadata**：DaemonSet 的名称、Label等基本信息。
- **metadata.annotations**：DaemonSet 的额外说明，可通过该参数设置腾讯云 TKE 的额外增强能力。
- **spec.template**：DaemonSet 管理的 Pod 的详细模板配置。

更多可查看 Kubernetes DaemonSet 官方文档。

Kubectl 创建 DaemonSet

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

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

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

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

4. 执行以下命令，验证创建是否成功。

```
kubectl get DaemonSet
```

返回类似以下信息，即表示创建成功。

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

Kubectl 更新 DaemonSet

执行以下命令，查看 DaemonSet 的更新策略类型。

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

DaemonSet 有以下两种更新策略类型：

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

方法一

执行以下命令，更新 DaemonSet。

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

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

方法二

执行以下命令，更新指定容器的镜像。

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

建议保持 DaemonSet 的其他参数不变，业务更新时，仅更新容器镜像。

Kubectl 回滚 DaemonSet

1. 执行以下命令，查看 DaemonSet 的更新历史。

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

2. 执行以下命令，查看指定版本详情。

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

3. 执行以下命令，回滚到前一个版本。

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

   如需指定回滚版本号，可执行以下命令。

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

Kubectl 删除 DaemonSet

执行以下命令，删除 DaemonSet。

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

CronJob 管理

最近更新时间：2022-04-18 14:14:11

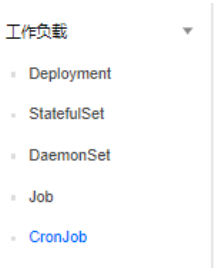
简介

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。

CronJob 控制台操作指引

创建 CronJob

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**，进入集群管理页面。
3. 单击需要创建 CronJob 的集群 ID，进入待创建 CronJob 的集群管理页面。
4. 选择**工作负载** > **CronJob**，进入 CronJob 信息页面。如下图所示：



5. 单击**新建**，进入 “新建Workload” 页面。
6. 根据实际需求，设置 CronJob 参数。关键参数信息如下：
   - **工作负载名**：自定义。
   - **标签**：一个键 – 值对（Key-Value），用于对资源进行分类管理。
   - **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）” 。
   - **定时规则**：根据业务需求选择任务的定期执行策略。
   - **保留成功 Job 数**：对应.spec.successfulJobsHistoryLimit，详情见 Jobs History Limits。
   - **保留失败 Job 数**：对应.spec.failedJobsHistoryLimit，详情见 Jobs History Limits。
   - **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。
   - **数据卷（选填）**：为容器提供存储，目前支持临时路径、主机路径、云硬盘数据卷、文件存储 NFS、配置文件、PVC，还需挂载到容器的指定路径中。
   - **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。
     - **镜像拉取策略**：提供以下3种策略，请按需选择。
       若不设置镜像拉取策略，当镜像版本为空或 latest 时，使用 Always 策略，否则使用 IfNotPresent 策略。
       - **Always**：总是从远程拉取该镜像。
       - **IfNotPresent**：默认使用本地镜像，若本地无该镜像则远程拉取该镜像。
       - **Never**：只使用本地镜像，若本地没有该镜像将报异常。
     - **CPU/内存限制**：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
     - **GPU 资源**：配置该工作负载使用的最少 GPU 资源。
     - **高级设置**：可设置 “工作目录”，“运行命令”，“运行参数”，“容器健康检查”，“特权级” 等参数。
   - **镜像访问凭证**：容器镜像默认私有，在创建工作负载时，需选择实例对应的镜像访问凭证。
   - **节点调度策略**：可根据调度规则，将 Pod 调度到符合预期的 Label 的节点中。
7. 单击**创建Workload**，完成创建。

查看 CronJob 状态

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**，进入集群管理页面。
3. 单击需要查看 CronJob 状态的集群 ID，进入待查看 CronJob 状态的集群管理页面。
4. 选择**工作负载** > **CronJob**，进入 CronJob 信息页面。
5. 单击需要查看状态的 CronJob 名称，即可查看 CronJob 详情。

Kubectl 操作 CronJob 指引

YAML 示例

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

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

创建 CronJob

方法一

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

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

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

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

方法二

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

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

2. 执行以下命令，验证创建是否成功。

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

返回类似以下信息，即表示创建成功。

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

删除 CronJob

> ⚠ 注意：
>
> - 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]
```

Job 管理

最近更新时间：2022-04-18 14:14:19

简介

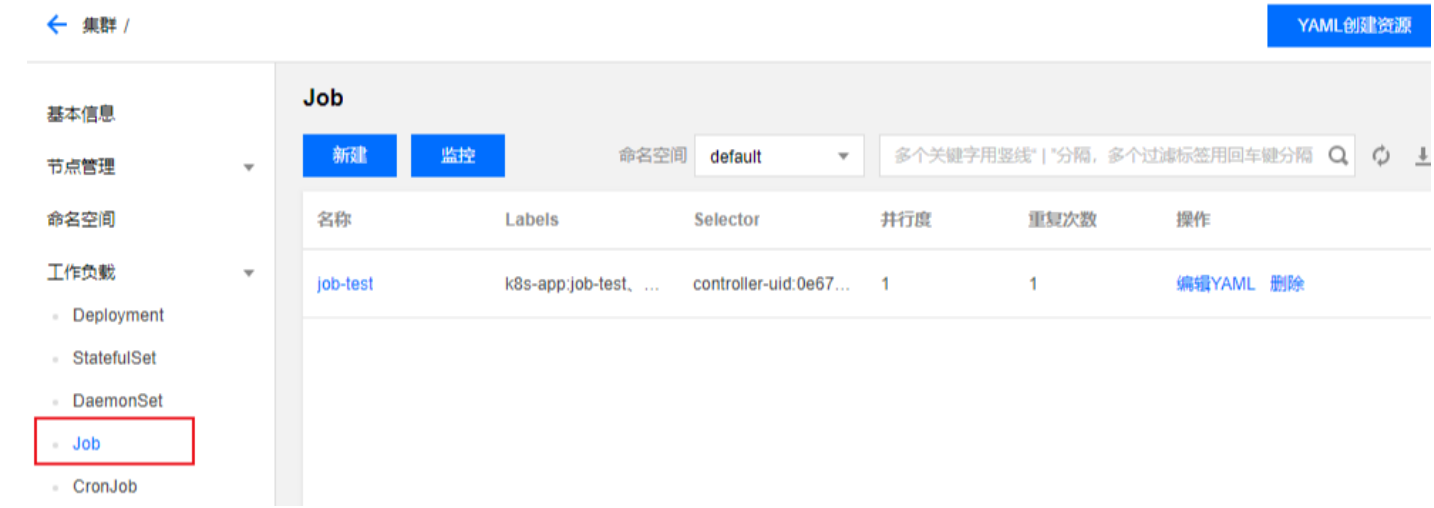
Job 控制器会创建 1-N 个 Pod，这些 Pod 按照运行规则运行，直至运行结束。Job 可用于批量计算、数据分析等场景。通过设置重复执行次数、并行度、重启策略等满足业务诉求。

Job 执行完成后，不再创建新的 Pod，也不会删除 Pod，您可在 “日志” 中查看已完成的 Pod 的日志。如果您删除了 Job，Job 创建的 Pod 也会同时被删除，将无法查看该 Job 创建的 Pod 的日志。

Job 控制台操作指引

创建 Job

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**，进入集群管理页面。
3. 单击需要创建 Job 的集群 ID，进入待创建 Job 的集群管理页面。
4. 选择 **工作负载** > **Job**，进入 Job 信息页面。如下图所示：



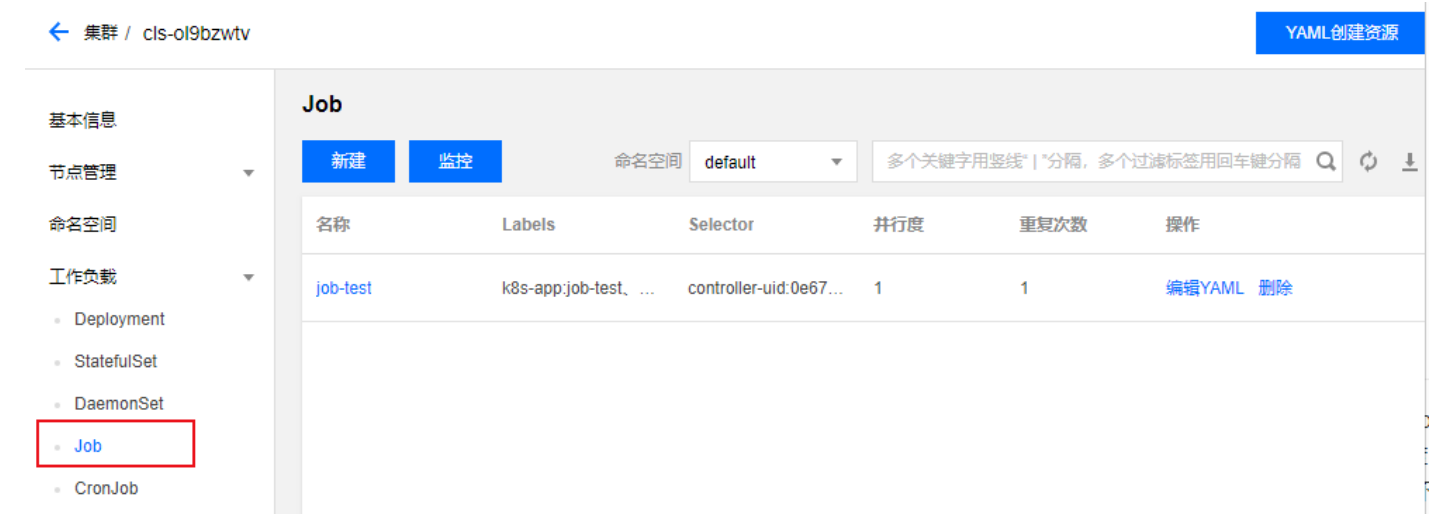
5. 单击**新建**，进入 “新建Workload” 页面。如下图所示：
6. 根据实际需求，设置 Job 参数。关键参数信息如下：
   - **工作负载名**：自定义。
   - **标签**：一个键 - 值对（Key-Value），用于对资源进行分类管理。
   - **命名空间**：根据实际需求进行选择。
   - **类型**：选择 “Job（单次任务）”。
   - **Job设置**：根据实际需求，为 Job 的一个 Pod 设置一个或多个不同的容器。
     - **重复次数**：设置 Job 管理的 Pod 需要重复执行的次数。
     - **并行度**：设置 Job 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。
   - **数据卷（选填）**：为容器提供存储，目前支持临时路径、主机路径、云硬盘数据卷、文件存储 NFS、配置文件、PVC，还需挂载到容器的指定路径中。
   - **实例内容器**：根据实际需求，为 Job 的一个 Pod 设置一个或多个不同的容器。
     - **名称**：自定义。
     - **镜像**：根据实际需求进行选择。
     - **镜像版本**：根据实际需求进行填写。
     - **CPU/内存限制**：可根据 Kubernetes 资源限制 进行设置 CPU 和内存的限制范围，提高业务的健壮性。
     - **GPU 资源**：配置该工作负载使用的最少 GPU 资源。
     - **高级设置**：可设置 “工作目录”，“运行命令”，“运行参数”，“容器健康检查”，“特权级” 等参数。
   - **镜像访问凭证**：容器镜像默认私有，在创建工作负载时，需选择实例对应的镜像访问凭证。

- **节点调度策略**：可根据调度规则，将 Pod 调度到符合预期的 Label 的节点中。

7. 单击**创建Workload**，完成创建。

查看 Job 状态

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**，进入集群管理页面。
3. 单击需要查看 Job 状态的集群 ID，进入待查看 Job 状态的集群管理页面。
4. 选择 “工作负载” > “Job”，进入 Job 信息页面。如下图所示：



5. 单击需要查看状态的 Job 名称，即可查看 Job 详情。

删除 Job

Job 执行完成后，不再创建新的 Pod，也不会删除 Pod，您可在 “日志” 中查看已完成的 Pod 的日志。如果您删除了 Job，Job 创建的 Pod 也会同时被删除，将查看不到该 Job 创建的 Pod 的日志。

Kubectl 操作 Job 指引

YAML 示例

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

- kind：标识 Job 资源类型。
- metadata：Job 的名称、Label等基本信息。
- metadata.annotations：Job 的额外说明，可通过该参数设置腾讯云 TKE 的额外增强能力。
- spec.completions：Job 管理的 Pod 重复执行次数。

- spec.parallelism：Job 并行执行的 Pod 数。
- spec.template：Job 管理的 Pod 的详细模板配置。

创建 Job

1. 参考 YAML 示例，准备 Job YAML 文件。
2. 安装 Kubectl，并连接集群。操作详情请参考 通过 Kubectl 连接集群。
3. 创建 Job YAML 文件。

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

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

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

4. 执行以下命令，验证创建是否成功。

```
kubectl get job
```

返回类似以下信息，即表示创建成功。

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

删除 Job

执行以下命令，删除 Job。

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

设置工作负载的资源限制

最近更新时间：2022-04-22 17:19:52

请求（Request）与限制（Limit）

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

> 注意：
> 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。

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

内存限制说明

内存资源只允许限制容器最大可使用内存量。以 MiB 为单位，允许为小数。

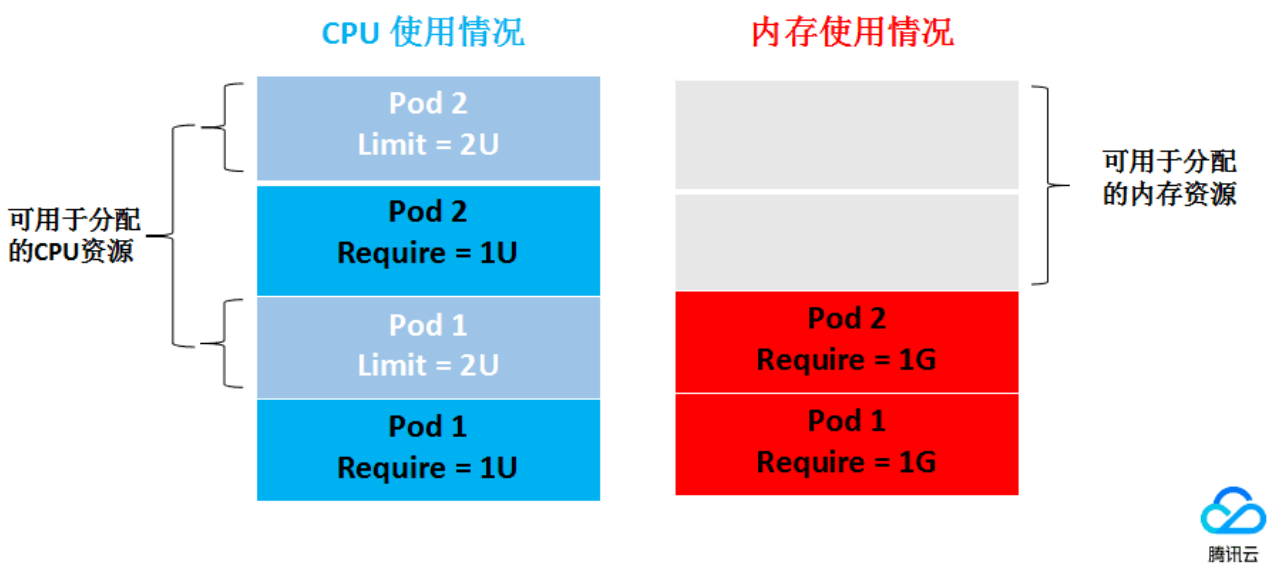
> 注意：
> - 内存 Request 作为调度时的依据，在创建时为该容器在节点上分配内存，称为 “已分配内存” 资源。
> - 内存资源为不可伸缩资源。当节点上所有容器使用内存均超量时，存在 OOM（Out Of Memory，即内存溢出）的风险。不设置 Limit 时，容器可以使用节点所有可使用资源，会导致其它容器的资源被占用，且该类型的容器所在的 Pod 容易被驱逐，不建议使用。建议 Limit = Request。

CPU 使用量和 CPU 使用率

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

使用示例

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



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

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

服务资源限制推荐

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

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

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

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

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

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

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

相关文档

容器的 Request 及 Limit 需根据服务类型、需求及场景进行灵活设置。详情可参见 设置 Request 与 Limit。

设置工作负载的调度规则

最近更新时间：2022-04-18 14:13:14

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。

使用方法

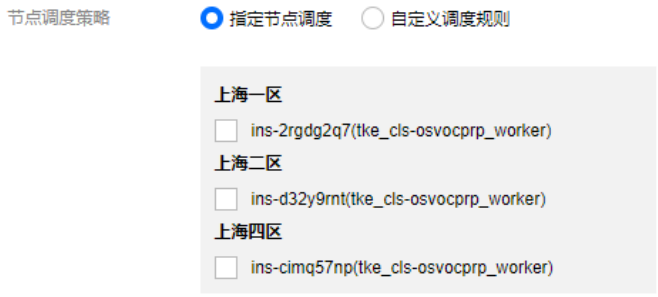
前置条件

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

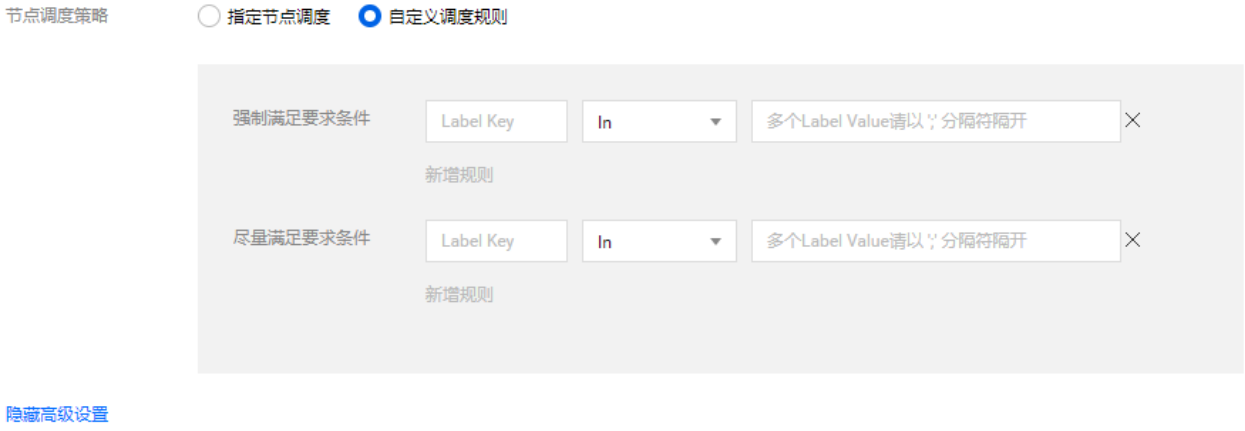
设置调度规则

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

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



- **自定义调度规则**：可自定义实例（ Pod ）调度规则，匹配实例标签。



自定义调度规则包含以下两种模式：

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

自定义调度规则均可以添加多条调度规则， 各规则操作符的含义如下：

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

原理介绍

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

设置工作负载的健康检查

最近更新时间：2022-06-09 11:14:34

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

健康检查类别

健康检查分为以下类别：

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

健康检查方式

TCP 端口探测

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

HTTP 请求探测

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

执行命令检查

执行命令检查是一种强大的检查方式，该方式要求用户指定一个容器内的可执行命令，集群会周期性地在容器内执行该命令。如果命令的返回结果是0，检查成功，否则检查失败。
对于 TCP 端口探测 和 HTTP 请求探测，都可以通过执行命令检查的方式来替代：

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

注意事项

- 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。
- 若执行的命令是一个 shell 脚本，则不能直接指定脚本作为执行命令，需要加上脚本的解释器。例如，脚本是 /data/scripts/health_check.sh，那么使用执行命令检查时，指定的程序应为：

```
sh
```

```
/data/scripts/health_check.sh
```

设置步骤以通过 容器服务控制台 创建 Deployment 为例：
i. 在集群的 “Deployment” 页面，单击**新建**。
ii. 进入“新建Workload”页面，选择“容器内实例”模块下方的**显示高级设置**。
iii. 在“容器健康检查”中，以选择**存活检查**为例，设置以下参数。

- **检查方法**：选择“执行命令检查”。
- **执行命令**：输入以下内容。

```
sh
```

```
/data/scripts/health_check.sh
```

iv. 其余参数设置请参考 Deployment 管理。

其它公共参数

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

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

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

设置工作负载的运行命令和参数

最近更新时间：2022-01-17 15:06:32

概述

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

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

工作目录说明

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

命令和参数的使用

如何将 docker run 命令适配到腾讯云容器服务，请参见 docker run 参数适配。

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

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

⚠ 注意：

- Docker entrypoint 对应容器服务控制台上的运行命令，Docker run 的 CMD 参数对应容器服务控制台上的运行参数。当有多个运行参数时，需在容器服务的运行参数中输入参数，且每个参数单独一行。
- 通过 容器服务控制台 设置容器运行命令和参数的示例请参考 Command 和 Args。

使用 TCR 企业版实例内容器镜像创建工作负载

最近更新时间：2022-06-09 11:35:31

操作场景

腾讯云容器镜像服务（Tencent Container Registry，TCR）企业版面向具有严格数据安全及合规性要求、业务分布在多个地域、集群规模庞大的企业级容器客户，提供企业级的独享镜像安全托管服务。相较于个人版服务，企业版支持容器镜像安全扫描、跨地域自动同步、Helm Chart 托管、网络访问控制等特性，详情请参见 容器镜像服务。

本文介绍如何在容器服务 TKE 中，使用容器镜像服务 TCR 内托管的私有镜像进行应用部署。

前提条件

在使用 TCR 内托管的私有镜像进行应用部署前，您需要完成以下准备工作：

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

操作步骤

准备容器镜像

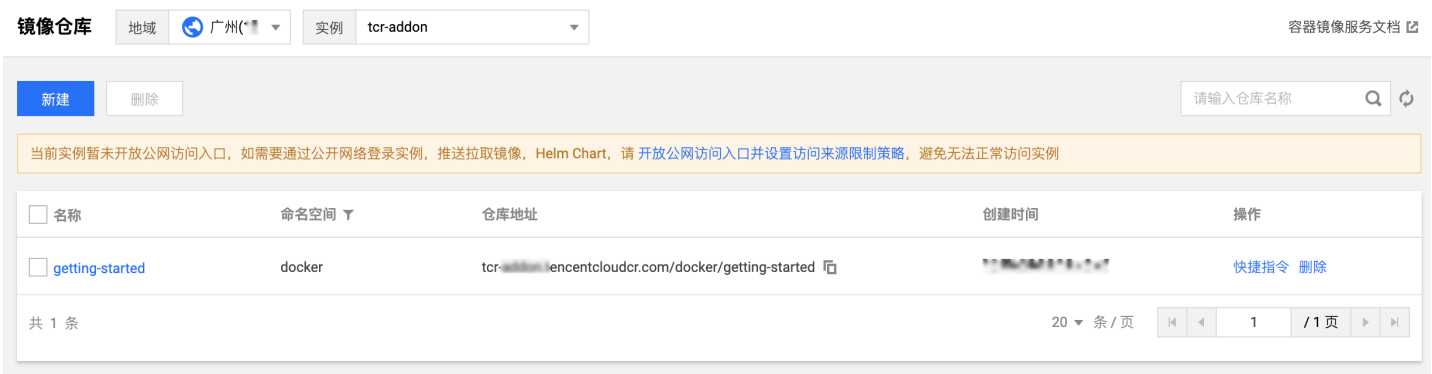
创建命名空间

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

创建镜像仓库（可选）

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

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



推送容器镜像

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

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

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

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

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

配置 TKE 集群访问 TCR 实例

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

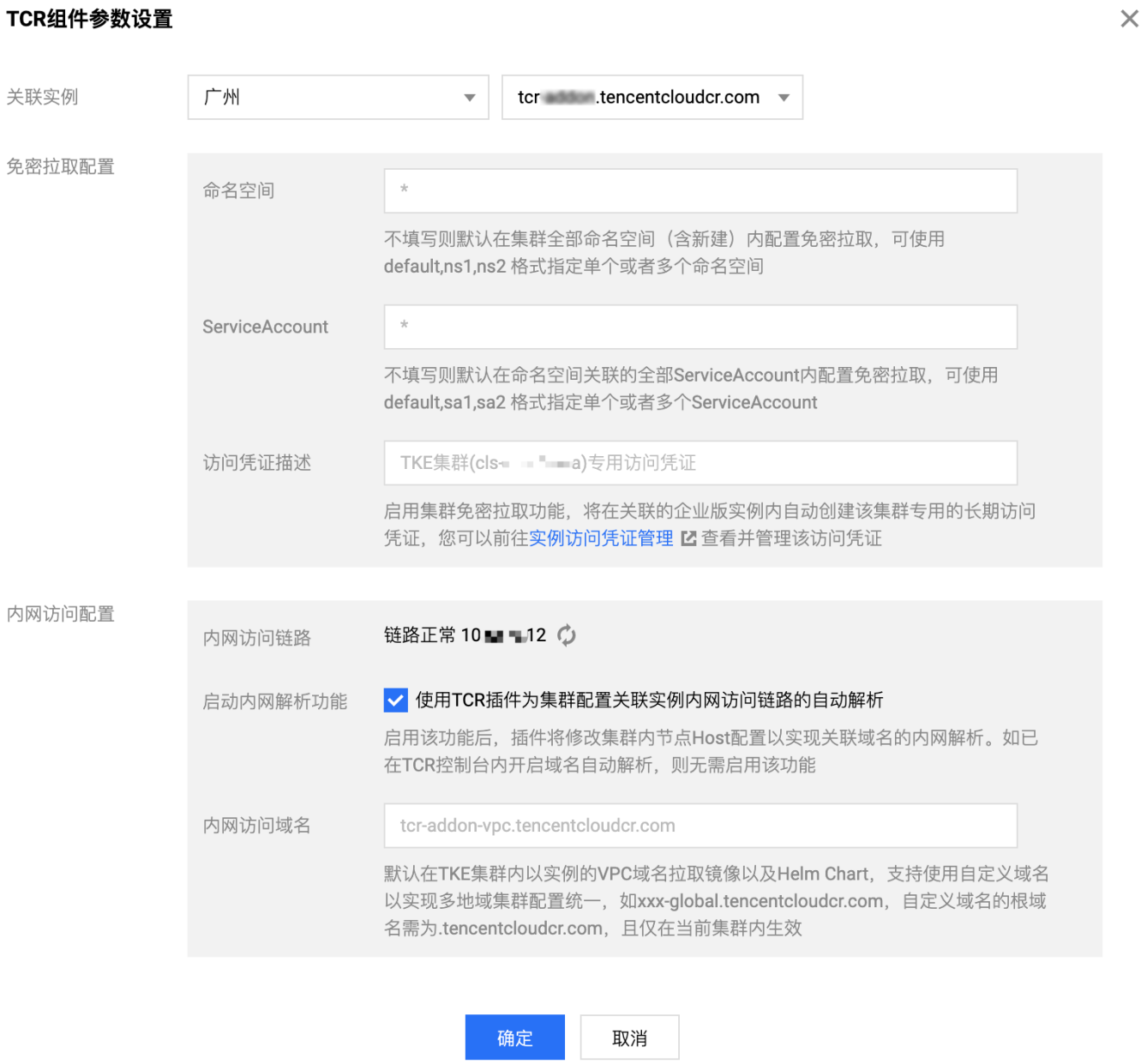
使用 TCR 扩展组件进行快速配置（推荐）

1. 登录容器服务控制台 ，选择左侧导航栏中的 集群。
2. 在“集群管理”页面，选择集群 ID，进入集群详情页。
3. 在集群详情页面，选择左侧**组件管理**，进入“组件管理”页面，并单击**新建**。
4. 在“新建扩展组件”页面，选择 “TCR” 组件。如下图所示：

> 说明：
> 当前 TCR 组件暂只支持 K8S 版本为 1.12、1.14、1.16、1.18、1.20 的集群，如集群版本暂不支持，请采用手动配置方式，或升级集群版本。

- 单击**查看详情**了解组件功能及配置说明。
- 单击**参数配置**开始配置组件。

5. 在“TCR组件参数设置”页面，参考**查看详情**中介绍的组件配置方式，配置相关参数。如下图所示：



- **关联实例**：选择与集群同地域的 TCR 实例。
- **免密拉取配置**：可采用默认配置。
- **内网访问配置**：可选功能，在 TCR 实例接入集群所在 VPC 并开启自动解析后，集群内节点可内网访问 TCR 实例，无需使用本功能。由于 TCR 侧自动解析功能依赖于 PrivateDNS，若当前集群所在地暂未支持 PrivateDNS 产品，可使用本配置实现内网访问。如内网访问链路中未展示为“链路正常”，请参考 内网访问控制，配置 TCR 实例与 TKE 集群所在私有网络 VPC 的内网链路。

6. 点击**确定**返回组件选择界面。
7. 在组件选择界面单击**完成**，开始为集群安装 TCR 扩展组件。

8. 组件安装完成后，集群将具备内网免密拉取该关联实例内镜像的能力，无需额外配置。如下图所示：

手动配置内网访问及访问凭证

1. 配置内网访问

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

     在创建 TKE 集群的“云服务器配置”步骤中，选择**高级设置**并在“节点启动配置”中输入如下内容：

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

   - **为已有集群配置节点 Host**

     登录集群各个节点，并执行以下命令：

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

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

2. 配置访问凭证

新建命名空间时参考以下步骤，下发访问凭证：

1. 登录容器服务控制台 ，选择左侧导航栏中的 **集群**。
2. 在“集群管理”页面，选择集群 ID，进入集群详情页。
3. 选择左侧的**命名空间**，进入 “Namespace” 页面并单击**新建**。

4. 进入“新建Namespace” 页面，勾选“自动下发容器镜像服务企业版访问凭证”，并选择该集群需访问的 TCR 实例。如下图所示：

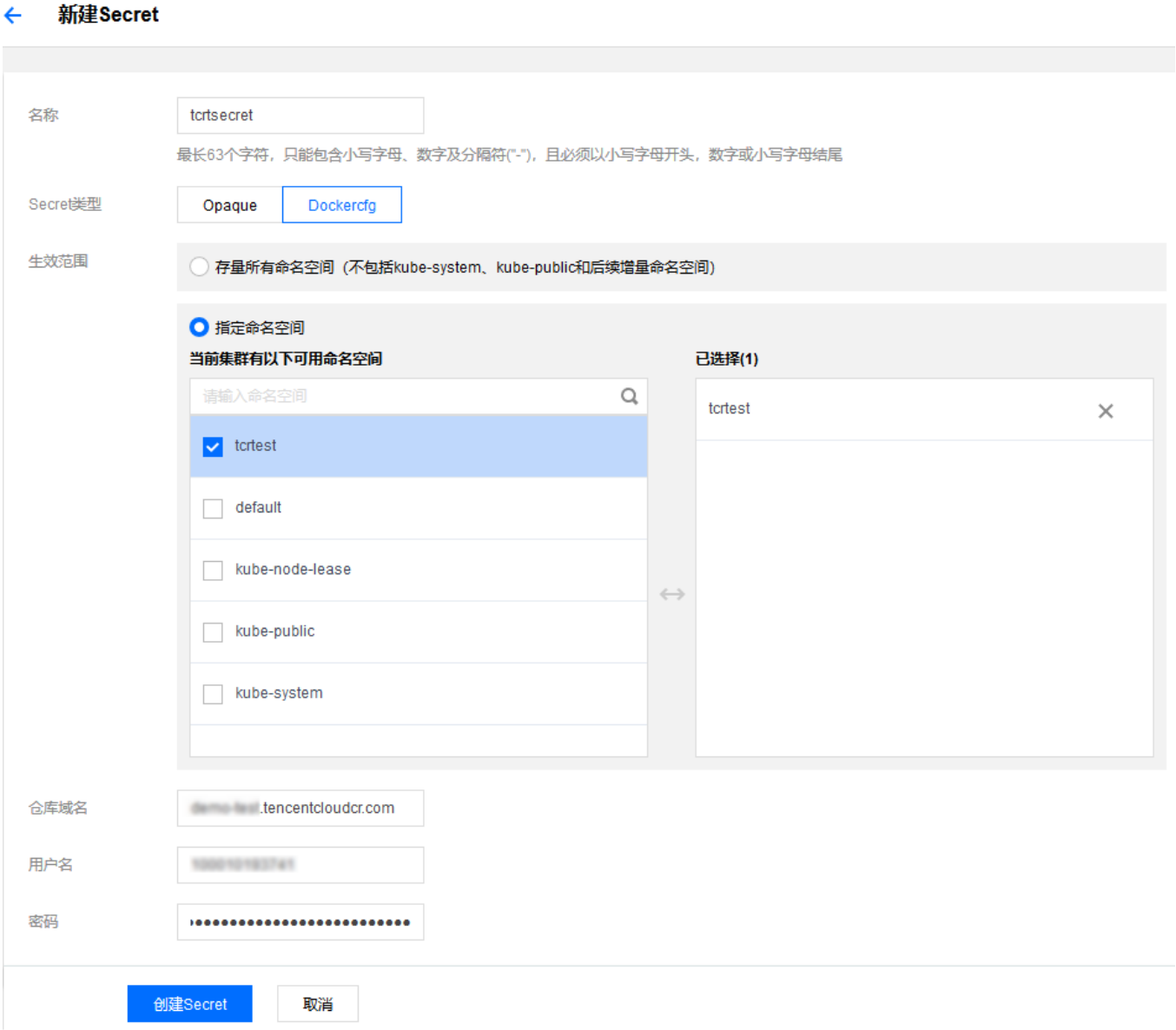
5. 单击**创建Namespace**进行创建。

创建完成后，该实例的访问凭证将自动下发至该命名空间。可选择左侧的**配置管理 > Secret**，进入 “Secret” 页面即可查看该访问凭证。例如 1000090225xx-tcr-m3ut3qxx-dockercfg。其中，1000090225xx 为创建命名空间的子账号 UIN，tcr-m3ut3qxx 为所选实例的实例 ID。

参考以下步骤，向已有命名空间下发访问凭证：

1. 参考 获取实例访问凭证，获取用户名及密码。
2. 在集群详情页，选择左侧的**配置管理 > Secret**，进入 “Secret” 页面。

3. 在“Secret”页面单击**新建**进入“新建Secret”页面，参考以下信息下发访问凭证。如下图所示：



主要参数信息如下：

- **Secret类型**：选择**Dockercfg**。
- **生效范围**：勾选需下发凭证的命名空间。
- **仓库域名**：填写 TCR 实例的访问域名。
- **用户名、密码**：填写 步骤1 已获取的用户名及密码。

4. 单击**创建Secret**即可完成下发。

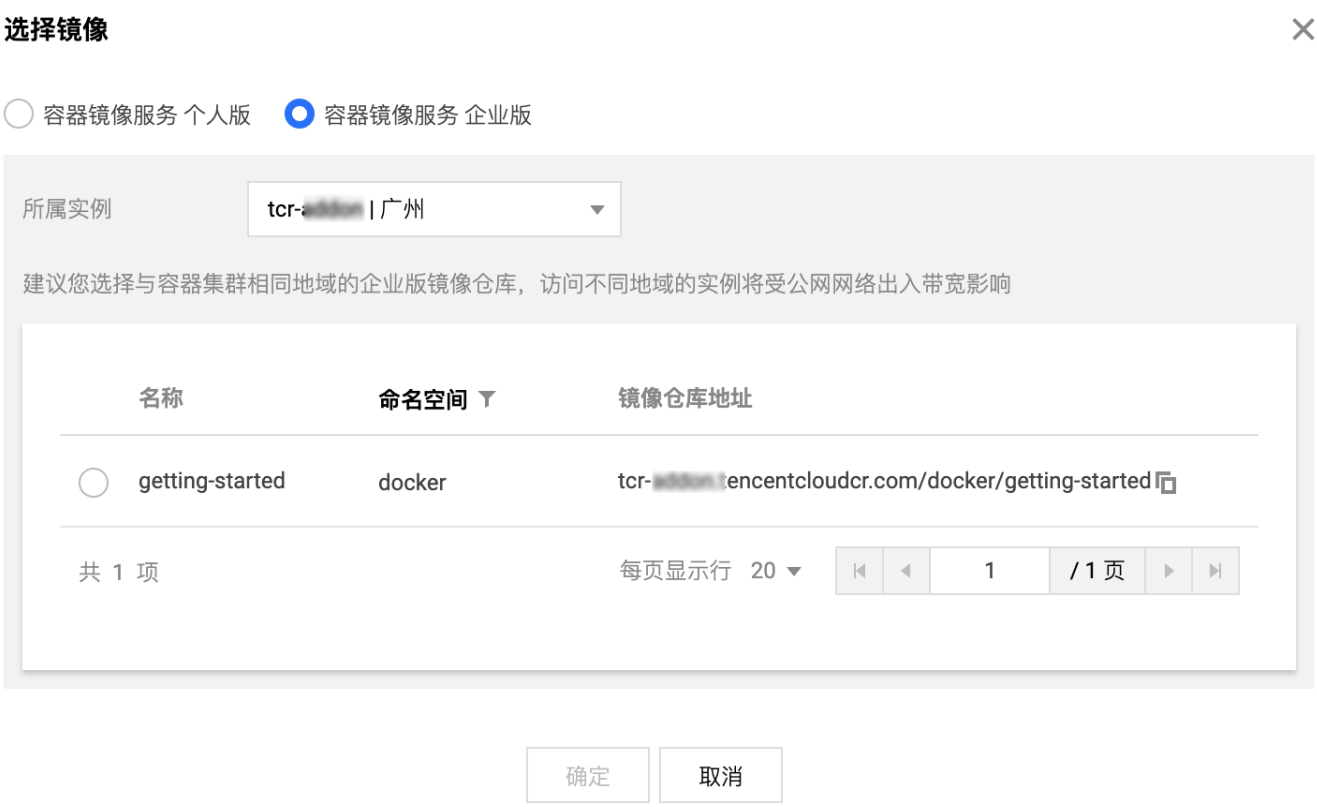
使用 TCR 实例内容器镜像创建工作负载

1. 在集群详情页面，选择左侧**工作负载** > **Deployment**。
2. 进入“Deployment”页面，并单击**新建**。
3. 进入“新建Workload”页面，参考以下信息创建工作负载。

主要参数信息如下，其他参数请按需设置：

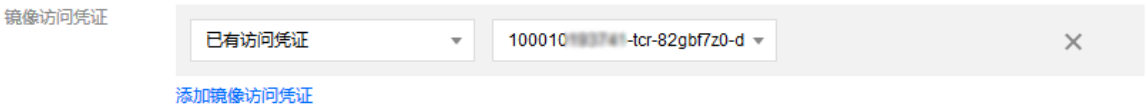
- **命名空间**：选择已下发访问凭证的命名空间。
- **实例内容器**：

- **镜像：单击选择镜像，并在弹出的“选择镜像”窗口中，选择 容器镜像服务 企业版，再根据需要选择地域、实例和镜像仓库。如下图所示：**

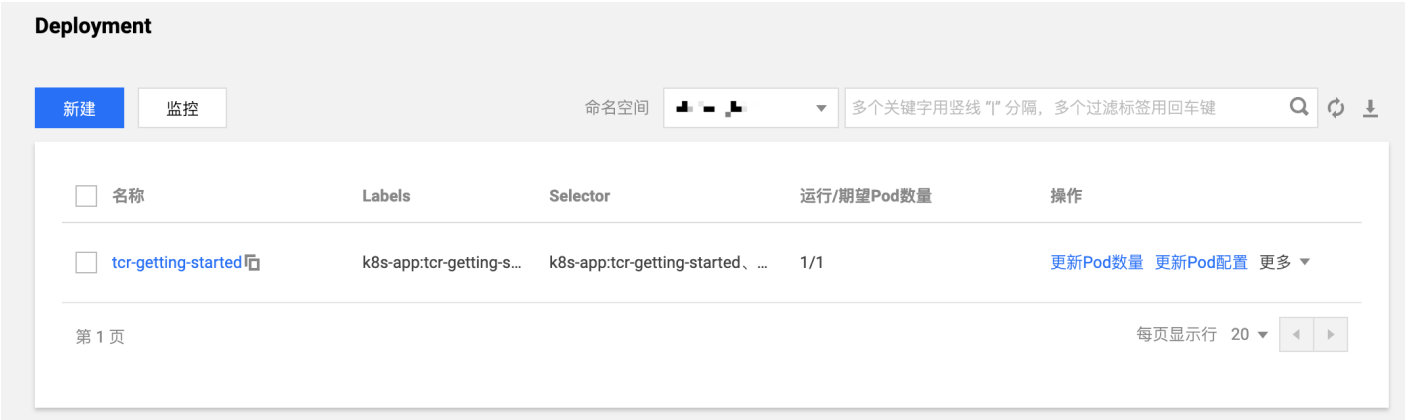


- **镜像版本：选择好镜像后，单击选择镜像版本，在弹出的“选择镜像版本”窗口中，根据需要选择该镜像仓库的某个版本。若不选择则默认为latest。**

- **镜像访问凭证：**
  - **集群已安装 TCR 扩展组件：无需配置。**
  - **集群未安装 TCR 扩展组件：选择添加镜像访问凭证，并选择 配置访问凭证 步骤中已下发的访问凭证。如下图所示：**



4. **完成其他参数设置后，单击创建workload后查看该工作负载的部署进度。**
   部署成功后，可在 “Deployment” 页面查看该工作负载的“运行/期望Pod数量”为“1/1”。如下图所示：



自动伸缩
自动伸缩基本操作

最近更新时间：2022-06-15 11:04:32

操作场景

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

工作原理

HPA 后台组件会每隔15秒向腾讯云云监控拉取容器和 Pod 的监控指标，然后根据当前指标数据、当前副本数和该指标目标值进行计算，计算所得结果作为服务的期望副本数。当期望副本数与当前副本数不一致时，HPA 会触发 Deployment 进行 Pod 副本数量调整，从而达到自动伸缩的目的。
以 CPU 利用率为例，假设当前有2个实例， 平均 CPU 利用率（当前指标数据）为90%，自动伸缩设置的目标 CPU 为60%， 则自动调整实例数量为：90% × 2 / 60% = 3个。

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

注意事项

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

前提条件

- 已 注册腾讯云账户。
- 已登录 腾讯云容器服务控制台 。
- 已创建集群。关于创建集群，详情请参见 创建集群。

操作步骤

开启自动扩缩容

可以通过以下三种方式开启自动扩缩容。

通过设置实例数量调节

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

3. 在“新建Workload”页面，设置实例数量为**自动调节**。如下图所示：

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

通过新建自动伸缩组

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

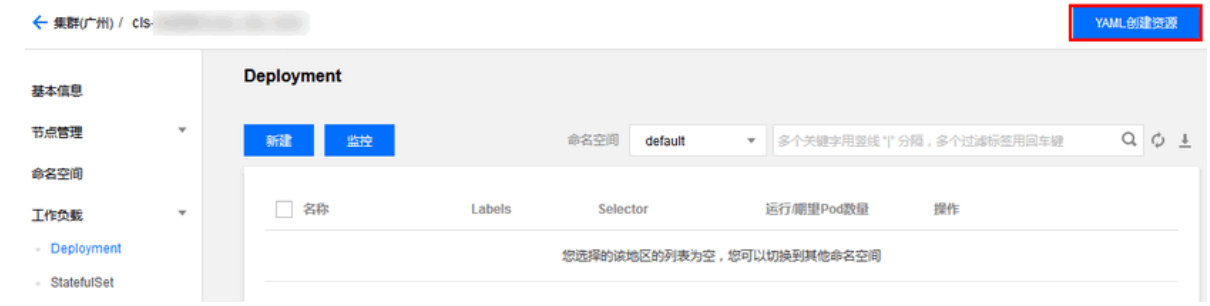
4. 在“新建HPA”页面，根据以下提示，进行 HPA 配置。如下图所示：

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

5. 单击**创建HPA**，完成 HPA 的创建。

通过 YAML 创建

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



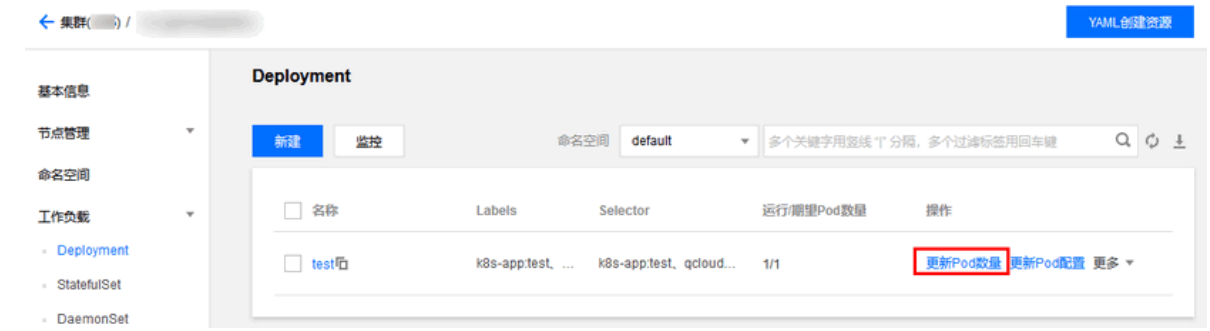
4. 在“YAML创建资源”页面，根据实际需求编辑内容，单击**完成**，即可新建 HPA 。

更新自动扩缩容规则

可以通过以下三种方式更新服务自动扩缩容规则。

通过更新 Pod 数量

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



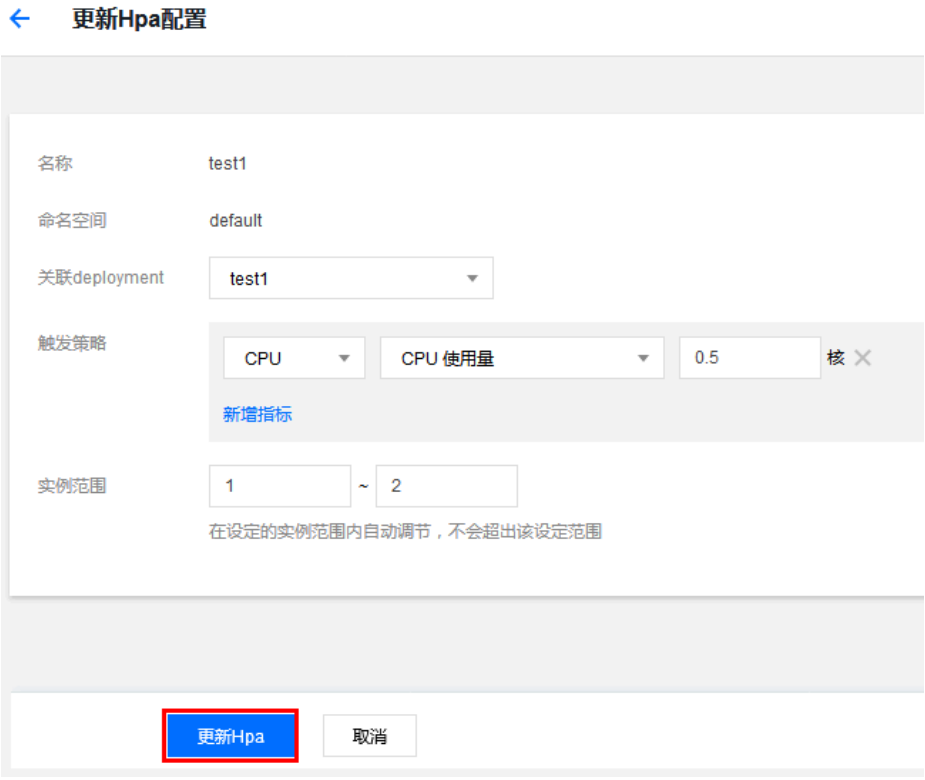
3. 在“更新Pod数量”页面，根据实际需求进行设置，并单击**更新实例数目**。如下图所示：

通过修改 Hpa 配置

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

3. 在“HorizontalPodAutoscaler”页面，单击需要更新配置的 HPA 所在行右侧的**修改配置**。如下图所示：

4. 在“更新Hpa配置”页面，根据实际需求进行设置，并单击**更新Hpa**。如下图所示：



通过编辑 YAML 更新

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

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

指标类型

相关指标和类型请参见 自动伸缩指标说明。

自动伸缩指标说明

最近更新时间：2022-05-16 11:11:35

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

自动伸缩指标

自动伸缩指标详情如下表所示：

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

CPU 指标

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

硬盘

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

网络

指标名称（控制台）	单位（控制台）	备注	type	metricName	默认单位
网络入带宽	Mbps	单 Pod 下所有容器的入方向带宽之和	Pods	k8s_pod_network_receive_bytes_bw	Bps
网络出带宽	Mbps	单 Pod 下所有容器的出方向带宽之和	Pods	k8s_pod_network_transmit_bytes_bw	Bps
网络入流量	KB/s	单 Pod 下所有容器的入方向流量之和	Pods	k8s_pod_network_receive_bytes	B/s
网络出流量	KB/s	单 Pod 下所有容器的出方向流量之和	Pods	k8s_pod_network_transmit_bytes	B/s

网络入包量	个/s	单 Pod 下所有容器的入方向包数之和	Pods	k8s_pod_network_receive_packets	个/s
网络出包量	个/s	单 Pod 下所有容器的出方向包数之和	Pods	k8s_pod_network_transmit_packets	个/s

内存

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

GPU

> 说明：
> 以下所有 GPU 相关的触发指标，当前仅支持在 EKS 集群中使用。

指标名称（控制台）	单位（控制台）	备注	type	metricName	默认单位
GPU 使用量	CUDA Core	Pod GPU 使用量	Pods	k8s_pod_gpu_used	CUDA Core
GPU 申请量	CUDA Core	Pod GPU 申请量	Pods	k8s_pod_gpu_request	CUDA Core
GPU 利用率（占 Request）	%	GPU 使用占 Request 的比例	Pods	k8s_pod_rate_gpu_used_request	%
GPU 利用率（占节点）	%	GPU 使用占 node 的比例	Pods	k8s_pod_rate_gpu_used_node	%
GPU memory 使用量	Mib	Pod GPU memory 使用量	Pods	k8s_pod_gpu_memory_used_bytes	B
GPU memory 申请量	Mib	Pod GPU memory 申请量	Pods	k8s_pod_gpu_memory_request_bytes	B
GPU memory 利用率（占 Request）	%	GPU memory 使用占 Request 的比例	Pods	k8s_pod_rate_gpu_memory_used_request	%

GPU memory 利用率（占节点）	%	GPU memory 使用占 node 的比例	Pods	k8s_pod_rate_gpu_memory_used_node	%

通过 YAML 创建和编辑 HPA

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

> ⚠ 注意：
> TKE 同样兼容原生的 Resource 类型。

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

配置
ConfigMap 管理

最近更新时间：2022-04-18 14:13:56

简介

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

ConfigMap 控制台操作指引

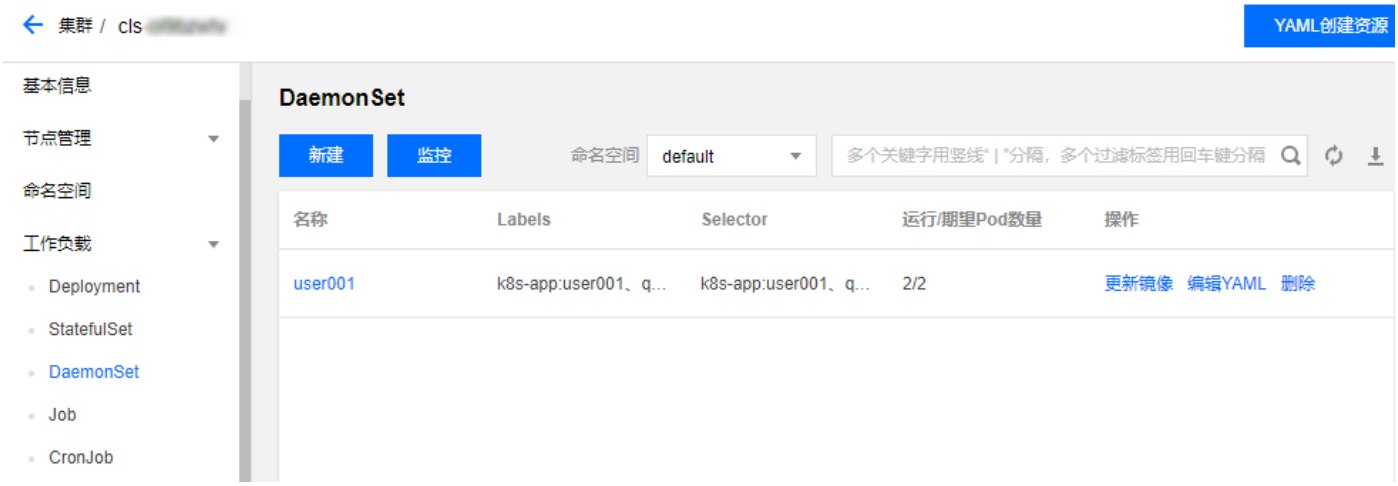
创建 ConfigMap

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**，进入集群列表页。
3. 单击需要创建 ConfigMap 的集群 ID，进入集群管理页面。
4. 选择 **配置管理** > **ConfigMap**，进入 ConfigMap 信息页面。
5. 单击**新建**，进入 “新建ConfigMap” 页面。
6. 根据实际需求，设置 ConfigMap 参数。关键参数信息如下：
   - 名称：自定义。
   - 命名空间：根据实际需求进行选择命名空间类型，定义变量名和变量值。
7. 单击**创建ConfigMap**，完成创建。

使用 ConfigMap

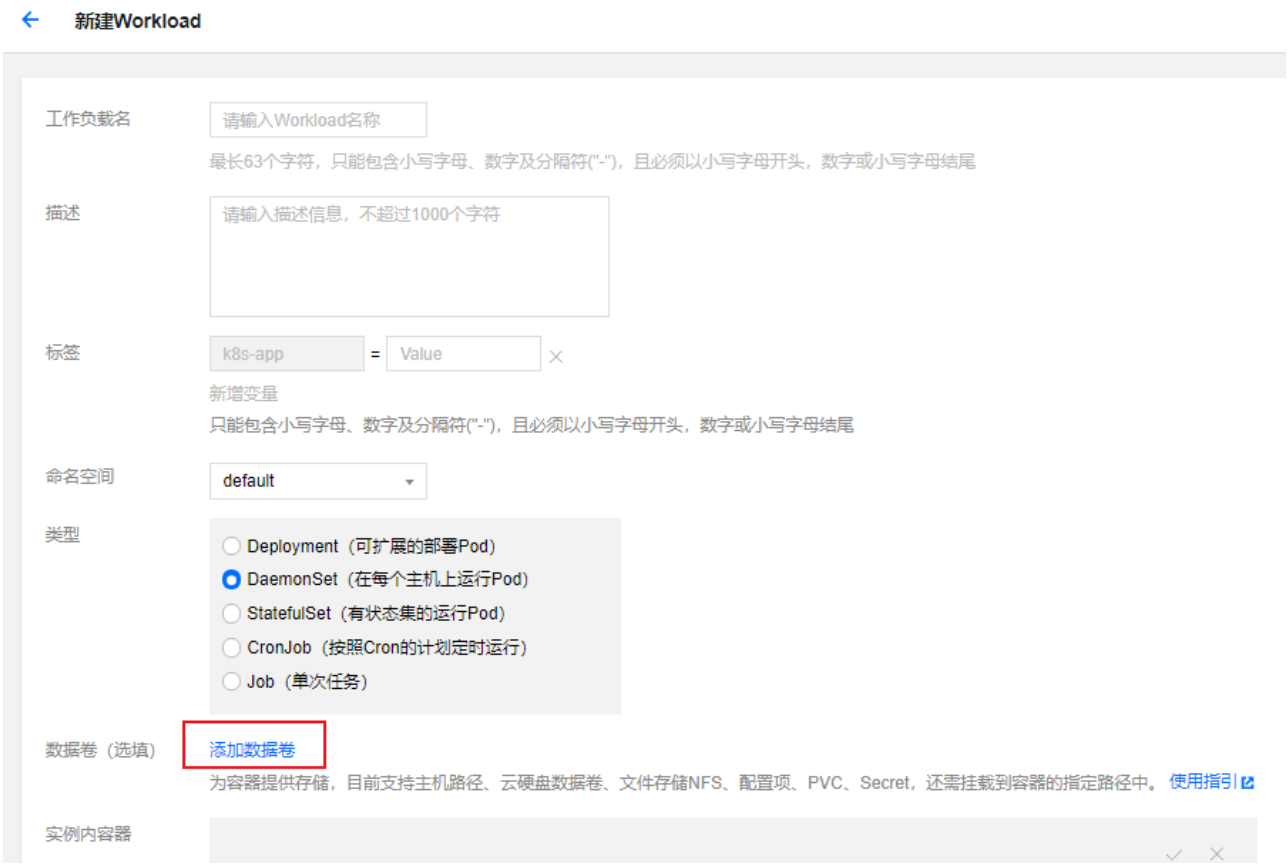
方式一：数据卷使用 ConfigMap 类型

1. 登录 容器服务控制台 。
2. 在左侧导航栏中单击**集群**，进入集群列表页。
3. 单击需要部署 Workload 的集群 ID，进入集群管理页面。
4. 在 “工作负载” 下，任意选择 Workload 类型，进入对应的信息页面。例如，选择**工作负载** > **DaemonSet**，进入 DaemonSet 信息页面。如下图所示：



5. 单击**新建**，进入 “新建Workload” 页面。

6. 根据页面信息，设置工作负载名、命名空间等信息。并在 “数据卷” 中，单击**添加数据卷**，添加数据卷。如下图所示：



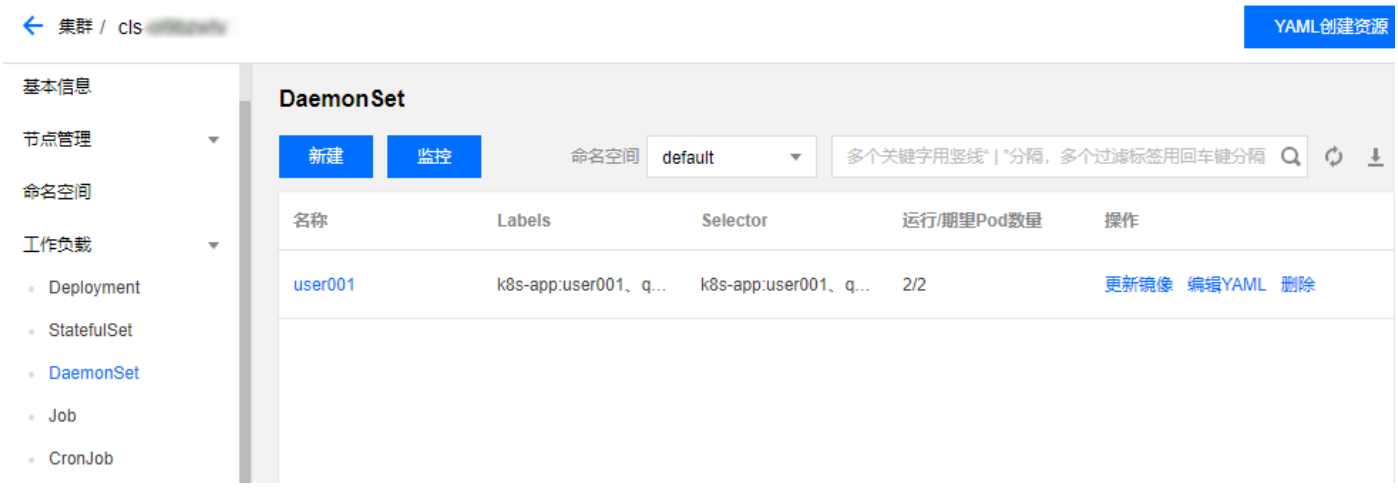
7. 选择 “使用ConfigMap” 方式，填写名称，单击**选择配置项**。如下图所示：

8. 在弹出的 “设置ConfigMap” 窗口中，参考以下信息配置挂载点，并单击**确认**。如下图所示：
   - **选择ConfigMap**：根据实际需求进行选择。
   - **选项**：提供“全部”和“指定部分Key”两种选择。
   - **Items**：当选择“指定部分Key”选项时，可以通过添加 item 向特定路径挂载，如挂载点是 /data/config，文件名是 filename，最终会该键值对的值会存储在 /data/config/filename 下。

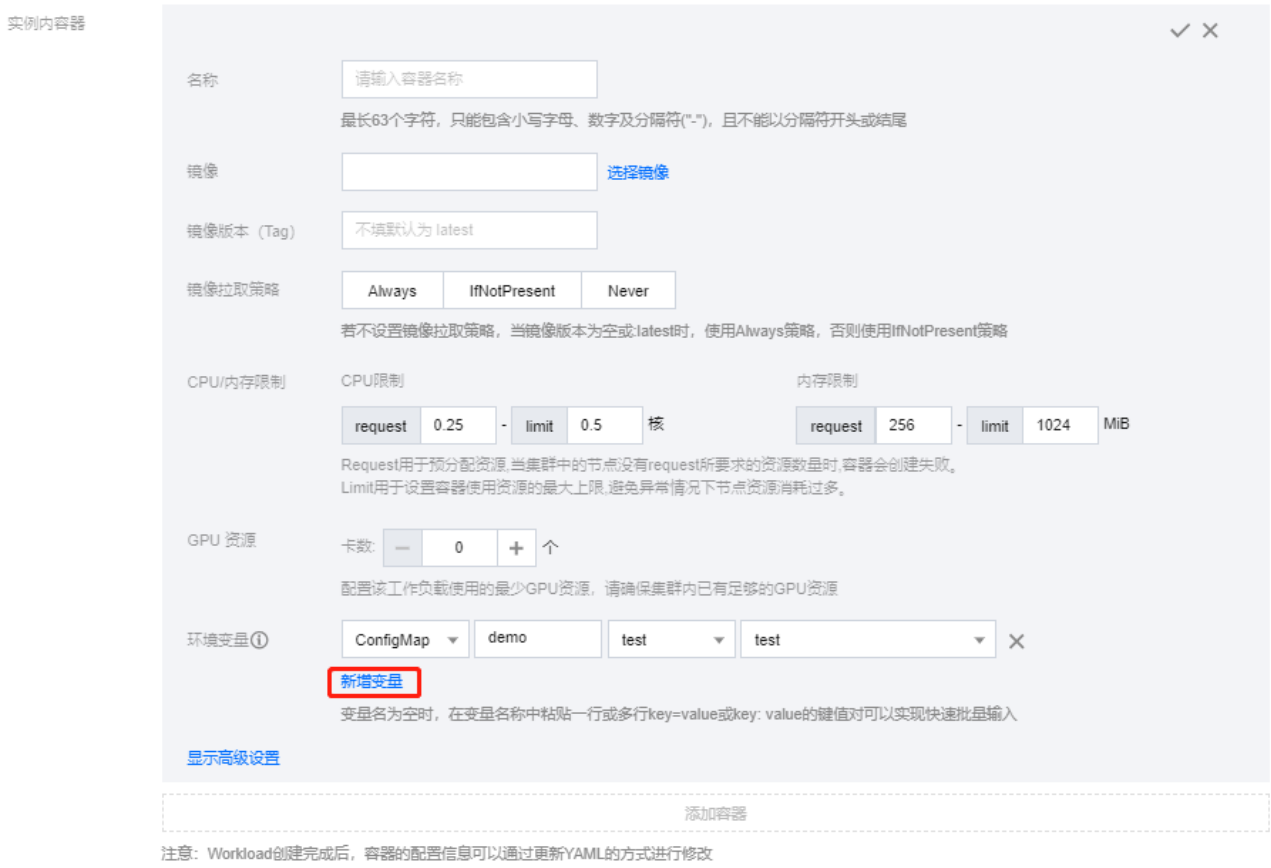
9. 单击**创建Workload**，完成创建。

方式二： 环境变量中使用 ConfigMap 类型

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**，进入集群列表页。
3. 单击需要部署 Workload 的集群 ID，进入待部署 Workload 的集群管理页面。
4. 在 “工作负载” 下，任意选择 Workload 类型，进入对应的信息页面。例如，选择**工作负载** > **DaemonSet**，进入 DaemonSet 信息页面。如下图所示：



5. 单击**新建**，进入 “新建Workload” 页面。
6. 根据页面信息，设置工作负载名、命名空间等信息。并在 “实例内容器” 的 “环境变量” 中，单击**新增变量**。如下图所示：

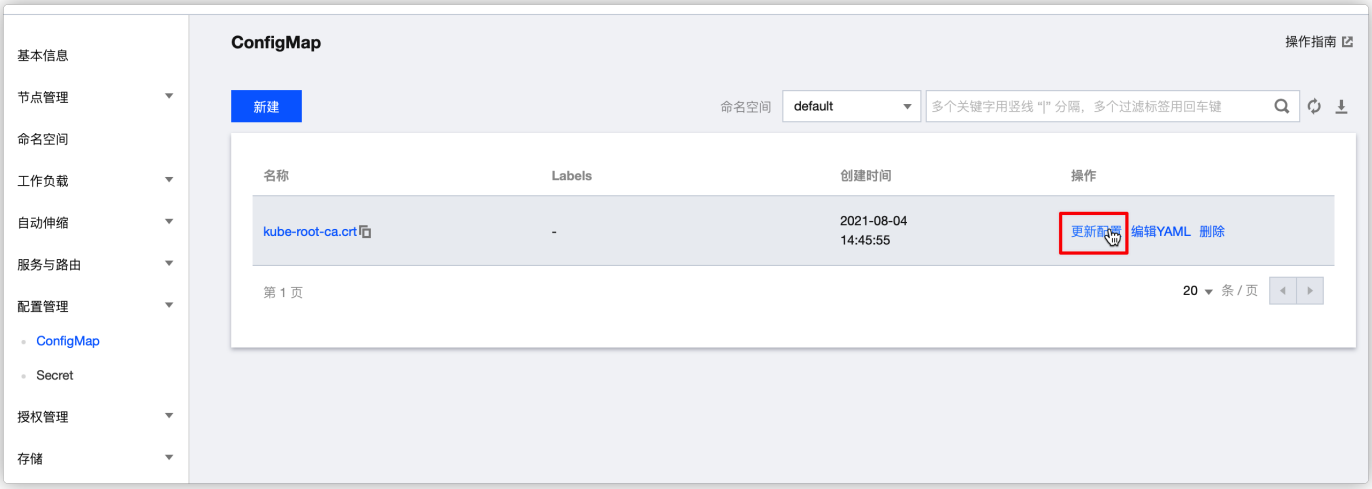


7. 选择 “ConfigMap” 环境变量方式，并根据实际需求选择资源。
8. 单击**创建Workload**，完成创建。

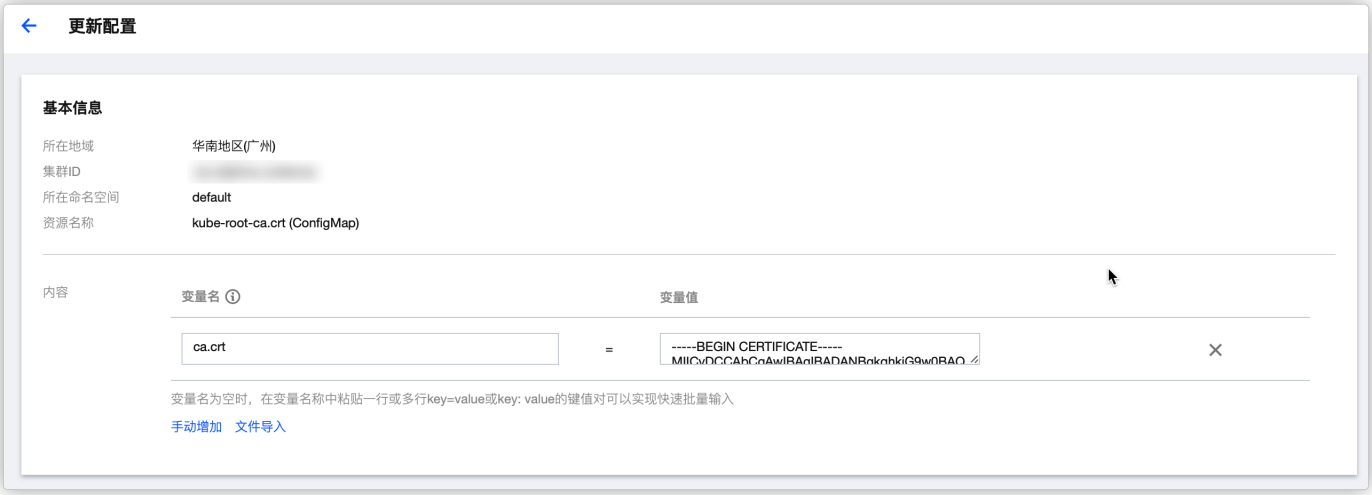
更新 ConfigMap

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**，进入集群列表页。
3. 单击需要更新 ConfigMap 的集群 ID，进入集群管理页面。
4. 选择 **配置管理** > **ConfigMap**，进入 ConfigMap 信息页面。

5. 在需要更新的 ConfigMap 行中，单击右侧的**更新配置**，进入更新 ConfigMap 页面。



6. 在 “更新配置” 页面，编辑 key-value 类型的键值对，单击**完成**。



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，可通过该参数设置腾讯云 TKE 的额外增强能力。

创建 ConfigMap

方式一：通过 YAML 示例文件方式创建

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

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

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

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

4. 执行以下命令，验证创建是否成功。

```
kubectl get configmap
```

返回类似以下信息，即表示创建成功。

```
NAME DATA AGE
test 2 39d
test-config 3 18d
```

方式二：通过执行命令方式创建

执行以下命令，在目录中创建 ConfigMap。

```
kubectl create configmap <map-name> <data-source>
```

- <map-name>：表示 ConfigMap 的名字。
- <data-source>：表示目录、文件或者字面值。

更多参数详情可参见 Kubernetes configMap 官方文档。

使用 ConfigMap

方式一：数据卷使用 ConfigMap 类型

YAML 示例如下：

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:latest
volumeMounts:
name: config-volume
mountPath: /etc/config
volumes:
name: config-volume
configMap:
name: test-config ## 设置 ConfigMap 来源
## items: ## 设置指定 ConfigMap 的 Key 挂载
```

```
## key: key1 ## 选择指定 Key
## path: keys ## 挂载到指定的子路径
restartPolicy: Never
```

方式二：环境变量中使用 ConfigMap 类型

YAML 示例如下：

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:latest
env:
- name: key1
valueFrom:
configMapKeyRef:
name: test-config ## 设置来源 ConfigMap 文件名
key: test-config.key1 ## 设置该环境变量的 Value 来源项
restartPolicy: Never
```

Secret 管理

最近更新时间：2022-06-09 11:01:34

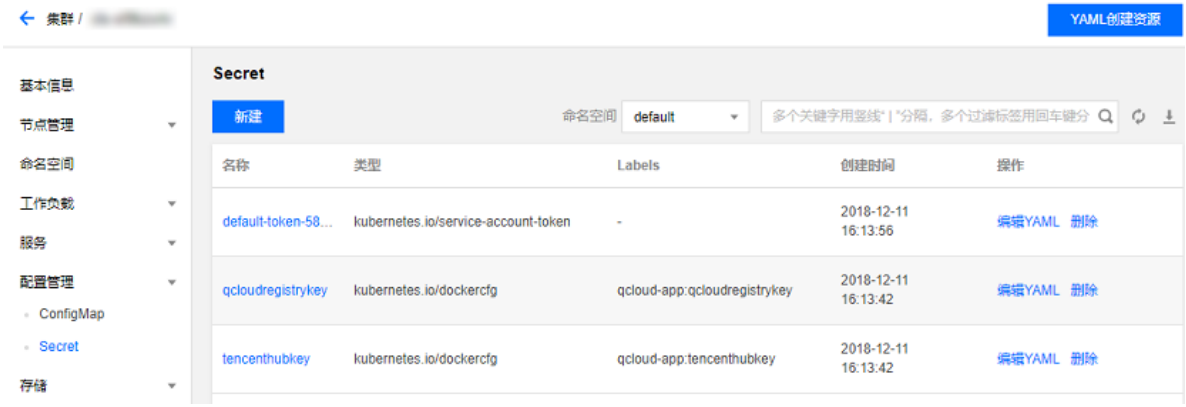
简介

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 key-value 类型的键值对，您可以通过控制台的 Kubectl 工具创建对应的 Secret 对象，也可以通过挂载数据卷、环境变量或在容器的运行命令中使用 Secret。

Secret 控制台操作指引

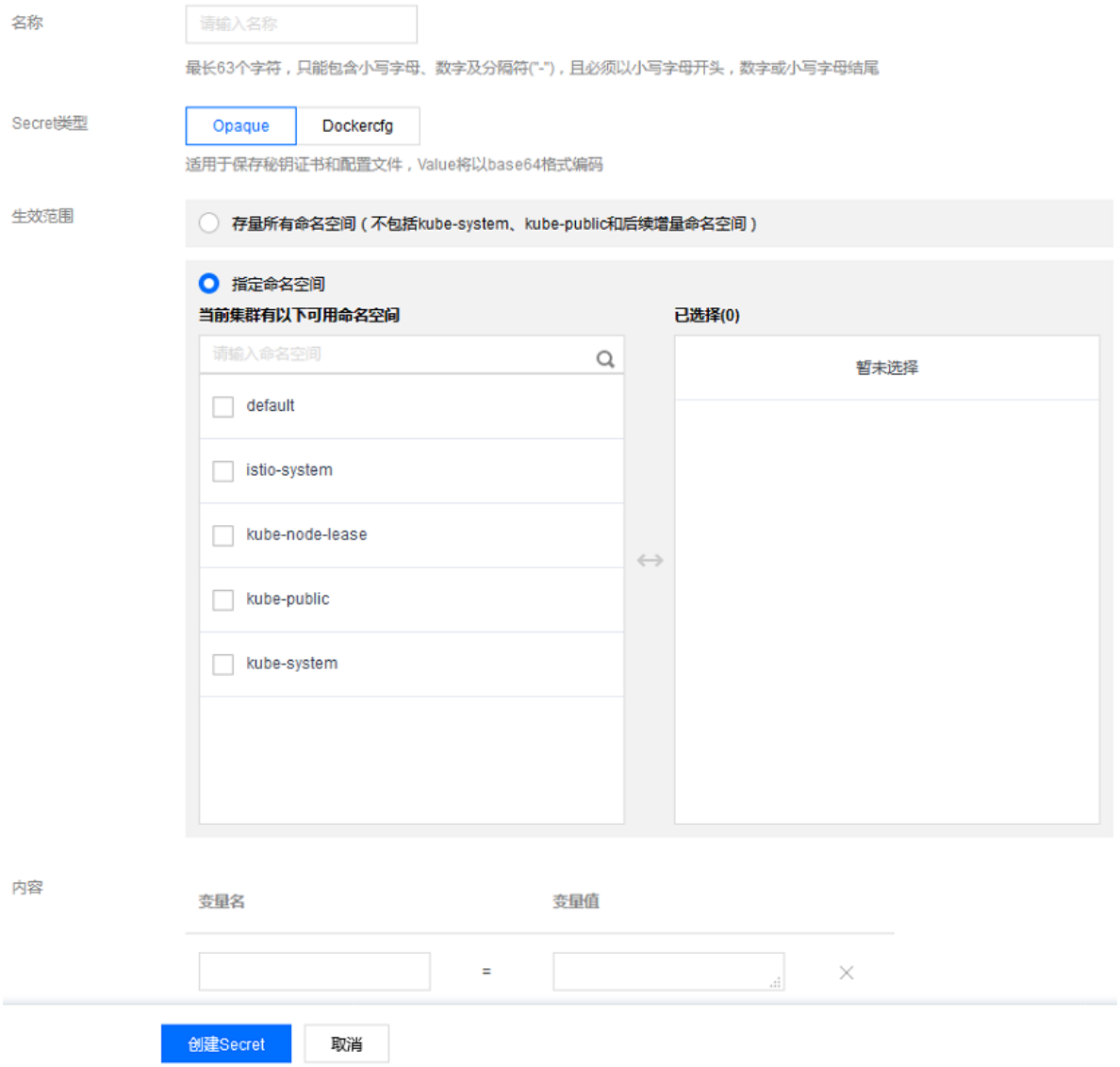
创建 Secret

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 选择需要创建 Secret 的集群 ID，进入待创建 Secret 的集群管理页面。
3. 选择左侧导航栏中的**配置管理** > **Secret**，进入 Secret 信息页面。如下图所示：



4. 单击**新建**，进入“新建Secret”页面。

5. 在“新建Secret”页面，根据实际需求，进行如下参数设置。如下图所示：



- **名称**：请输入自定义名称。
- **Secret类型**：提供Opaque和Dockercfg两种类型，请根据实际需求进行选择。
  - **Opaque**：适用于保存密钥证书和配置文件，Value 将以 base64 格式编码。
  - **Dockercfg**：适用于保存私有 Docker Registry 的认证信息。
- **生效范围**：提供以下两种范围，请根据实际需求进行选择。
  - **存量所有命名空间**：不包括 kube-system、kube-public 和后续增量命名空间。
  - **指定命名空间**：支持选择当前集群下一个或多个可用命名空间。
- **内容**：根据不同的 Secret 类型，进行配置。
  - 当 Secret 类型为**Opaque**时：根据实际需求，设置变量名和变量值。
  - 当 Secret 类型为**Dockercfg**时：
    - 仓库域名：请根据实际需求输入域名或 IP。
    - 用户名：请根据实际需求输入第三方仓库的用户名。
    - 密码：请根据实际需求设置第三方仓库的登录密码。

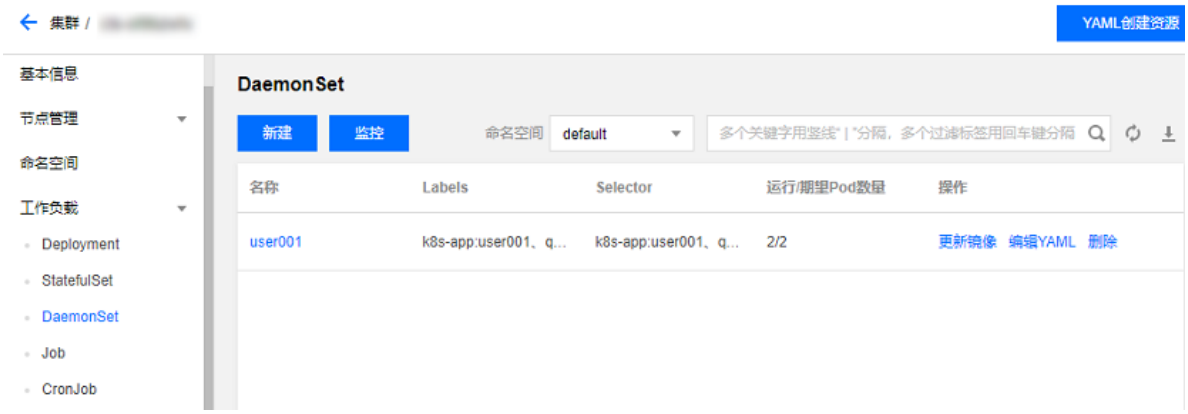
> 说明：
> 如果本次为首次登录系统，则会新建用户，相关信息写入 ~/.dockercrg 文件中。

6. 单击**创建 Secret**，即可完成创建。

使用 Secret

方式一：数据卷使用 Secret 类型

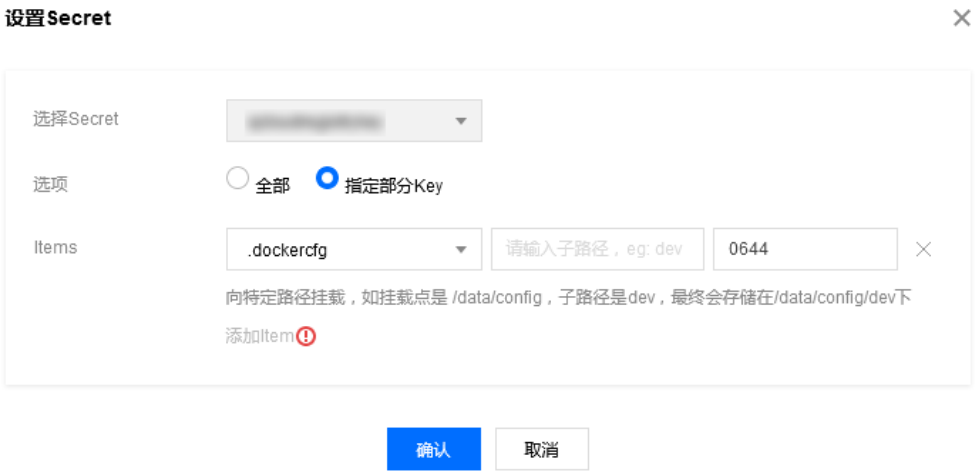
1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 选择需要部署 Workload 的集群 ID，进入待部署 Workload 的集群管理页面。
3. 在**工作负载**下，任意选择 Workload 类型，进入对应的信息页面。
   例如，选择**工作负载 >DaemonSet**，进入 DaemonSet 信息页面。如下图所示：



4. 单击**新建**，进入 “新建Workload” 页面。
5. 根据页面信息，设置工作负载名、命名空间等信息。并在 “数据卷” 中，单击**添加数据卷**。如下图所示：

6. 选择**使用Secret**方式，填写名称，并单击**选择Secret**。如下图所示：

7. 在弹出的“设置Secret”窗口中，配置挂载点，并单击**确认**。如下图所示：

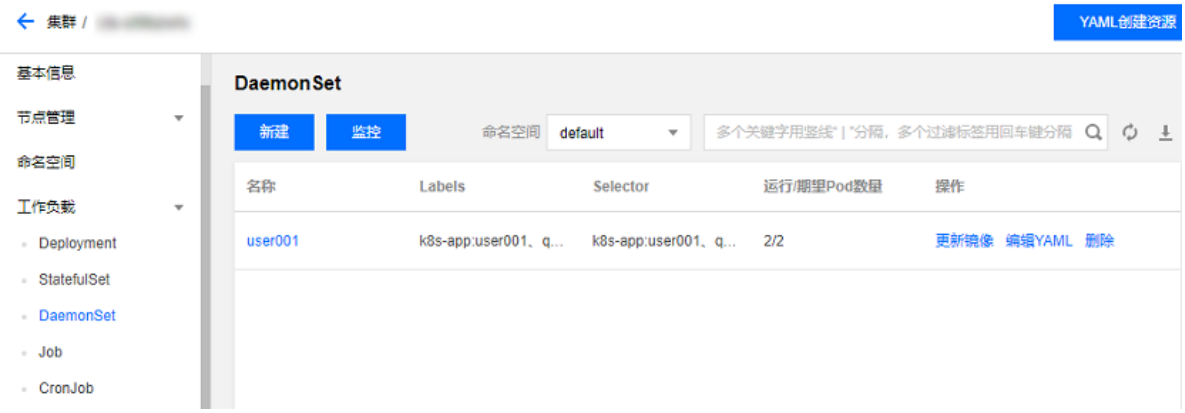


- **选择Secret**：根据实际需求进行。
- **选项**：提供**全部**和**指定部分 Key**两种选择。
- **Items**：当选择**指定部分 Key**选项时，可以通过添加 Item 向特定路径挂载，如挂载点是 /data/config，子路径是 dev，最终会存储在 /data/config/dev 下。

8. 单击**创建Workload**，完成创建。

方式二：环境变量中使用 Secret 类型

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 单击需要部署 Workload 的集群 ID，进入待部署 Workload 的集群管理页面。
3. 在**工作负载**下，任意选择 Workload 类型，进入对应的信息页面。
   例如，选择**工作负载 > DaemonSet**，进入 DaemonSet 信息页面。如下图所示：



4. 单击**新建**，进入 “新建Workload” 页面。

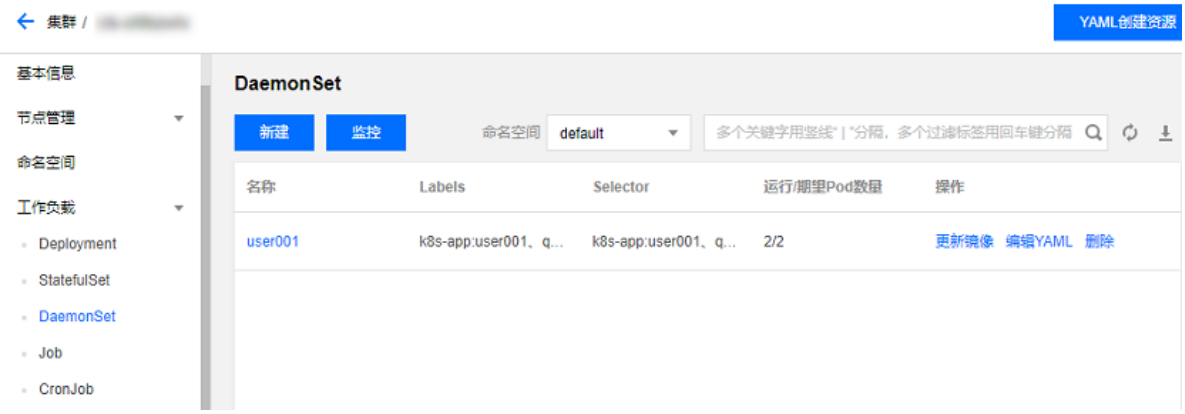
5. 根据页面信息，设置工作负载名、命名空间等信息。并在“实例内容器”的“环境变量”中，单击**引用ConfigMap/Secret**。如下图所示：

6. 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：

7. 单击**创建Workload**，完成创建。

方法三：使用第三方镜像仓库时引用

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 选择需要部署 Workload 的集群 ID，进入待部署 Workload 的集群管理页面。
3. 在**工作负载**下，任意选择 Workload 类型，进入对应的信息页面。
   例如，选择**工作负载** > **DaemonSet**，进入 DaemonSet 信息页面。如下图所示：

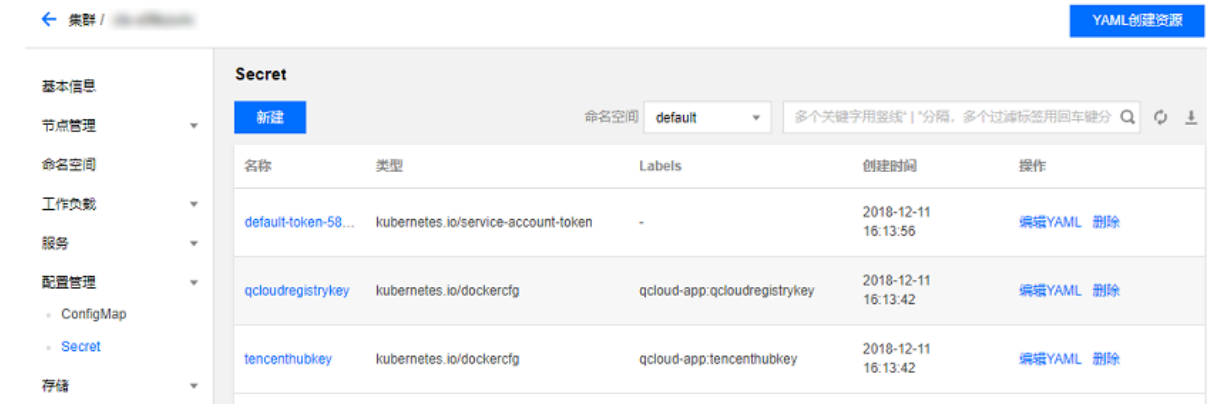


4. 单击**新建**，进入 “新建Workload” 页面。
5. 根据页面信息，设置工作负载名、命名空间等信息。单击本页面左下角**显示高级设置** 。
6. 单击**添加**，请根据实际情况选择dockercfg类型的Secret 。如下图所示：

7. 单击**创建Workload**，完成创建。

更新 Secret

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 选择需要更新 YAML 的集群 ID，进入待更新 YAML 的集群管理页面。
3. 选择**配置管理** > **Secret**，进入 Secret 信息页面。如下图所示：



4. 在需要更新 YAML 的 Secret 行中，单击**编辑YAML**，进入更新 Secret 页面。
5. 在“更新Secret”页面，编辑 YAML，并单击**完成**即可更新 YAML。

> 说明：
> 如需修改 key-values，则编辑 YAML 中 data 的参数值，并单击**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

1. 依次执行以下命令，获取 Pod 的用户名和密码。

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

2. 执行 Kubectl 命令，创建 Secret。

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

3. 执行以下命令，查看 Secret 详情。

```
kubectl describe secrets/ test-secret
```

方式二：YAML 文件手动创建

> 说明：
> 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
```

方法三：使用第三方镜像仓库时引用

YAML 示例如下：

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:latest
imagePullSecrets:
- name: test-secret ## 设置来源 Secret 文件名
restartPolicy: Never
```

Service 管理
概述

最近更新时间：2022-06-16 10:26:25

Service 基本概念

用户在 Kubernetes 中可以部署各种容器，其中一部分是通过 HTTP、HTTPS 协议对外提供七层网络服务，另一部分是通过 TCP、UDP 协议提供四层网络服务。而 Kubernetes 定义的 Service 资源就是用来管理集群中四层网络的服务访问。

Kubernetes 的 ServiceTypes 允许指定 Service 类型，默认为 ClusterIP 类型。ServiceTypes 的可取值以及行为描述如下：

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

ClusterIP 和 NodePort 类型的 Service，在不同云服务商或是自建集群中的行为表现通常情况下相同。而 LoadBalancer 类型的 Service，由于使用了云服务商的负载均衡进行服务暴露，云服务商会围绕其负载均衡的能力提供不同的额外功能。例如，控制负载均衡的网络类型，后端绑定的权重调节等，详情请参见 Service 管理 相关文档。

服务访问方式

根据上述 ServiceTypes 定义。您可以使用腾讯云容器服务（TKE）提供的以下四种服务访问方式：

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

负载均衡相关概念

Service 工作原理

腾讯云容器集群中的 Service Controller 组件负责用户 Service 资源的同步。当用户创建、修改或删除 Service 资源时、集群节点或 Service Endpoints 出现变化时、组件容器发生飘移重启时，组件都会对用户的 Service 资源进行同步。

Service Controller 会依照用户 Service 资源的描述创建对应的负载均衡资源，并对监听器及其后端进行配置。当用户删除集群 Service 资源时，也会回收对应负载均衡资源。

Service 生命周期管理

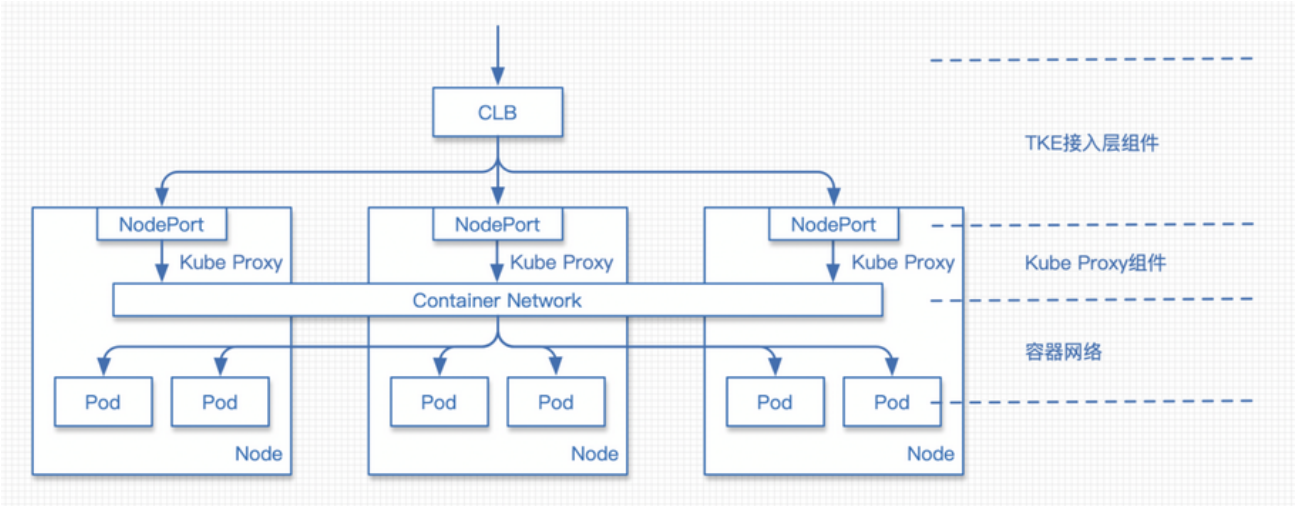
Service 对外服务的能力依赖于负载均衡所提供的资源，服务资源管理也是 Service 的重要工作之一。Service 在资源的生命周期管理中会使用以下标签：

- tke-createdBy-flag = yes：标识该资源是由容器服务创建。
  - 若有此标签，Service 会在销毁时删除对应资源。
  - 若无此标签，Service 会在销毁时，仅删除负载均衡内的监听器资源，而不删除负载均衡自身。
- tke-clusterId = <ClusterId>：标识该资源被哪一个 Cluster 所使用的。
  - 若 ClusterId 正确，则Service 会在销毁时，删除对应标签。

> 说明：
>
> - 若用户使用了已有负载均衡，则 Service 仅会使用该负载均衡，而不会删除该负载均衡。
> - 若用户在负载均衡上面开启了 删除保护，或者使用 私有连接，则删除 Service 时，不会删除该负载均衡。

当 LoadBalancer 类型的 Service 集群资源被创建时，对应负载均衡的生命周期就开始了。直到 Service 资源被删除或是负载均衡被重建时，负载均衡的生命周期就结束了。在此期间负载均衡会持续根据 Service 资源的描述进行同步。**当用户切换 Service 的网络访问时，例如公网 > VPC 内网、VPC 内网 > 公网、VPC 子网切换、更换使用的已有负载均衡，此类操作都会涉及到负载均衡的重建或销毁。**

LoadBalancer 类型 Service 工作原理如下图所示：



Service 高危操作

- 使用传统型负载均衡（已不推荐使用）。
- 修改或者删除由容器服务添加的负载均衡标签，再购买新的负载均衡并恢复其标签。
- 通过负载均衡控制台，修改由容器服务所管理负载均衡的监听器名称。

Service 功能

Service 相关操作及功能如下，您可参考以下文档进一步了解：

- Service 基本功能
- Service 负载均衡配置
- Service 使用已有 CLB
- Service 后端选择

参考资料

您也可以参考开源文档 Kubernetes Service，了解关于 Service 的更多信息。

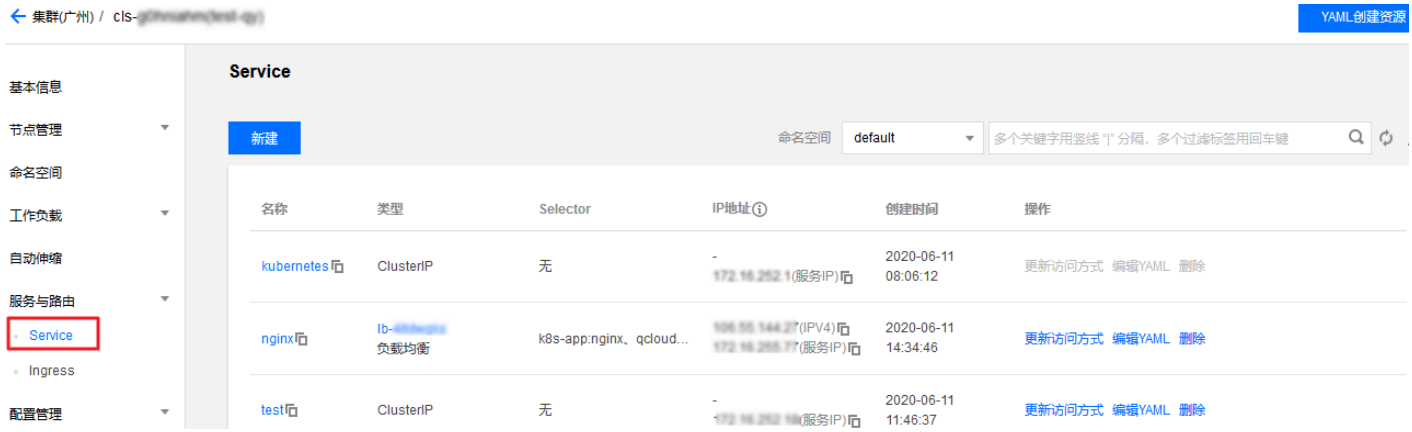
Service 基本功能

最近更新时间：2022-04-18 14:12:18

Service 控制台操作指引

创建 Service

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面单击需要创建 Service 的集群 ID，进入待创建 Service 的集群管理页面。
3. 选择**服务与路由** > **Service**，进入 “Service” 管理页面。如下图所示：



4. 单击**新建**，进入 “新建Service” 页面。
   根据实际需求，设置 Service 参数。关键参数信息如下：
   - **服务名称**：自定义。
   - **命名空间**：根据实际需求进行选择。
   - **访问设置**：请参考 概述 并根据实际需求进行设置。

   > 说明：
   >
   > - 如需使用已有负载均衡器，请参考 使用已有 CLB。
   > - 由于4层 CLB 仅限制 **CLB VIP + 监听器协议 + 后端 RS VIP + 后端 RS 端口4元组唯一**，且未包含 CLB 监控端口。因此不支持 CLB 监听端口不同，协议及 RS 相同的场景。容器服务也不支持同一个业务对外开放相同协议的不同端口。

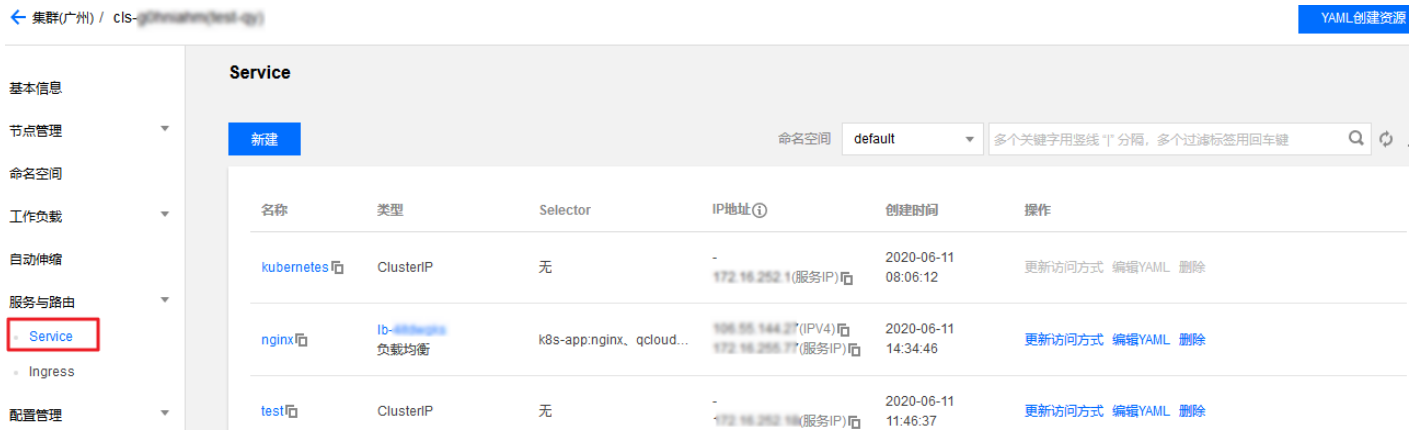
5. 单击**创建服务**，完成创建。

更新 Service

更新 YAML

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面中，选择需要更新 YAML 的集群 ID，进入待更新 YAML 的集群管理页面。

3. 选择**服务与路由** > **Service**，进入 Service 信息页面。如下图所示：



4. 单击需更新 YAML 的 Service 所在行右侧的**编辑YAML**，进入更新 Service 页面。
5. 在 “更新Service” 页面，编辑 YAML 后单击**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 ##若是创建内网访问的 Service 需指定该条 annotation
name: my-service
spec:
selector:
app: MyApp
ports:
- protocol: TCP
port: 80
targetPort: 9376
type: LoadBalancer
```

- **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，可通过该参数设置腾讯云容器服务的额外增强能力。
- **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**：使用腾讯云提供的负载均衡器公开服务，默认创建公网负载均衡，指定 annotations 可创建内网负载均衡。
  - **ExternalName**：将服务映射到 DNS，仅适用于 kube-dns1.7及更高版本。

创建 Service

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

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

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

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

4. 执行以下命令，验证创建是否成功。

```
kubectl get services
```

返回类似以下信息，即表示创建成功。

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

更新 Service

方法1

执行以下命令，更新 Service。

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

方法2

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

```
kubectl create/apply
```

删除 Service

执行以下命令，删除 Service。

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

Service 负载均衡配置

最近更新时间：2022-03-11 11:39:18

TkeServiceConfig

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

使用场景

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

配置说明

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

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

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

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

Service 与 TkeServiceConfig 关联行为

1. 创建 Loadbalancer 模式 Service 时，设置注解 **service.cloud.tencent.com/tke-service-config-auto: "true"**，将自动创建 <ServiceName>-auto-service-config。您也可以通过 **service.cloud.tencent.com/tke-service-config:<config-name>** 直接指定您自行创建的 TkeServiceConfig。两个注解不可同时使用。
2. 其中自动创建的 TkeServiceConfig 存在以下同步行为：
   - 更新 Service 资源时，新增若干四层监听器时，如果该监听器或转发规则没有对应的 TkeServiceConfig 配置片段。 Service-Controller 将主动添加 TkeServiceConfig 对应片段。
   - 删除若干四层监听器时，Service-controller 组件将主动删除 TkeServiceConfig 对应片段。
   - 删除 Service 资源时，联级删除该 TkeServiceConfig。
   - 用户修改 Service 默认的 TkeServiceConfig，TkeServiceConfig 内容同样会被应用到负载均衡。
3. 您也可以参考下列 TkeServiceConfig 完整配置参考自行创建需要的 CLB 配置，Service 通过注解：**service.cloud.tencent.com/tke-service-config:<config-name>** 引用该配置。
4. 其中您手动创建的 TkeServiceConfig 存在以下同步行为：
   - 当用户在 Service 中添加配置注解时，负载均衡将会立即进行设置同步。
   - 当用户在 Service 中删除配置注解时，负载均衡将会保持不变。
   - 修改 TkeServiceConfig 配置时，引用该配置 Service 的负载均衡将会根据新的 TkeServiceConfig 进行设置同步。
   - Service 的监听器未找到对应配置时，该监听器将不会进行修改。
   - Service 的监听器找到对应配置时，若配置中没有声明的属性，该监听器将不会进行修改。

完整配置参考

```
apiVersion: cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
name: sample # 配置的名称
namespace: default # 配置的命名空间
spec:
```

```
loadBalancer:
l4Listeners: # 四层规则配置，适用于Service的监听器配置。
- protocol: TCP # 协议端口锚定Service的四层规则。必填，枚举值：TCP|UDP。
port: 80 # 必填，可选值：1~65535。
session: # 会话保持相关配置。选填
enable: true # 是否开启会话保持。必填，布尔值
sessionExpireTime: 100 # 会话保持的时间。选填，默认值：30，可选值：30~3600，单位：秒。
healthCheck: # 健康检查相关配置。选填
enable: true # 是否开启健康检查。必填，布尔值
intervalTime: 10 # 健康检查探测间隔时间。选填，默认值：5，可选值：5~300，单位：秒。
healthNum: 2 # 健康阈值，表示当连续探测几次健康则表示该转发正常。选填，默认值：3，可选值：2~10，单位：次。
unHealthNum: 3 # 不健康阈值，表示当连续探测几次健康则表示该转发异常。选填，默认值：3，可选值：2~10，单位：次。
timeout: 10 # 健康检查的响应超时时间，响应超时时间要小于检查间隔时间。选填，默认值：2，可选值：2~60，单位：秒。
scheduler: WRR # 请求转发方式配置。WRR、LEAST_CONN、IP_HASH分别表示按权重轮询、最小连接数、按IP哈希。选填，枚举值：WRR|LEAST_CONN。
internetMaxBandwidthOut: 100 # 最大出带宽，仅对公网属性的LB生效。选填，可选值：0~2048，单位Mbps。
```

示例

Deployment 示例：jetty-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: jetty
name: jetty-deployment
namespace: default
spec:
progressDeadlineSeconds: 600
replicas: 3
revisionHistoryLimit: 10
selector:
matchLabels:
app: jetty
strategy:
rollingUpdate:
maxSurge: 25%
maxUnavailable: 25%
type: RollingUpdate
template:
metadata:
creationTimestamp: null
labels:
app: jetty
spec:
containers:
- image: jetty:9.4.27-jre11
imagePullPolicy: IfNotPresent
name: jetty
ports:
- containerPort: 80
protocol: TCP
```

```
- containerPort: 443
protocol: TCP
resources: {}
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
dnsPolicy: ClusterFirst
restartPolicy: Always
schedulerName: default-scheduler
securityContext: {}
terminationGracePeriodSeconds: 30
```

Service 示例：jetty-service.yaml

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

该示例中包含以下配置：

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

TkeServiceConfig 示例：jetty-service-config.yaml

```
apiVersion: cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
name: jetty-service-config
namespace: default
spec:
loadBalancer:
l4Listeners:
- protocol: TCP
```

```
port: 80
healthCheck:
enable: false
- protocol: TCP
port: 443
session:
enable: true
sessionExpireTime: 3600
healthCheck:
enable: true
intervalTime: 10
healthNum: 2
unHealthNum: 2
timeout: 5
scheduler: WRR
```

该示例中包含以下配置：
名称为 jetty-service-config**。且在四层监听器配置中，声明了以下两段配置：**

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

kubectl 配置命令

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

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

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

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

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

Service 使用已有 CLB

最近更新时间：2022-01-19 14:37:43

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

使用已有负载均衡的同步行为

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

使用已有负载均衡同步腾讯云标签行为

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

注意事项

- 指定使用的负载均衡需和集群处于同一 VPC。
- 请确保您的容器业务不和云服务器 CVM 业务共用一个负载均衡。
- 不支持您在负载均衡控制台操作 TKE 所管理负载均衡的监听器和后端绑定的服务器，您的更改会被 TKE 的自动同步所覆盖。
- 使用已有的负载均衡时：
  - Service Controller 将不负责该已有负载均衡的释放与回收。
  - 仅支持使用通过负载均衡控制台创建的负载均衡器，不支持复用由 TKE 自动创建的负载均衡，会破坏其他 Service 负载均衡的生命周期管理。
- **复用**负载均衡时：
  - 不支持跨集群复用负载均衡。
  - 您需要使用**复用**功能时，建议有明确的监听器端口管理，否则负载均衡在多个 Service 的使用下，会出现管理混乱。
  - 复用负载均衡的端口冲突时，将会被拒绝。如果在修改中出现冲突，那么出现冲突的监听器后端同步无法确保正确。
  - 复用负载均衡的 Service 不支持开启 Local 访问（传统型负载均衡限制）。
  - 删除 Service，则复用负载均衡绑定的后端云服务器需要自行解绑，同时会保留一个 tag tke-clusterId: cls-xxxx，需自行清理。

Service 示例

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

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

说明：

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

使用场景示例

使用包年包月的负载均衡对外提供服务

Service Controller 组件管理负载均衡生命周期时，仅支持购买按量计费的负载均衡资源。当用户需要长时间使用负载均衡时，包年包月计费模式在价格上有一定的优势。在此类场景下，用户就可以独立购买和管理负载均衡，再通过注解控制 Service 使用已有负载均衡，并将负载均衡的生命周期管理从 Service Controller 组件中剥离。

在同一端口暴露 TCP 和 UDP 服务

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

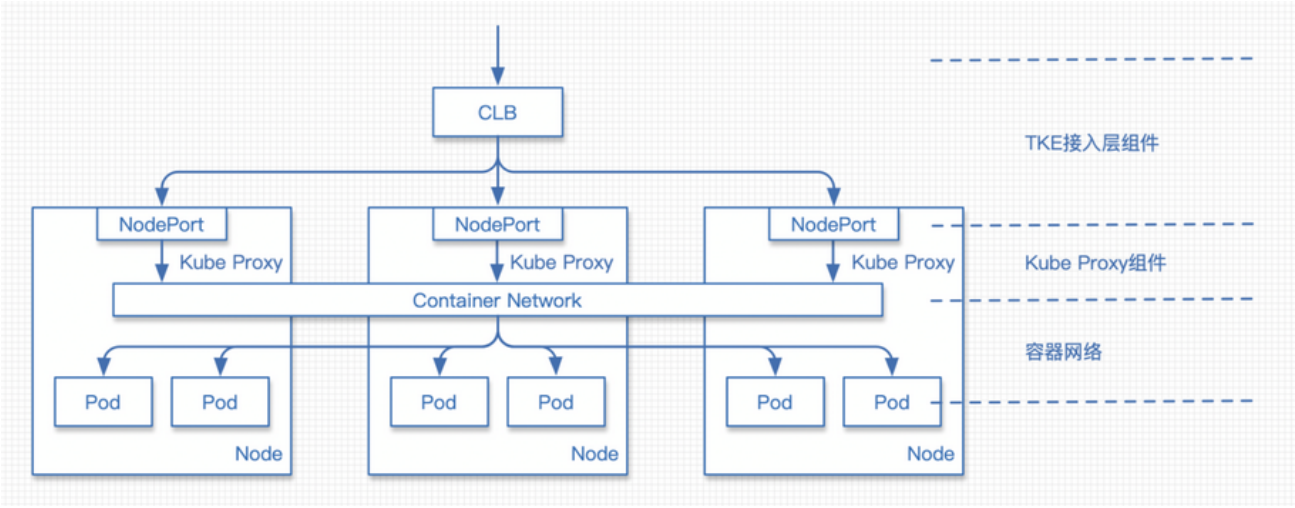
```
apiVersion: v1
kind: Service
metadata:
annotations:
service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
name: game-service-a
spec:
ports:
- name: 80-80-tcp
port: 80
protocol: TCP
targetPort: 80
selector:
app: game
type: LoadBalancer
---
apiVersion: v1
kind: Service
metadata:
annotations:
service.kubernetes.io/tke-existed-lbid: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

Service 后端选择

最近更新时间：2022-05-16 11:08:23

默认后端选择

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



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

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

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

指定接入层后端

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

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

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

注意事项

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

使用场景

大规模集群下的测试应用

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

示例

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

该示例包含以下配置：

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

Service Local 模式

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

- 优点：
  - 避免了 NAT 与节点间转发带来的性能损失。
  - 为服务端保留了请求来源 IP。
- 缺点：
  - 没有工作负载的节点，NodePort 将无法提供服务。

注意事项

- 负载均衡的同步是需要时间的。当 Local 类型的服务工作负载数量很少时，工作负载的飘移或滚动更新会很快。此时后端如未来得及同步，后端的服务可能会出现不可用的情况。
- 仅适用于处理低流量、低负载的业务，不建议在生产环境中使用。

示例：Service 开启 Local 转发（externalTrafficPolicy: Local）

```
apiVersion: v1
kind: Service
metadata:
name: nginx-service
spec:
externalTrafficPolicy: Local
ports:
- name: 80-80-no
```

```
port: 80
protocol: TCP
targetPort: 80
selector:
app: nginx
type: LoadBalancer
```

Local 默认后端选择

默认情况下，当 Service 开启 Local 模式之后，仍会按默认方式挂载几乎所有节点的 NodePort 作为后端。负载均衡会根据健康检查的结果，避免流量进入没有工作负载的后端节点。为了避免这些没有工作负载的后端被绑定，用户可以通过 service.kubernetes.io/local-svc-only-bind-node-with-pod: "true" 注解，在 Local 模式下指定绑定有工作负载节点作为后端。更多信息请参考 Kubernetes Service Local。

示例：Service 开启 Local 转发并开启 Local 绑定

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

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

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

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

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

Service 跨域绑定

最近更新时间：2022-05-19 16:41:15

简介

使用公网 CLB 型 Service 时，默认是在当前集群所在 VPC 内的随机可用区生成 CLB，现目前 TKE 的公网 CLB Service 已支持指定可用区、包括其他地域的可用区。本文将为您介绍如何通过控制台和 YAML 两种方式为 CLB Service 跨域绑定和指定可用区。

应用场景

- 需要支持 CLB 的跨地域接入或跨 VPC 接入，即 CLB 所在的 VPC 和当前集群所在的 VPC 不在同一 VPC 内。
- 需要指定 CLB 的可用区以实现资源的统一管理。

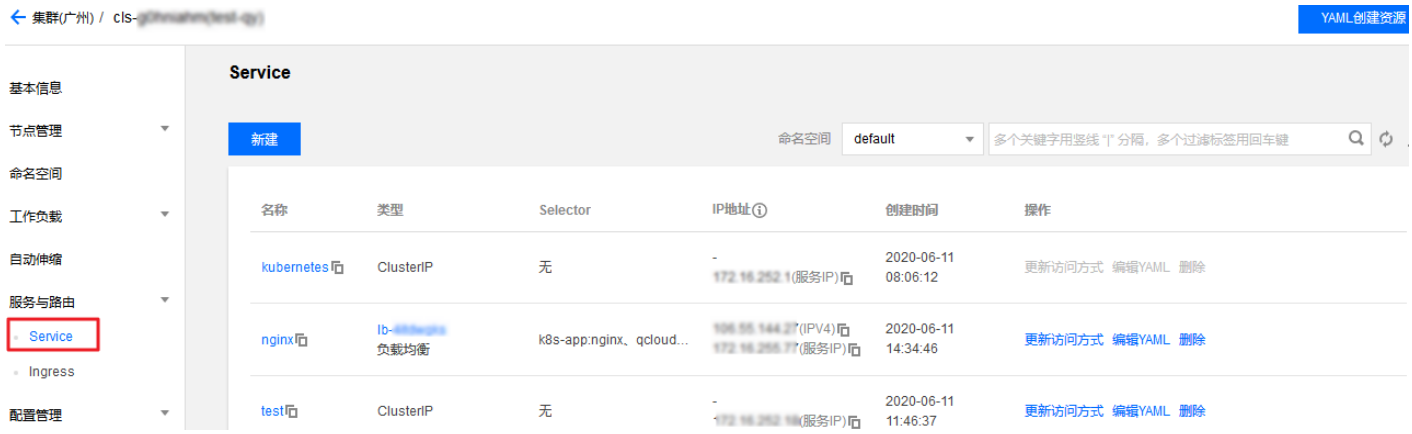
> 说明：
>
> 1. 跨域绑定仅支持“带宽上移账户”。若您无法确定账户类型，请参见 判断账户类型。
> 2. 如需使用非本集群所在 VPC 的 CLB，需先通过 云联网 打通当前集群 VPC 和 CLB 所在的 VPC。
> 3. 在确保 VPC 已经打通之后，请 在线咨询 申请使用该功能。
> 4. 以下 YAML 中，需要您输入地域 ID ，您可以通过 地域和可用区 查看地域 ID。

操作步骤

公网 CLB Service 跨域绑定和指定可用区支持通过控制台和 YAML 两种方式进行操作，操作步骤如下：

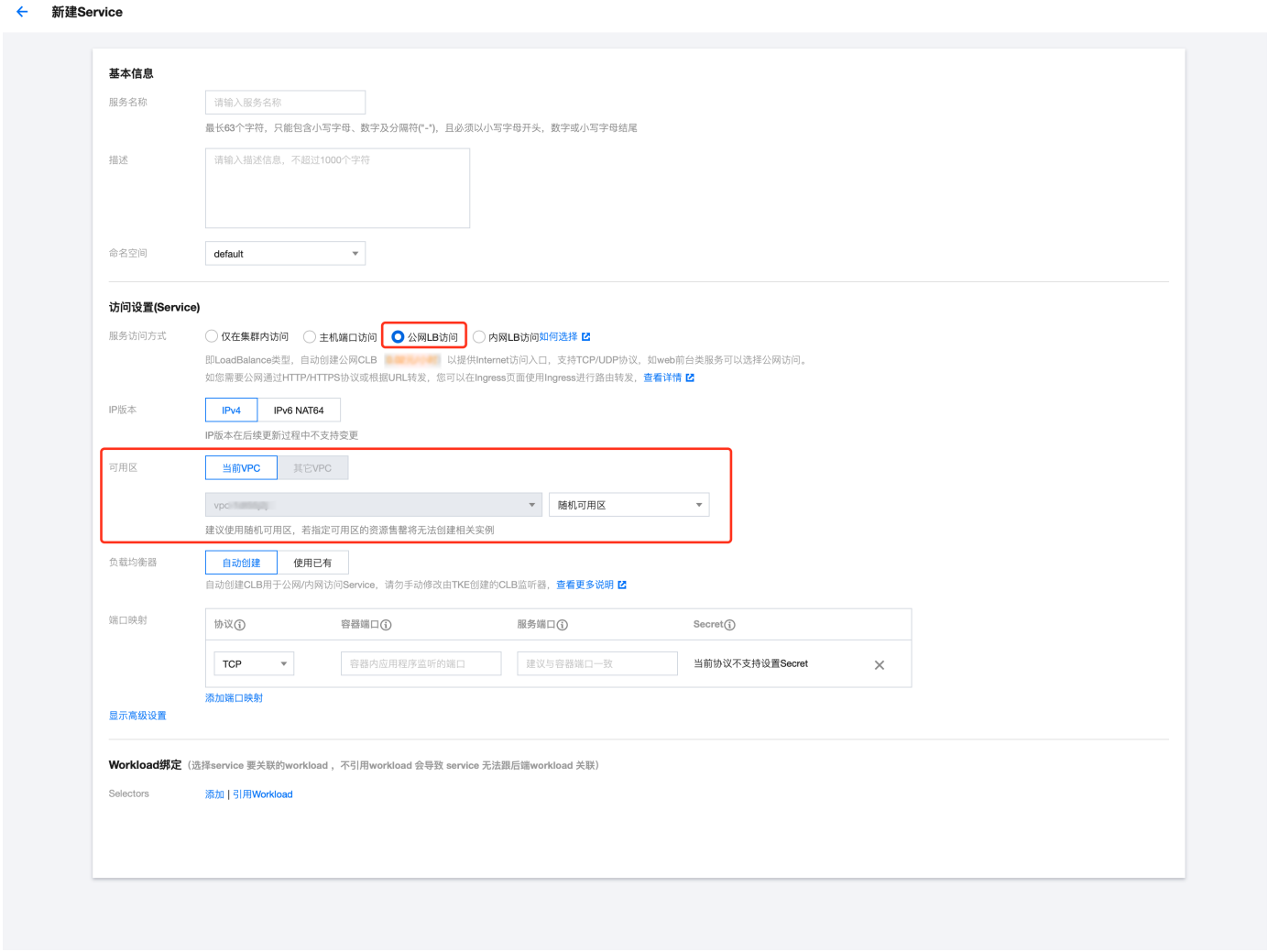
控制台方式

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面单击需要创建 Service 的集群 ID，进入待创建 Service 的集群管理页面。
3. 选择**服务与路由** > **Service**，进入 “Service” 管理页面。如下图所示：



4. 单击**新建**，进入“新建Service”页面。
5. 在“新建 Service”页面中配置相关可用区规则。配置规则说明如下：
   - **服务访问方式**：选择“公网LB访问”。
   - **当前VPC**：使用本集群所在 VPC 内的 CLB，建议使用随机可用区，若指定可用区的资源售罄将无法创建相关实例。

- **其它VPC**：仅支持通过 云联网 与当前集群的 VPC 打通的其他 VPC。建议使用随机可用区，若指定可用区的资源售罄将无法创建相关实例。



YAML 方式

> 1. 如需使用非本集群所在 VPC 的 CLB，需先通过 云联网 打通当前集群 VPC 和 CLB 所在的 VPC。
> 2. 在确保 VPC 已经打通之后，请 在线咨询 申请使用该功能。

示例1

如果仅需指定本集群所在 VPC 的可用区，例如集群的 VPC 在广州地域，CLB Service 需要指定广州一区的 CLB，可以在 Service 的 YAML 中添加如下 annotation：

```
service.kubernetes.io/service.extensiveParameters: '{"ZoneId":"ap-guangzhou-1"}'
```

示例2

如需使用非本集群所在 VPC 内的 CLB，可以在 Service 的 YAML 中添加如下 annotation：

```
service.cloud.tencent.com/cross-region-id: "ap-guangzhou"
service.cloud.tencent.com/cross-vpc-id: "vpc-646vhcjj"
```

> ⚠ 如果您还需要指定可用区，需要再添加示例1中的 annotation。

示例3

选择已有负载均衡进行异地接入，示例如下：

```
service.cloud.tencent.com/cross-region-id: "ap-guangzhou"
service.kubernetes.io/tke-existed-lbid: "lb-342wppll"
```

示例4

annotation 在 Service YAML 中的写法如下所示：

```
# 创建异地接入的负载均衡
apiVersion: v1
kind: Service
metadata:
annotations:
service.cloud.tencent.com/cross-region-id: "ap-chongqing"
service.cloud.tencent.com/cross-vpc-id: "vpc-mjekzyps"
name: echo-server-service
namespace: default
spec:
......
---
# 用户复用其他地域负载均衡的场景
apiVersion: v1
kind: Service
metadata:
annotations:
service.cloud.tencent.com/cross-region-id: "ap-chongqing"
service.kubernetes.io/tke-existed-lbid: "lb-o8ugf2wb"
name: echo-server-service
namespace: default
spec:
......
```

完整 Service Annotation 说明请参见 Service Annotation 说明 文档。

Service 优雅停机

最近更新时间：2022-03-24 11:47:33

简介

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

应用场景

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

> 注意：
> 仅针对 直连场景 生效，请检查您的集群是否支持直连模式。

操作步骤

步骤1：使用 Annotation 标明使用优雅停机

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

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

步骤2：使用 preStop 和 terminationGracePeriodSeconds

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

容器终止流程

以下为容器在 Kubernetes 环境中的终止流程：

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

具体操作步骤

1. **使用 preStop**
   要实现优雅终止，务必在业务代码里处理 SIGTERM 信号。主要逻辑是不接受新的流量进入，继续处理存量流量，所有连接全部断开才退出，了解更多可参见示例。

若您的业务代码中未处理 SIGTERM 信号，或者您无法控制使用的第三方库或系统来增加优雅终止的逻辑，也可以尝试为 Pod 配置 preStop，在其实现优雅终止的逻辑，示例如下：

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

更多关于 preStop 的配置请参见 Kubernetes API 文档。

在某些极端情况下，Pod 被删除的一小段时间内，仍然可能有新连接被转发过来，因为 kubelet 与 kube-proxy 同时 watch 到 Pod 被删除，kubelet 有可能在 kube-proxy 同步完规则前就已经停止容器，这时可能导致一些新的连接被转发到正在删除的 Pod，而通常情况下，当应用受到 SIGTERM 后都不再接受新连接，只保持存量连接继续处理，因此可能导致 Pod 删除的瞬间部分请求失败。

针对上述情况，可以利用 preStop 先 sleep 短暂时间，等待 kube-proxy 完成规则同步再开始停止容器内进程。示例如下：

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

2. **使用 terminationGracePeriodSeconds 调整优雅时长**

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

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

```
preStop:
exec:
command:
- sleep
- 5s
...
```

相关文档

- 故障处理：Nginx Ingress Controller 后端解绑不优雅的问题

使用 LoadBalancer 直连 Pod 模式 Service

最近更新时间：2022-05-26 16:01:10

操作场景

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

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

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

容器网络模式为 VPC-CNI

使用限制

- 集群 Kubernetes 版本需要高于 1.12。
- 集群网络模式必须开启 VPC-CNI 弹性网卡模式。
- 直连模式 Service 使用的工作负载需使用 VPC-CNI 弹性网卡模式。
- 默认 CLB 的后端数量限制是200个，如果您绑定的工作负载的副本数超过200时，可通过 在线咨询 提升负载均衡 CLB 的配额。
- 满足 CLB 本身绑定弹性网卡的功能限制，详情请参见 绑定弹性网卡。
- 开启直连 Pod 模式的工作负载更新时，将会根据 CLB 的健康检查状态进行滚动更新，会对更新速度造成一定影响。
- 不支持 HostNetwork 类型的工作负载。

操作步骤

控制台操作指引

1. 登录 容器服务控制台 。
2. 参考 控制台创建 Service 步骤，进入 “新建Service” 页面，根据实际需求设置 Service 参数。
   其中，部分关键参数信息需进行如下设置，如下图所示：

- **服务访问方式：选择为公网LB访问或内网LB访问。**
- **网络模式：勾选采用负载均衡直连Pod模式。**
- **Workload绑定：选择引用Workload。**

3. 单击创建服务，完成创建。

YAML 操作指引

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

```
kind: Service
apiVersion: v1
metadata:
annotations:
service.cloud.tencent.com/direct-access: "true" ##开启直连 Pod 模式
name: my-service
spec:
selector:
app: MyApp
ports:
- protocol: TCP
port: 80
targetPort: 9376
type: LoadBalancer
```

annotation 扩展

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

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

注意事项

如何保证滚动更新时的可用性保证

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

直连模式滚动更新的变化

当用户开始为应用做滚动更新的时候，Kubernetes 会根据更新策略进行滚动更新。但其判断一批 Pod 启动的标识仅包括 Pod 自身的状态，并不会考虑该 Pod 在负载均衡上是否配置健康检查且通过。如在接入层组件高负载时，不能及时对此类 Pod 进行及时调度，则滚动更新成功的 Pod 可能并没有正在对外提供服务，从而导致服务的中断。
为了关联滚动更新和负载均衡的后端状态，TKE 接入层组件引入了 Kubernetes 1.12中引入的新特性 ReadinessGate 。TKE 接入层组件仅在确认后端绑定成功并且健康检查通过时，通过配置 ReadinessGate 的状态来使 Pod 达到 Ready 的状态，从而推动整个工作负载的滚动更新。

在集群中使用 ReadinessGate

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

ReadinessGate 机制的灾难恢复

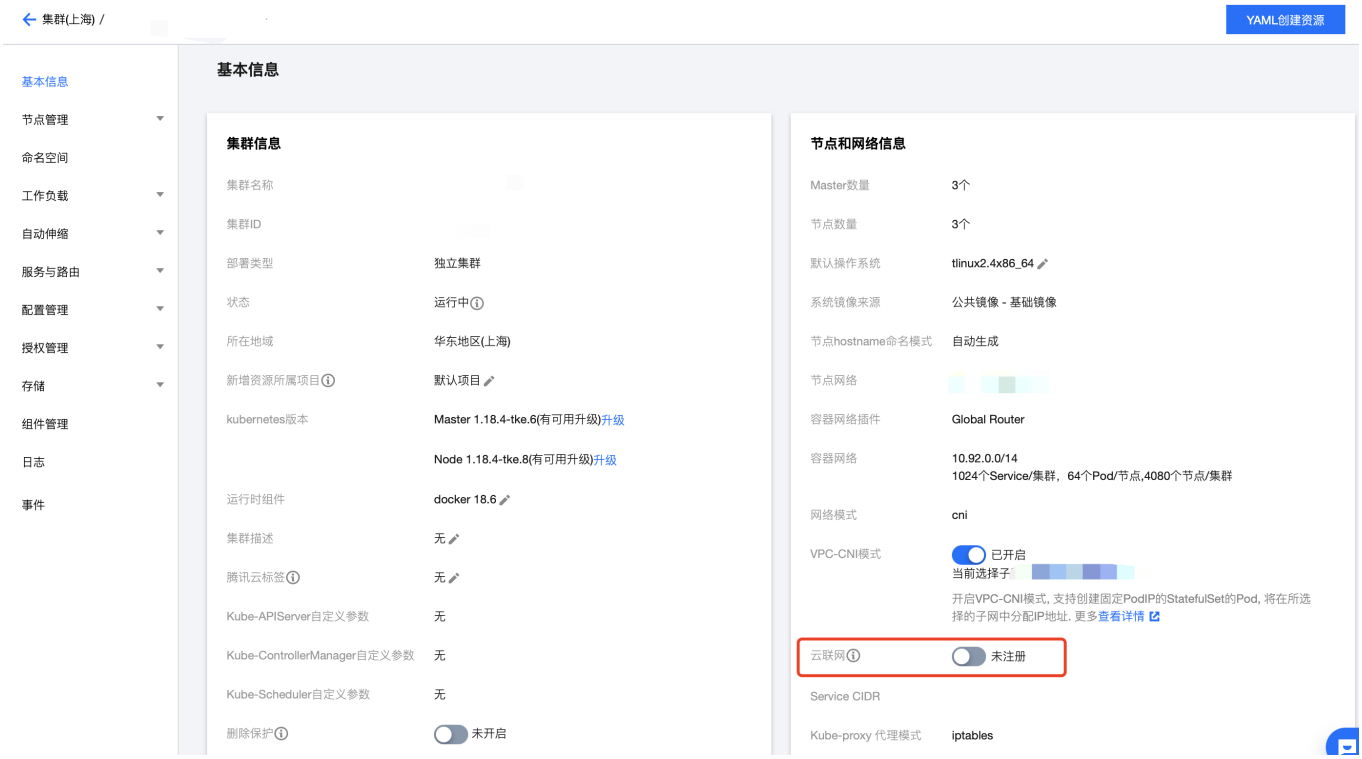
用户集群中的服务注册或证书有可能被用户删除，虽然这些系统组件资源不应该被用户修改或破坏。在用户对集群的探索或是误操作下，这类问题会不可避免的出现。因此接入层组件在启动时会检查以上资源的完整性，在完整性受到破坏时会重建以上资源，加强系统的鲁棒性。详情可参见 Kubernetes Pods ReadinessGate 特性。

容器网络模式为 GlobalRouter

使用限制

- 单个工作负载仅能运行在一种网络模式下，您可选择弹性网卡直连或 GlobalRoute 直连。
- 仅支持带宽上移账号，如若当前账户是传统账号类型（带宽非上移），可参见 账户类型升级说明。
- 默认 CLB 的后端数量限制是 200 个，如果您绑定的工作负载的副本数超过 200 时，可通过 在线咨询 提升负载均衡 CLB 的配额。
- 使用 CLB 直连 Pod，需注意网络链路受云服务器的安全组限制，确认安全组配置是否放开对应的协议和端口，**需要开启 CVM 上工作负载对应的端口**。
- 开启直连后，默认将启用 ReadinessGate 就绪检查，将会在 Pod 滚动更新时检查来自负载均衡的流量是否正常，需要为业务方配置正确的健康检查配置，详情可参见 TkeServiceConfig 介绍。
- 直连 Globalrouter 模式下的 Pod，您可通过以下两种方式进行使用：
  - **通过 云联网 使用。**推荐使用该方式，云联网可以校验绑定的 IP 地址，防止出现绑定出错、地址回环等 IP 绑定常见问题。操作步骤如下：
    a. 创建云联网实例。详情可参见 新建云联网实例。
    b. 将集群所在 VPC 添加至已创建的云联网实例中。

c. 将容器网段注册到云联网，在集群的“基本信息”页面中开启云联网。



- **您可通过 在线咨询 进行申请。此方式缺少云联网的 IP 校验功能，不推荐使用。**

YAML 操作指引

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

前置使用条件

- 在 kube-system/tke-service-controller-config ConfigMap 中新增 GlobalRouteDirectAccess: "true" 以开启 GlobalRoute 直连能力。

在 Service YAML 里开启直连模式

```
kind: Service
apiVersion: v1
metadata:
annotations:
service.cloud.tencent.com/direct-access: "true" ##开启直连 Pod 模式
name: my-service
spec:
selector:
app: MyApp
ports:
- protocol: TCP
port: 80
targetPort: 9376
type: LoadBalancer
```

annotation 扩展

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

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

多 Service 复用 CLB

最近更新时间：2022-07-04 18:30:06

操作场景

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

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

说明事项

- **于2020年8月17日前创建的 TKE 集群，其 Service 创建的 CLB 默认支持复用相同的 CLB。**
  开启复用功能的集群，其中 Service 创建的 CLB 将默认配置 <serviceUUID>:tke-lb-serviceId 和 <serviceUUID>_<lb_listener_id>:<lb_listener_id> 两个标签。每个 CLB 具备单独的 key 和 value，生成的标签数量较多。您可通过 在线咨询 联系我们关闭此类型集群的复用 CLB 功能，并清理标签。
- **于2020年8月17日起创建的 TKE 集群，默认关闭多 Service 复用相同 CLB 的功能。**
  关闭复用功能的集群，其中 Service 创建的 CLB 将默认配置 tke-lb-serviceuuid:<serviceUUID> 标签。所有 Service 使用同一批标签 Key，标签 Key 数量可控。您可通过 在线咨询 联系我们开启需要使用多个 Service 复用相同 CLB 的功能。
- **如果您的集群是 EKS 集群，集群默认已开启了 CLB 复用能力，但需要注意以下内容：**
  i. 用于复用的 CLB 必须为用户手动购买，而非 EKS 自动购买。EKS 自动购买的 CLB 在复用时会报错，是为了保护复用 CLB 的 Service 的 CLB 不被 EKS 回收。
  ii. CLB 购买成功后，需要在 Service 里添加两个 Annotation：
    - service.kubernetes.io/qcloud-share-existed-lb:"true"
    - service.kubernetes.io/tke-existed-lbid:lb-xxx

使用限制

- 在 Service 复用场景下，单个负载均衡管理的监听器数量不能超过10个。
- 在 Service 复用场景下，只能使用用户自行创建的负载均衡。因为容器服务 TKE 集群创建的负载均衡在被复用的情况下，负载均衡资源可能因为无法释放而导致泄漏。
- 如果需要使用当前 TKE 创建的负载均衡资源进行复用，可以在当前 Service 添加 service.kubernetes.io/tke-existed-lbid 注解，并删除该负载均衡上的 tke-createdBy-flag = yes 标签。

> ⚠ 注意：
> 使用当前 TKE 创建的负载均衡资源进行复用后，因为缺少了标签，该 CLB 的生命周期将不由 TKE 侧控制，需要自行管理，请谨慎操作。

操作步骤

1. 参考 创建负载均衡实例，创建集群所在 VPC 下的公网或内网类型的负载均衡。
2. 参考 创建 Deployment 或 创建 Service，创建 Loadbalancer 类型的 Service，选择**使用已有**负载均衡，并选择 步骤1 中创建的负载均衡实例。如下图所示：

3. 重复步骤2，即可完成通过多个 Service 复用相同负载均衡器 CLB。

Service 扩展协议

最近更新时间：2022-05-19 16:41:35

Service 默认支持的协议

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

- 在 直连场景 下接入，使用扩展协议时没有任何限制，支持 TCP 和 UDP 协议混用。
- 非直连场景下，ClusterIP 和 NodePort 模式支持混用。但社区对 LoadBalance 类型的 Service 有限制，目前仅能使用同类型协议。
- 当 LoadBalance 声明为 TCP 时，端口可以使用扩展协议的能力，将负载均衡的协议变更为 TCP_SSL、HTTP、HTTPS。
- 当 LoadBalance 声明为 UDP 时，端口可以使用扩展协议的能力，将负载均衡的协议变更为 UDP。

TKE 扩展 Service 转发协议

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

前置说明

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

注解名称

service.cloud.tencent.com/specify-protocol

扩展协议注解示例

TCP_SSL 示例

```
{"80":{"protocol":["TCP_SSL"],"tls":"cert-secret"}}
```

HTTP 示例

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

HTTPS 示例

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

TCP/UDP混合示例

```
{"80":{"protocol":["TCP","UDP"]}} # 仅[直连模式](https://cloud.tencent.com/document/product/457/41897)支持
```

混合示例

```
{"80":{"protocol":["TCP_SSL","UDP"],"tls":"cert-secret"}} # 仅[直连模式](https://cloud.tencent.com/document/product/457/41897)支持
```

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

扩展协议使用说明

扩展协议YAML使用说明

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

扩展协议控制台使用说明

- 在创建 Service 时，若以“公网LB”或“内网LB”的形式暴露服务，非 直连模式 情况下，“端口映射”中，仅支持 TCP 和 TCP SSL 一起使用。如下图所示：

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

案例说明

原生 Service 不支持协议混用，TKE 经过特殊改造后，在 直连场景 中支持混合协议的使用。

需注意的是，YAML 中仍使用相同的协议，但可以通过 Annotation 明确每个端口的协议类型。如下示例展示了 80 端口使用 TCP 协议，8080 端口使用 UDP 协议。

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

Service Annotation 说明

最近更新时间：2022-03-25 11:03:34

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

注解使用方式

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

Annotation 集合

service.kubernetes.io/loadbalance-id

说明：

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

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

说明：

通过该 Annotation 指定创建内网类型 CLB，取值为子网 ID。

使用示例：

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

service.kubernetes.io/tke-existed-lbid

说明：

使用已存在的 CLB，需注意不同使用方式对腾讯云标签的影响。

使用示例：

使用方式详情见 Service 使用已有 CLB。

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

说明：

Service Local 模式下仅绑定有 Pod 存在的节点。

使用示例：

使用方式详情见 Service Local 模式。

service.cloud.tencent.com/local-svc-weighted-balance

说明：

- 与 Annotation service.kubernetes.io/local-svc-only-bind-node-with-pod 搭配使用。
- CLB 后端的权重将会由节点上工作负载的数量决定。

使用示例：

使用方式详情见 Service Local 模式。

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

说明：

指定标签设置负载均衡后端绑定的节点。

使用示例：

使用方式详情见 指定接入层后端。

service.cloud.tencent.com/direct-access

说明：

使用负载均衡直连 Pod。

使用示例：

使用方式详情见 使用 LoadBalancer 直连 Pod 模式 Service。

service.cloud.tencent.com/tke-service-config

说明：

通过 tke-service-config 配置负载均衡 CLB。

使用示例：

使用方式详情见 Service 负载均衡配置。

service.cloud.tencent.com/tke-service-config-auto

说明：

通过该注解可自动创建 TkeServiceConfig。

使用示例：

使用方式详情见 Service 与 TkeServiceConfig 关联行为。

service.kubernetes.io/loadbalance-nat-ipv6

说明：

只读注解，创建 NAT64 IPv6 负载均衡时，负载均衡的 IPv6 地址将会展示到注解中。

使用示例：

```
service.kubernetes.io/loadbalance-nat-ipv6: "2402:4e00:1402:7200:0:9223:5842:2a44"
```

service.kubernetes.io/loadbalance-type（即将废弃）

说明：

- 控制自动创建的负载均衡类型，传统型负载均衡、应用型负载均衡。
- 可选值：yunapi_clb（传统型）、classic（传统型）、yunapiv3_forward_clb（应用型）
- 默认值：yunapiv3_forward_clb（应用型）

> ⚠ 注意：
> 除非有特殊原因，否则不推荐使用传统型负载均衡，传统型负载均衡已经停止迭代准备下线，并且缺失大量特性。

service.cloud.tencent.com/specify-protocol

说明：

支持通过注解为指定的监听端口配置 TCP、UDP、TCP SSL、HTTP、HTTPS。

使用示例：

使用方式详情见 Service 扩展协议。

service.kubernetes.io/service.extensiveParameters

说明：

该 Annotation 使用的是 CLB 创建时的参数，当前仅在创建时支持配置，创建后不支持修改，创建后修改本注解无效。
参考 创建负载均衡实例 为创建负载均衡追加自定义参数。

使用示例：

- 创建 NAT64 IPv6 实例：
  service.kubernetes.io/service.extensiveParameters: '{"AddressIPVersion":"IPV6"}'
- 购买电信负载均衡：
  service.kubernetes.io/service.extensiveParameters: '{"VipIsp":"CTCC"}'

service.cloud.tencent.com/enable-grace-shutdown

说明：

支持 CLB 直连模式的优雅停机。

使用示例：

仅在直连模式下支持，需要配合使用 service.cloud.tencent.com/direct-access ，使用方式详情见 Service 优雅停机。

kubernetes.io/service.internetChargeType

说明：

负载均衡的付费类型，当前仅在创建时支持配置，创建后不支持修改付费类型，创建后修改本注解无效。
指定创建负载均衡时，负载均衡的付费类型。请配合 kubernetes.io/service.internetMaxBandwidthOut 注解一起使用。

可选值：

BANDWIDTH_POSTPAID_BY_HOUR	按带宽按小时后计费
TRAFFIC_POSTPAID_BY_HOUR	按流量按小时后计费

使用示例：

```
kubernetes.io/service.internetChargeType: "TRAFFIC_POSTPAID_BY_HOUR"
```

kubernetes.io/service.internetMaxBandwidthOut

说明：

CLB 带宽设置，当前仅在创建时支持配置，创建后不支持修改带宽，创建后修改本注解无效。
指定创建负载均衡时，负载均衡的最大出带宽，仅对公网属性的 LB 生效。需配合 kubernetes.io/service.internetChargeType 注解一起使用。

可选值：

范围支持1到2048，单位 Mbps。

使用示例：

```
kubernetes.io/service.internetMaxBandwidthOut: "2048"
```

Ingress 管理
Ingress Controllers 说明

最近更新时间：2022-06-09 14:48:02

各类型 Ingress Controllers 介绍

应用型 CLB

应用型 CLB 是基于腾讯云负载均衡器 CLB 实现的 TKE Ingress Controller，可以配置实现不同 URL 访问到集群内不同的 Service。CLB 直接将流量通过 NodePort 转发至 Pod（CLB 直连 Pod 时直接转发到 Pod），一条 Ingress 配置绑定一个 CLB 实例（IP），适合仅需做简单路由管理，对 IP 地址收敛不敏感的场景。详情可参见 CLB 类型 Ingress。

Istio Ingress Gateway

基于腾讯云负载均衡器 CLB 和 Istio Ingress Gateway（由腾讯云服务网格 TCM 提供）的 Ingress Controller，控制面与相关支撑组件由腾讯云维护，集群内仅需容器化部署执行流量转发的数据面，可使用原生 Kubernetes Ingress 或提供更多精细化流量管理能力的 Istio API。CLB 后增加了一层代理（envoy），适合对接入层路由管理有更多诉求，有 IP 地址收敛诉求，有跨集群、异构部署服务入口流量管理诉求的场景。

专享型 API 网关

专享型 API 网关是基于腾讯云 API 网关专享实例实现的 TKE Ingress Controller，适用于有多个 TKE 集群，需要统一接入层的场景、以及对接入层有认证、流控等诉求的场景。详情可参见 API 网关类型 Ingress。API Gateway Ingress 主要有以下优势：

- API 网关直接连接 TKE 集群的 Pod，无任何中间节点。
- 一个 API 网关 TKE 通道可以同时对接多个 TKE 服务，多个服务间采用加权轮询算法分配流量。
- 支持 API 网关提供的认证鉴权、流量控制、灰度分流、缓存、熔断降级等高级能力拓展。
- 采用 API 网关专享实例支撑，底层物理资源由用户独享，性能稳定，SLA 高。

Nginx Ingress Controller

Nginx Ingress Controller 是基于腾讯云负载均衡器 CLB 和 Nginx 反向代理（容器化部署在集群内）的 Ingress Controller，通过 Annotations 扩展了原生 Kubernetes Ingress 的功能。CLB 后增加了一层代理（nginx），适合对接入层路由管理有更多诉求，及有 IP 地址收敛诉求的场景。详情可参见 Nginx 类型 Ingress。

各类型 Ingress Controllers 功能对比

模块	功能	应用型 CLB	Istio Ingress Gateway（由腾讯云服务网格 TCM 提供）	专享型 API 网关	Nginx Ingress Controller
流量管理	支持协议	http，https	http，https，http2，grpc，tcp，tcp + tls	http，https，http2，grpc	http，https，http2，grpc，tcp，udp
	IP 管理	一条 Ingress 规则对应一个 IP（CLB）	多条 Ingress 规则对应一个 IP（CLB），IP 地址收敛	多条 Ingress 规则对应一个 IP（专享型 API 网关），IP 地址收敛	多条 Ingress 规则对应一个 IP（CLB），IP 地址收敛
	特征路由	host，URL	更多特征支持：header、method、query parameter 等	更多特征支持：header、method、query parameter 等	更多特征支持：header、cookie 等
	流量行为	不支持	支持，重定向，重写等	支持重定向，自定义请求，自定义响应	支持，重定向，重写等
	地域感知负载均衡	不支持	支持	不支持	不支持

模块	功能	应用型 CLB	Istio Ingress Gateway（由腾讯云服务网格 TCM 提供）	专享型 API 网关	Nginx Ingress Controller
应用访问寻址	服务发现	单 Kubernetes 集群	多 Kubernetes 集群 + 异构服务	多 Kubernetes 集群	单 Kubernetes 集群
安全	SSL 配置	支持	支持	支持	支持
	认证授权	不支持	支持	支持	支持
可观测性	监控指标	支持（需要在 CLB 中查看）	支持（云原生监控、云监控）	支持（需要在 API 网关中查看）	支持（云原生监控）
	调用追踪	不支持	支持	不支持	不支持
	组件运维	关联 CLB 已托管，仅需集群内运行 TKE Ingress Controller	控制面已托管，需集群内运行数据面 Ingress Gateway	Kubernetes集群内不需要运行管控面，只需要开启集群内网访问功能	需集群内运行 Nginx Ingress Controller（控制面 + 数据面）

CLB 类型 Ingress

概述

最近更新时间：2022-04-22 17:18:16

Service 提供了基于四层网络的集群内容器服务的暴露能力，Service 暴露类型（例如 ClusterIP、NodePort 或 LoadBalancer）均基于四层网络服务的访问入口，缺少基于七层网络的负载均衡、SSL 或基于名称的虚拟主机等七层网络能力。Ingress 提供七层网络下 HTTP、 HTTPS 协议服务的暴露，及七层网络下的常见能力。

Ingress 基本概念

Ingress 是允许访问到集群内 Service 规则的集合，您可以通过配置转发规则，实现不同 URL 可以访问到集群内不同的 Service。为了使 Ingress 资源正常工作，集群需运行 Ingress Controller，容器服务在集群内默认启用了基于腾讯云负载均衡器实现的 TKE Ingress Controller 。

Ingress 生命周期管理

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

标签	描述
tke-createdBy-flag = yes	• 标识该资源是容器服务创建，拥有该标签的 Ingress 会在销毁时删除对应资源。 • 如果没有该标签，Ingress 会在销毁时，仅删除负载均衡内的监听器资源，而不删除负载均衡自身。
tke-clusterId = <clusterId>	• 标识该资源被哪一个 Cluster 所使用。 • Ingress 会在销毁时，删除对应标签（ClusterId 需正确）。
tke-lb-ingress-uuid = <Ingress UUID>	• 标识该资源被哪一个 Ingress 所使用。 • Ingress 目前不支持复用，当用户指定 Ingress 使用已有负载均衡时，标签的值若不正确会被拒绝。 • Ingress 会在销毁时，删除对应标签（Ingress UUID 需正确）。

Ingress Controller 使用方法

除了腾讯云服务提供的 TKE Ingress Controller 以外，Kubernetes 社区还有各种类型的第三方 Ingress Controller，这些 Ingress 控制器均为完成服务的七层网络暴露。Kubernetes 社区基本支持使用 kubernetes.io/ingress.class 注解用于区分各种 Ingress 控制器，以确定当前 Ingress 资源应被哪一个控制器处理。TKE Ingress Controller 也支持使用该注解，具体规则及使用建议如下：

- 当 Ingress 资源没有描述注解 kubernetes.io/ingress.class 时，TKE Ingress Controller 会管理当前 Ingress 资源。
- 当 Ingress 资源有注解 kubernetes.io/ingress.class 且值为 qcloud 时，TKE Ingress Controller 会管理当前 Ingress 资源。
- 当 Ingress 资源修改注解 kubernetes.io/ingress.class 的内容时，TKE Ingress Controller 会根据注解内容将其纳入或脱离管理范围，其操作会涉及到资源的创建与释放。
- 当您确认完全不需要使用 TKE Ingress Controller 时，可以将集群中的 Deployment（kube-system:l7-lb-controller）的工作副本数量调整为0，从而关闭 TKE Ingress Controller 功能。

> 说明：
> 关闭该功能前，请确保集群中没有被 TKE Ingress Controller 管理的 Ingress 资源，避免出现负载均衡资源释放失败的情况。

Ingress 相关操作

Ingress 相关操作及功能如下，您可参考以下文档进一步了解：

- Ingress 基本功能
- Ingress 使用已有 CLB
- Ingress 使用 TkeServiceConfig 配置 CLB
- Ingress 证书配置

Ingress 基本功能

最近更新时间：2022-04-18 14:14:05

简介

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

注意事项

- 确保您的容器业务不和 CVM 业务共用一个 CLB。
- 不支持您在 CLB 控制台操作 TKE 管理的 CLB 的监听器、转发路径、证书和后端绑定的服务器，您的更改会被 TKE 自动覆盖。
- 使用已有的 CLB 时：
  - 只能使用通过 CLB 控制台创建的负载均衡器，不支持复用由 TKE 自动创建的 CLB。
  - 不支持多个 Ingress 复用 CLB。
  - 不支持 Ingress 和 Service 共用 CLB。
  - 删除 Ingress 后，复用 CLB 绑定的后端云服务器需要自行解绑，同时会保留一个 tag tke-clusterId: cls-xxxx，需自行清理。

Ingress 控制台操作指引

创建 Ingress

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**进入集群管理页面。
3. 单击需要创建 Ingress 的集群 ID，进入待创建 Ingress 的集群管理页面。
4. 选择**服务** > **Ingress**，进入 Ingress 信息页面。
5. 单击**新建**，进入“新建Ingress”页面。如下图所示：

6. 根据实际需求，设置 Ingress 参数。关键参数信息如下：
   - Ingress名称：自定义。
   - 网络类型：默认为 “公网”，请根据实际需求进行选择。
   - IP 版本：提供 IPv4 和 IPv6 NAT64 两种版本，请根据实际需求进行选择。
   - 负载均衡器：可自动创建或使用已有 CLB。
   - 命名空间：根据实际需求进行选择。
   - 监听端口：默认为**Http:80**，请根据实际情况进行选择。
     如果勾选**Https:443**则需绑定服务器证书，以保证访问安全。如下图所示：

     详情请参见 SSL 证书格式要求及格式转换说明。
   - 转发配置：根据实际需求进行设置。
7. 单击**创建Ingress**，完成创建。

更新 Ingress

更新 YAML

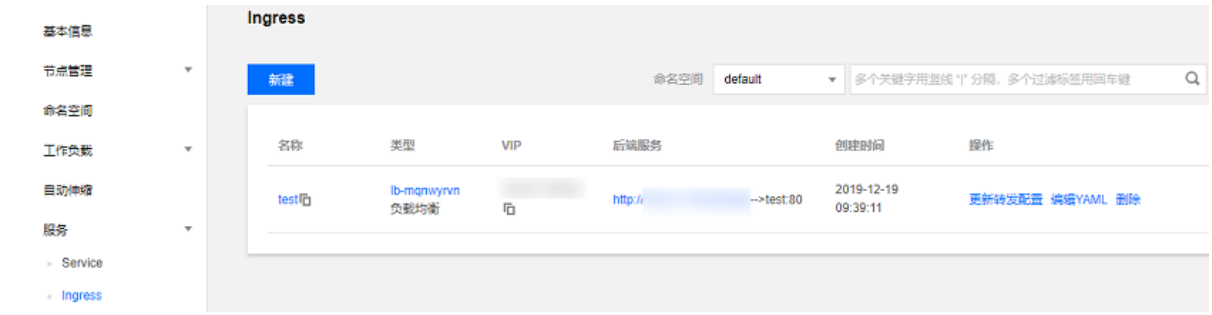
1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**，进入集群管理页面。
3. 单击需要更新 YAML 的集群 ID，进入待更新 YAML 的集群管理页面。
4. 选择**服务** > **Ingress**，进入 Ingress 信息页面。如下图所示：

5. 在需要更新 YAML 的 Ingress 行中，单击**编辑YAML**，进入更新 Ingress 页面。
6. 在 “更新Ingress” 页面，编辑 YAML，单击**完成**，即可更新 YAML。

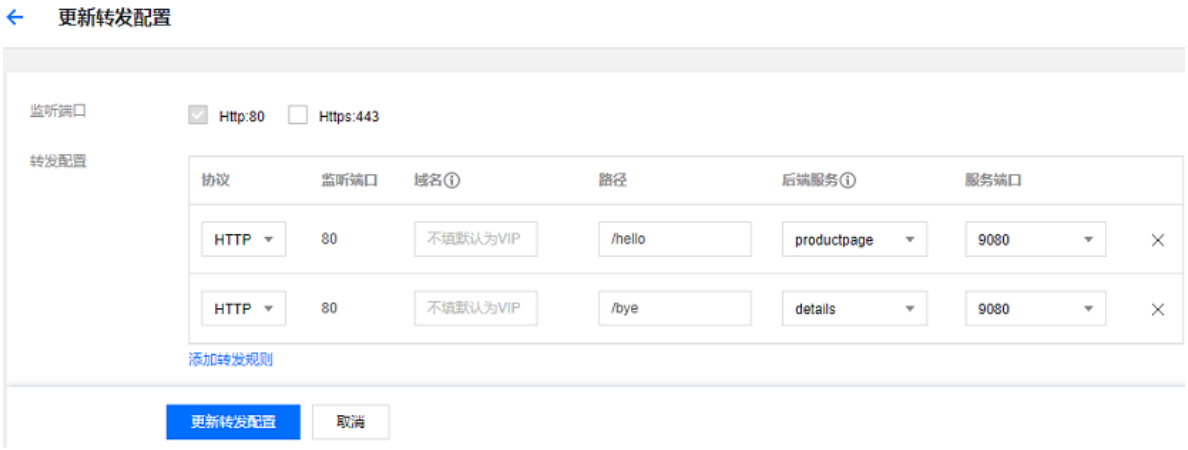
更新转发规则

1. 集群管理页面，单击需要更新 YAML 的集群 ID，进入待更新 YAML 的集群管理页面。

2. 选择**服务** > **Ingress**，进入 Ingress 信息页面。如下图所示：



3. 在需要更新转发规则的 Ingress 行中，单击**更新转发配置**，进入更新转发配置页面。如下图所示：



4. 根据实际需求，修改转发配置，单击**更新转发配置**，即可完成更新。

Kubectl 操作 Ingress 指引

YAML 示例

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
annotations:
kubernetes.io/ingress.class: qcloud ## 可选值：qcloud（CLB类型ingress），nginx（nginx-ingress），traefik
## kubernetes.io/ingress.existLbId： lb-xxxxxxxx ##指定使用已有负载均衡器创建公网/内网访问的Ingress
## kubernetes.io/ingress.subnetId: subnet-xxxxxxxx ##若是创建CLB类型内网ingress需指定该条annotation
name: my-ingress
namespace: default
spec:
rules:
- host: localhost
http:
paths:
- backend:
serviceName: non-service
servicePort: 65535
path: /
```

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

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

如果您已有的应用型 CLB 为空闲状态，需要提供给 TKE 创建的 Ingress 使用，或期望在集群内使用相同的 CLB ，您可以通过以下 annotations 进行设置：

> 说明：
> 请了解 注意事项 后开始使用。

```
metadata:
annotations:
kubernetes.io/ingress.existLbId： lb-6swtxxxx
```

annotations: 创建 CLB 类型内网 Ingress

如果您需要使用内网负载均衡，可以通过以下 annotations 进行设置：

```
metadata:
annotations:
kubernetes.io/ingress.subnetId: subnet-xxxxxxxx
```

说明事项

如果您使用的是 IP 带宽包 账号，在创建公网访问方式的服务时需要指定以下两个 annotations 项：

- kubernetes.io/ingress.internetChargeType 公网带宽计费方式，可选值有：
  - TRAFFIC_POSTPAID_BY_HOUR（按使用流量计费）
  - BANDWIDTH_POSTPAID_BY_HOUR（按带宽计费）
- kubernetes.io/ingress.internetMaxBandwidthOut 带宽上限，范围：[1,2000] Mbps。

  例如：

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

  关于 IP 带宽包 的更多详细信息，欢迎查看文档 共享带宽包产品类别。

创建 Ingress

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

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

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

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

4. 执行以下命令，验证创建是否成功。

   ```
   kubectl get ingress
   ```

   返回类似以下信息，即表示创建成功。

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

更新 Ingress

方法一

执行以下命令，更新 Ingress。

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

方法二

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

```
kubectl create/apply
```

Ingress 使用已有 CLB

最近更新时间：2022-04-13 16:40:18

腾讯云容器服务 TKE 具备通过 kubernetes.io/ingress.existLbId: <LoadBalanceId> 注解使用已有负载均衡的功能，您可使用该注解指定 Ingress 关联的负载均衡实例。

> 说明：
> Ingress 与 Service 的区别：Ingress 不支持多个实例使用同一个负载均衡实例，即不支持复用功能。

注意事项

- 请确保您的容器业务不与云服务器 CVM 业务共用一个负载均衡资源。
- 不支持在负载均衡控制台操作 Ingress Controller 管理的负载均衡监听器以及后端绑定的服务器，更改会被 Ingress Controller 自动覆盖。
- 使用已有负载均衡时：
  - 不支持多个 Ingress 复用同一个负载均衡。
  - 指定的负载均衡不能存在任何已有监听器。如已存在，请提前删除。
  - 仅支持使用通过负载均衡控制台创建的负载均衡器，不支持使用由 Service Controller 自动创建和管理的负载均衡，即 Service 和 Ingress 不能混用同一个负载均衡。
  - Ingress Controller 不负责负载均衡的资源管理，即在 Ingress 资源删除时，负载均衡资源不会被删除回收。

使用场景

使用包年包月的负载均衡对外提供服务

Ingress Controller 管理负载均衡生命周期时，仅支持购买按量计费的资源。但由于包年包月的负载均衡在价格上有一定的优势，用户需要长时间使用负载均衡时，通常会优先选择购买包年包月负载均衡。

在此类场景下，用户就可以独立购买和管理负载均衡。通过注解控制 Ingress 使用已有负载均衡，将负载均衡的生命周期管理从 Ingress Controller 中剥离。示例如下：

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
annotations:
kubernetes.io/ingress.existLbId: lb-mgzu3mpx
name: nginx-ingress
spec:
rules:
- http:
paths:
- backend:
serviceName: nginx-service
servicePort: 80
path: /
```

> 说明：
> kubernetes.io/ingress.existLbId: lb-mgzu3mpx 注解表明了该 Ingress 将使用已有负载均衡 lb-mgzu3mpx 进行 Ingress 服务配置。

Ingress 使用 TkeServiceConfig 配置 CLB

最近更新时间：2022-06-09 11:38:40

TkeServiceConfig

TkeServiceConfig 是腾讯云容器服务 TKE 提供的自定义资源 CRD，通过 TkeServiceConfig 能够帮助您更灵活的进行 Ingress 管理负载均衡的各种配置。

使用场景

Ingress YAML 的语义无法定义的负载均衡参数和功能，可以通过 TkeServiceConfig 来配置。

配置说明

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

> ⚠ 注意：
> TkeServiceConfig 资源需要和 Ingress 处于同一命名空间。

TkeServiceConfig 不会帮您配置并修改协议、端口、域名以及转发路径，您需要在配置中描述协议、端口、域名还有转发路径以便指定配置下发的转发规则。

每个七层的监听器下可有多个域名，每个域名下可有多个转发路径。因此，在一个 TkeServiceConfig 中可以声明多组域名、转发规则配置，目前主要针对负载均衡的健康检查以及对后端访问提供配置。

- 通过指定协议和端口，配置能够被准确地下发到对应监听器：
  - spec.loadBalancer.l7Listeners.protocol：七层协议
  - spec.loadBalancer.l7Listeners.port：监听端口
- 通过指定协议、端口、域名以及访问路径，可以配置转发规则级别的配置。例如，后端健康检查、负载均衡方式。
  - spec.loadBalancer.l7Listeners.protocol：七层协议
  - spec.loadBalancer.l7Listeners.port：监听端口
  - spec.loadBalancer.l7Listeners.domains[].domain：域名
  - spec.loadBalancer.l7Listeners.domains[].rules[].url：转发路径
  - spec.loadBalancer.l7listeners.protocol.domain.rules.url.forwardType: 指定后端协议
    - 后端协议是指 CLB 与后端服务之间的协议：后端协议选择 HTTP 时，后端服务需部署 HTTP 服务。后端协议选中 HTTPS 时，后端服务需部署 HTTPS 服务，HTTPS 服务的加解密会让后端服务消耗更多资源。更多请查看 CLB 配置 HTTPS 监听器

> 说明：
> 当您的域名配置为默认值，即公网或内网 VIP 时，可以通过 domain 填空值的方式进行配置。

Ingress 与 TkeServiceConfig 关联行为

1. 创建 Ingress 时，设置 **ingress.cloud.tencent.com/tke-service-config-auto:<true>** ，将自动创建 <IngressName>-auto-ingress-config。 您也可以通过 **ingress.cloud.tencent.com/tke-service-config:<config-name>** 直接指定您自行创建的 TkeServiceConfig。两个注解不可同时使用。
2. 您为 Service\Ingress 使用的自定义配置，名称不能以 -auto-service-config 与 -auto-ingress-config 为后缀。
3. 其中自动创建的 TkeServiceConfig 存在以下同步行为：
   - 更新 Ingress 资源时，新增若干7层转发规则，如果该转发规则没有对应的 TkeServiceConfig 配置片段。Ingress-Controller 将主动添加 TkeServiceConfig 对应片段。
   - 删除若干7层转发规则时，Ingress-Controller 组件将主动删除 TkeServiceConfig 对应片段。
   - 删除 Ingress 资源时，联级删除该 TkeServiceConfig。
   - 用户修改 Ingress 默认的 TkeServiceConfig，TkeServiceConfig 内容同样会被应用到负载均衡。
4. 您也可以参考下列 TkeServiceConfig 完整配置参考，自行创建需要的 CLB 配置，Service 通过注解 **ingress.cloud.tencent.com/tke-service-config:<config-name>** 引用该配置。
5. 其中您手动创建的 TkeServiceConfig 存在以下同步行为：

- 当用户在 Ingress 中使用配置注解时，负载均衡将会即刻进行设置同步。
- 当用户在 Ingress 中删除配置注解时，负载均衡将会保持不变。
- 修改 TkeServiceConfig 配置时，引用该配置的 Ingress 的负载均衡将会根据新的 TkeServiceConfig 进行设置同步。
- 当 Ingress 的监听器没有找到对应配置时，该监听器将不会进行修改。
- Ingress 的监听器找到对应配置时，若配置中没有声明的属性，该监听器将不会进行修改。

示例

Deployment 示例：jetty-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: jetty
name: jetty-deployment
namespace: default
spec:
progressDeadlineSeconds: 600
replicas: 3
revisionHistoryLimit: 10
selector:
matchLabels:
app: jetty
strategy:
rollingUpdate:
maxSurge: 25%
maxUnavailable: 25%
type: RollingUpdate
template:
metadata:
creationTimestamp: null
labels:
app: jetty
spec:
containers:
- image: jetty:9.4.27-jre11
imagePullPolicy: IfNotPresent
name: jetty
ports:
- containerPort: 80
protocol: TCP
- containerPort: 443
protocol: TCP
resources: {}
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
dnsPolicy: ClusterFirst
restartPolicy: Always
schedulerName: default-scheduler
securityContext: {}
terminationGracePeriodSeconds: 30
```

Service 示例：jetty-service.yaml

```
apiVersion: v1
kind: Service
metadata:
name: jetty-service
namespace: default
spec:
ports:
- name: tcp-80-80
port: 80
protocol: TCP
targetPort: 80
- name: tcp-443-443
port: 443
protocol: TCP
targetPort: 443
selector:
app: jetty
type: NodePort
```

该示例包含以下配置：
Service 的 NodePort 类型，声明了两个 TCP 服务。一个在80端口，一个在443端口。

Ingress：jetty-ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
annotations:
kubernetes.io/ingress.rule-mix: "true"
kubernetes.io/ingress.http-rules: '[{"path":"/health","backend":{"serviceName":"jetty-service","servicePort":"80"}}]'
kubernetes.io/ingress.https-rules: '[{"path":"/","backend":{"serviceName":"jetty-service","servicePort":"443","host":"sample.tencent.com"}}]'
ingress.cloud.tencent.com/tke-service-config: jetty-ingress-config
# 指定已有的 tke-service-config
# service.cloud.tencent.com/tke-service-config-auto: true
# 自动创建 tke-service-config
name: jetty-ingress
namespace: default
spec:
rules:
- http:
paths:
- backend:
serviceName: jetty-service
servicePort: 80
path: /health
- host: "sample.tencent.com"
http:
paths:
- backend:
serviceName: jetty-service
```

```
servicePort: 443
path: /
tls:
- secretName: jetty-cert-secret
```

该示例包含以下配置：

- **使用了混合协议，使用默认域名（公网 IP）暴露了一个 HTTP 服务，使用** sample.tencent.com **域名暴露了一个 HTTPS 服务。**
- **HTTP 服务的转发路径是** /health**，HTTPS 服务的转发路径是/。**
- **使用了** jetty-ingress-config **负载均衡配置。**

TkeServiceConfig 示例：jetty-ingress-config.yaml

```
apiVersion: cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
name: jetty-ingress-config
namespace: default
spec:
loadBalancer:
l7Listeners:
- protocol: HTTP
port: 80
domains:
- domain: "" # domain为空表示使用VIP作为域名
rules:
- url: "/health"
forwardType: HTTP # 指定后端协议为 HTTP
healthCheck:
enable: false
- protocol: HTTPS
port: 443
domains:
- domain: "sample.tencent.com"
rules:
- url: "/"
forwardType: HTTPS # 指定后端协议为 HTTPS
session:
enable: true
sessionExpireTime: 3600
healthCheck:
enable: true
intervalTime: 10
healthNum: 2
unHealthNum: 2
httpCheckPath: "/checkHealth"
httpCheckDomain: "sample.tencent.com" #注意：健康检查必须使用固定域名进行探测，如果您在.spec.loadBalancer.l7Listeners.protocol.domains.domain 里填写的是泛域名，一定要使用 httpCheckDomain 字段明确具体需要健康检查的域名，否则泛域名不支持健康检查。
httpCheckMethod: HEAD
scheduler: WRR
```

该示例包含以下配置：

该 TkeServiceConfig **名称为** jetty-ingress-config**。且在七层监听器配置中，声明了两段配置：**

1. 80端口的 HTTP 监听器将会被配置，其中包含域名配置，是默认域名对应负载均衡的 VIP。
   /health 路径下的健康检查被关闭了。
2. 443端口的 HTTPS 监听器将会被配置。其中包含域名配置，域名是 sample.tencent.com。该域名下仅描述了一个转发路径为/的转发规则配置，其中配置包含以下内容：
   - 打开健康检查，健康检查间隔调整为10s，健康阈值2次，不健康阈值2次。通过 HEAD 请求进行健康检查，检查路径为 /checkHealth，检查域名为 sample.tencent.com。
   - 打开会话保持功能，会话保持的超时时间设置为3600s。
   - 转发策略配置为：按权重轮询。

kubectl 配置命令

```
➜ kubectl apply -f jetty-deployment.yaml
➜ kubectl apply -f jetty-service.yaml
➜ kubectl apply -f jetty-ingress.yaml
➜ kubectl apply -f jetty-ingress-config.yaml

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

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

# 更新修改TkeServiceConfig配置
➜ kubectl edit tkeserviceconfigs.cloud.tencent.com jetty-ingress-config
tkeserviceconfigs.cloud.tencent.com/jetty-ingress-config edited
```

Ingress 跨域绑定

最近更新时间：2022-03-16 16:49:19

简介

使用 CLB 型 Ingress 时，默认是在当前集群所在 VPC 内的随机可用区生成 CLB。现目前 TKE 的 CLB Ingress 已支持指定可用区、包括其他地域的可用区。本文将为您介绍如何通过控制台和 YAML 两种方式为 CLB Ingress 跨域绑定和指定可用区。

应用场景

- 需要支持 CLB 的跨地域接入或跨 VPC 接入，即 CLB 所在的 VPC 和当前集群所在的 VPC 不在同一 VPC 内。
- 需要指定 CLB 的可用区以实现资源的统一管理。

> 说明：
>
> 1. 如需使用非本集群所在 VPC 的 CLB，需先通过 云联网 打通当前集群 VPC 和 CLB 所在的 VPC。
> 2. 在确保 VPC 已经打通之后，请 在线咨询 申请使用该功能。
> 3. 以下 YAML 中，需要您输入地域 ID ，您可以通过 地域和可用区 查看地域 ID。

操作步骤

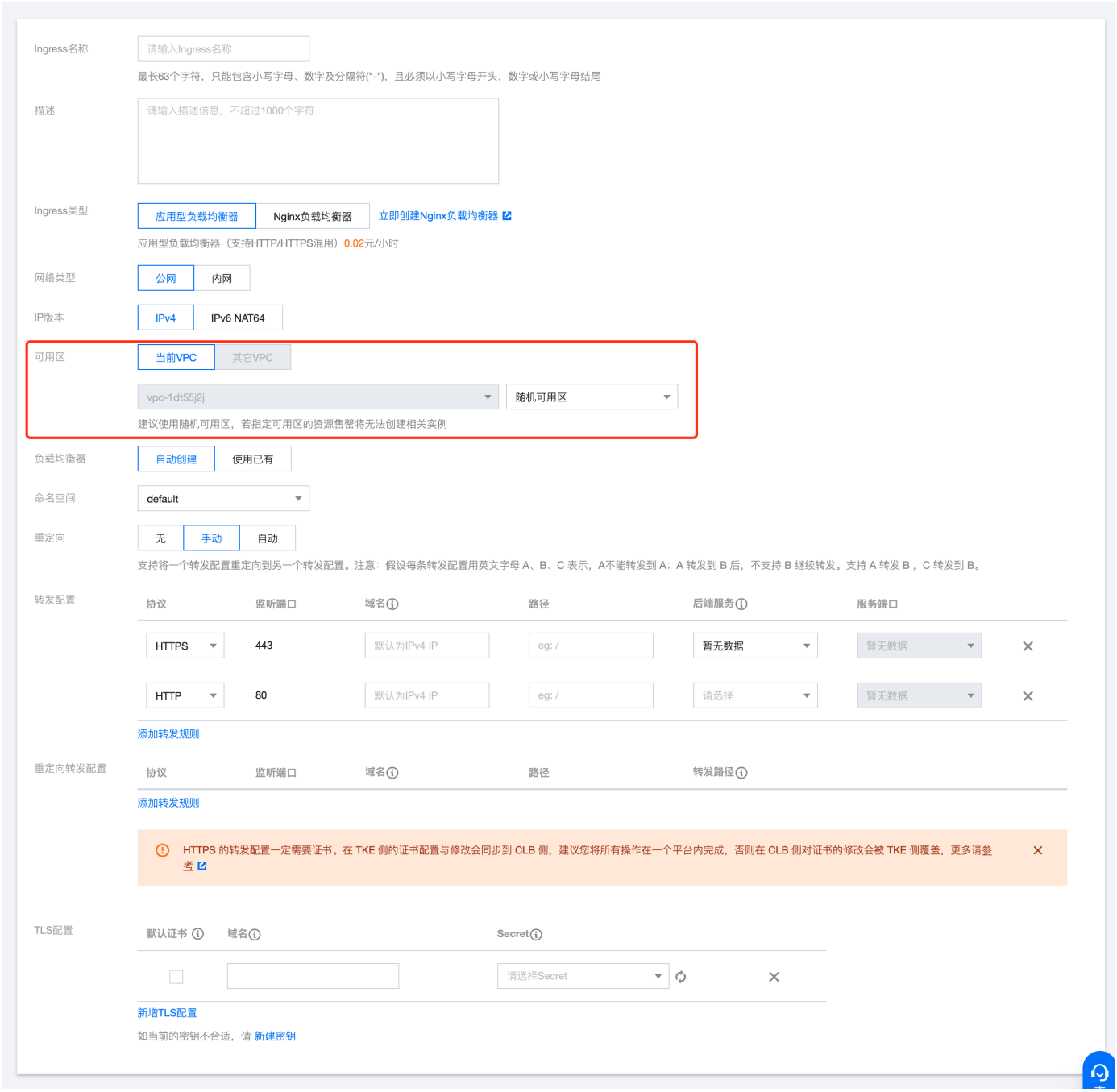
CLB Ingress 跨域绑定和指定可用区支持通过控制台和 YAML 两种方式进行操作，操作步骤如下：

控制台方式

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面，选择需修改 Ingress 的集群 ID。
3. 在集群详情页，选择左侧**服务与路由** > **Ingress**。如下图所示：

4. 单击**新建**，在“新建 Ingress”页面中配置相关可用区规则。配置规则说明如下：
   - **当前VPC**：使用本集群所在 VPC 内的 CLB，建议使用随机可用区，若指定可用区的资源售罄将无法创建相关实例。

- **其它VPC**：仅支持通过 云联网 与当前集群的 VPC 打通的其他 VPC。建议使用随机可用区，若指定可用区的资源售罄将无法创建相关实例。



YAML 方式

> **说明**
> 1. 如需使用非本集群所在 VPC 的 CLB，需先通过 云联网 打通当前集群 VPC 和 CLB 所在的 VPC。
> 2. 在确保 VPC 已经打通之后，请 在线咨询 申请使用该功能。

示例1

如果仅需要指定本集群所在 VPC 的可用区，例如集群的 VPC 在广州地域，CLB Ingress 需要指定广州一区的 CLB，可以在 Ingress 的 YAML 中添加如下 annotation：

```
kubernetes.io/ingress.extensiveParameters: '{"ZoneId":"ap-guangzhou-1"}'
```

示例2

如需使用非本集群所在 VPC 内的 CLB，需先添加如下两条 annotation：

```
ingress.cloud.tencent.com/cross-region-id:
ingress.cloud.tencent.com/cross-vpc-id:
```

具体示例如下：

- 创建异地接入的负载均衡，需先添加如下两条 annotation：

```
ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"
ingress.cloud.tencent.com/cross-vpc-id: "vpc-646vhcjj"
```

> **注意**
> 若您还需指定可用区，则需要再添加示例1中的 annotation。

- 选择已有负载均衡进行异地接入，添加如下两条 annotation：

```
ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"
kubernetes.io/ingress.existLbId: "lb-342wppll"
```

> **注意**
> 若您还需指定可用区，则需要再添加示例1中的 annotation。

完整 Ingress Annotation 说明请参见 Ingress Annotation 说明。

Ingress 重定向

最近更新时间：2022-04-18 14:17:22

简介

域名重定向，指当用户通过浏览器访问某个 URL 时，Web 服务器被设置自动跳转到另外一个 URL。

应用场景

- 网站支持 HTTP 和 HTTPS，例如 http://tencent.com 和 https://tencent.com 需要访问到同一个 Web 服务。
- 网站更换过域名，例如 https://tengxun.com 更换为 https://tencent.com，两个域名访问到同一个 Web 服务。
- 网站部分内容做过调整，原始 URL 已经无法访问，可以重定向到一个新的提供服务的 URL。

⚠
- 当用户使用重定向后，将会多出如下一条注解，该注解表明 Ingress 的转发规则由 TKE 管理，后期不能被删除和修改，否则将和 CLB 侧设置的重定向规则冲突。

```
ingress.cloud.tencent.com/rewrite-support: "true"
```

- 假设用字母表示域名地址，若 A 已经重定向至 B，则：
  - A 不能再重定向至 C（除非先删除旧的重定向关系，再建立新的重定向关系）。
  - B 不能重定向至任何其他地址。
  - A 不能重定向到 A。

重定向有如下两种方式：

- **自动重定向**：用户需要先创建出一个 HTTPS:443 监听器，并在其下创建转发规则。通过调用本接口，系统会自动创建出一个 HTTP:80 监听器（如果之前不存在），并在其下创建转发规则，与 HTTPS:443 监听器下的域名等各种配置对应。
- **手动重定向**：用户手动配置原访问地址和重定向地址，系统自动将原访问地址的请求重定向至对应路径的目的地址。同一域名下可以配置多条路径作为重定向策略，实现 HTTP 和 HTTPS 之间请求的自动跳转。

注意事项

- 若您没有 TKE Ingress 重定向注解声明，会兼容原有不管理重定向规则的逻辑，即：您可以在负载均衡 CLB 的控制台里面配置重定向规则，TKE Ingress 不处理用户在 CLB 控制台配置的这些重定向规则。
- 若您没有 TKE Ingress 重定向注解声明，因为 CLB 的重定向保护限制，如果转发配置 A 重定向到转发配置 B，此时无法直接删除转发配置 B，必须先删掉该重定向规则，才能删除转发配置 B。
- 若您使用 TKE Ingress 重定向注解声明，CLB 下所有重定向规则都是由 TKE Ingress 管理，所有重定向规则仅在 TKE Ingress 里面的相关 Annotation 里面生效，此时用户在 CLB 控制台如果修改重定向配置，最终会被 TKE Ingress 里配置的重定向规则覆盖。

操作步骤

Ingress 支持通过控制台和 YAML 两种方式进行重定向，具体步骤如下：

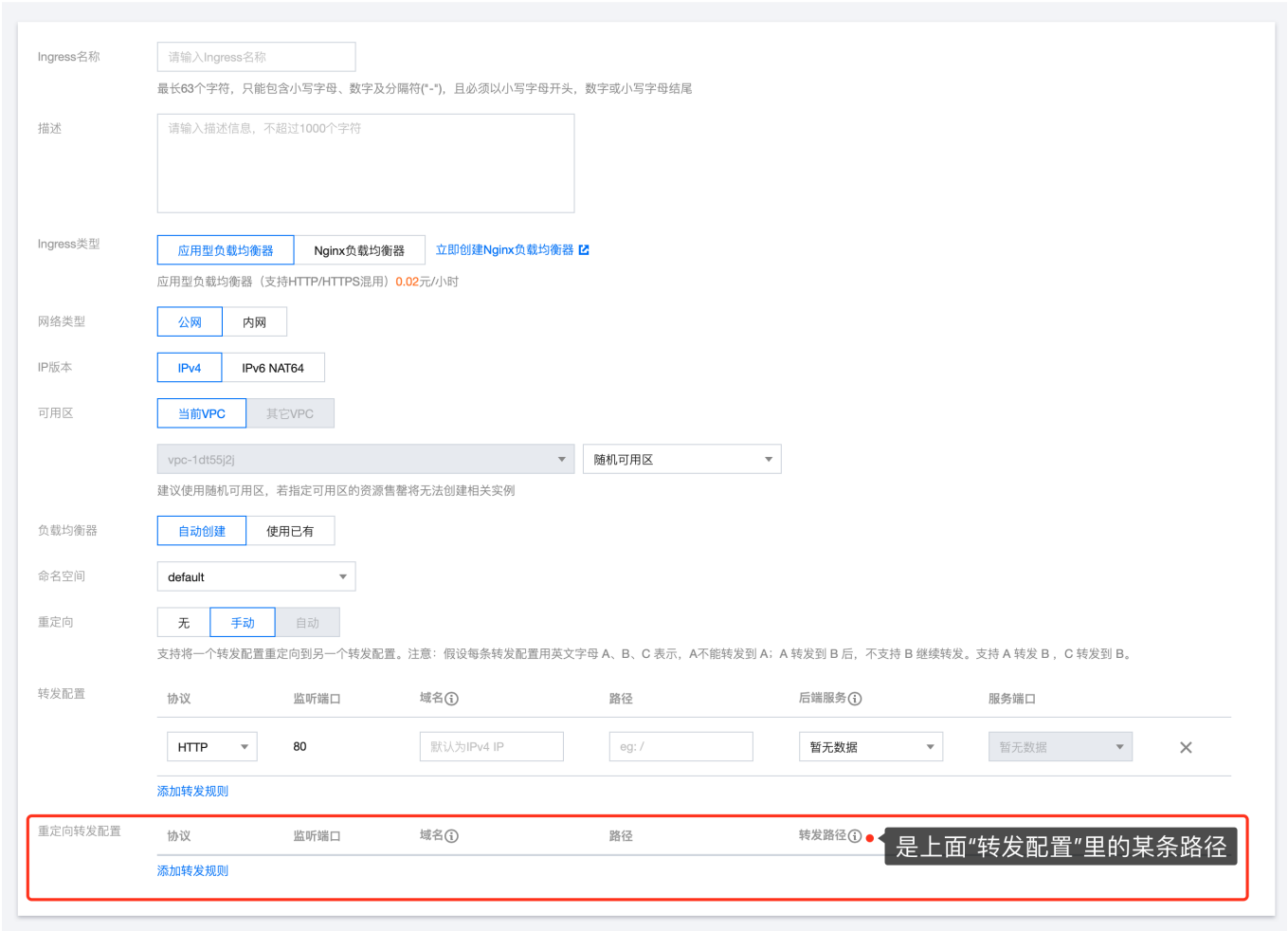
控制台方式

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面，选择需修改 Ingress 的集群 ID。

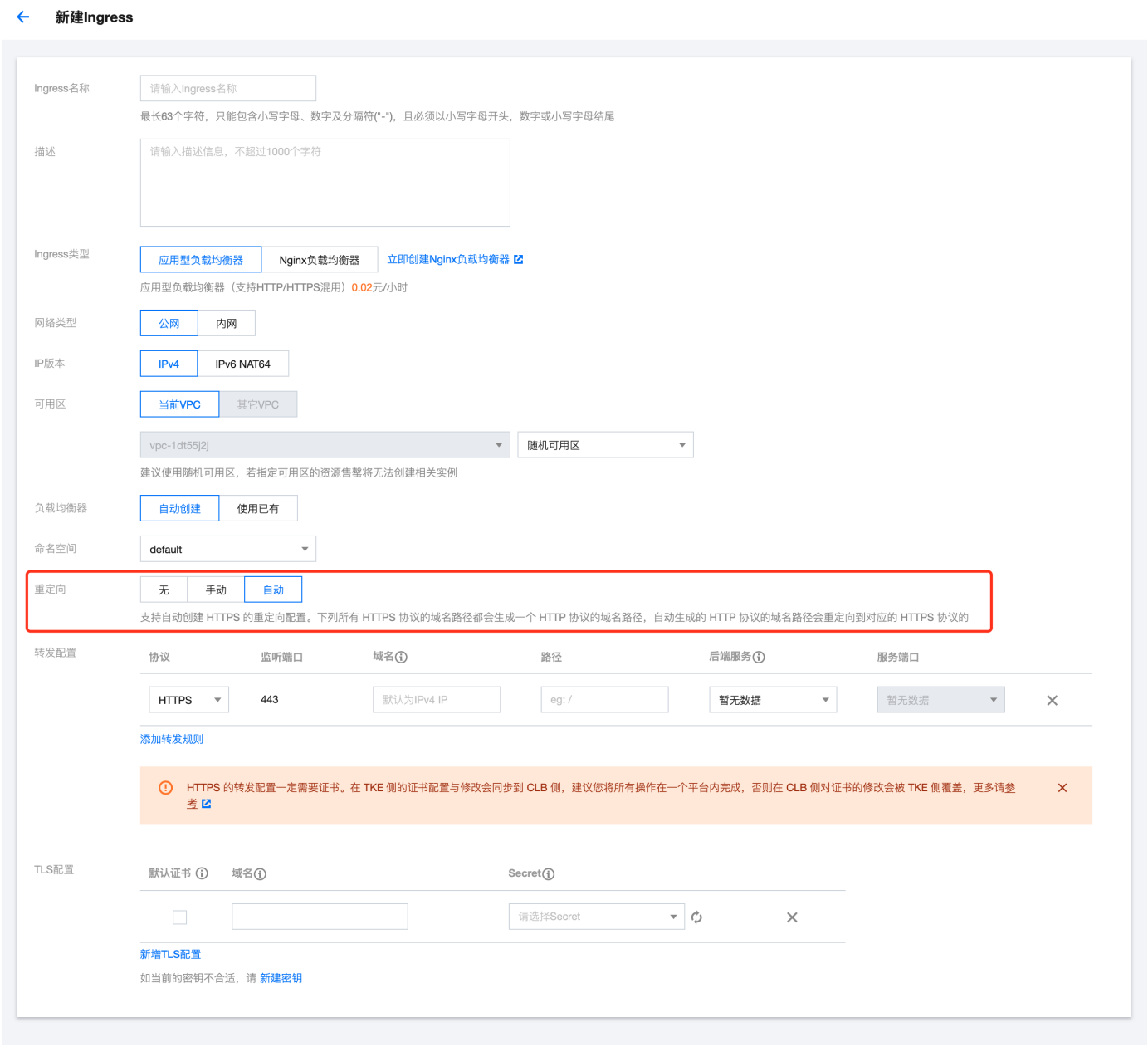
3. 在集群详情页，选择左侧**服务与路由** > **Ingress**。如下图所示：

4. 单击**新建**，在“新建 Ingress”页面中配置相关重定向规则。配置规则说明如下：
   - **无**：不使用重定向规则。
   - **手动**：会在“**转发配置**”下方出现一栏“**重定向转发配置**”。
     - “**转发配置**”里面填写的方式和普通 Ingress 的转发配置一样，后端是某个服务。
     - “**重定向转发配置**”里面填写的方式和普通 Ingress 的转发配置一样，但后端是某个“转发配置”里的某条路径。

     ← **新建Ingress**



   - **自动**：仅对“**转发配置**”里的协议为 “HTTPS” 的路径生效，都将自动生成一个 “HTTP” 的路径，路径完全一样，只有协议不一样。“HTTP” 的路径的转发规则自动重定向到 “HTTPS” 的路径。



YAML 方式

自动重定向：HTTP 重定向到 HTTPS

> ⚠ 仅对 HTTPS 协议的转发规则生效。

在 Ingress YAML 中配置如下注解：

```
ingress.cloud.tencent.com/auto-rewrite: "true"
```

配置该注解之后，转发路径中的所有 HTTPS 监听器中存在的七层转发规则都将被对应到 HTTP 监听器中作为重定向规则。域名与路径都保持一致。

手动重定向

手动重定向需要增加一个 annotation ，修改一个 annotation：

- 增加的 annotation：

```
ingress.cloud.tencent.com/rewrite-support: "true" # 表示允许重定向
```

- 修改的 annotation：

```
# 原注解 格式：
{
"host": "<domain>",
"path": "<path>",
"backend": {
"serviceName": "<service name>",
"servicePort": "<service port>"
}
}

# 新注解格式：
{
"host": "<domain>",
"path": "<path>",
"backend": {
"serviceName": "<service name>",
"servicePort": "<service port>"
},
"rewrite": {
"port": "<rewrite-port>",
"host": "<rewrite-domain>",
"path": "<rewrite-path>"
}
}
```

示例

某服务之前通过 121.4.25.44/path2 **进行访问，在发布新版本之后，期望通过** 121.4.25.44/v2/path2 **进行访问。可以进行如下修改：**

- 增加一条 annotation：

```
ingress.cloud.tencent.com/rewrite-support: "true" # 允许重定向
```

- 修改原 annotation：

```
# 将 /v1/path1 替换为 /path1 80端口；把 /v2/path2 替换为 /path2 80端口。注意：host 可以省略
kubernetes.io/ingress.http-rules: '
[{
"path": "/path1",
"backend": {
"serviceName": "path1",
"servicePort": "80"
}
},
{
"path": "/path2",
"backend": {
"serviceName": "path2",
"servicePort": "80"
}
```

```
},
{
"path": "/v1/path1",
"rewrite": {
"port": 80,
"path": "/path1"
}
},
{
"path": "/v2/path2",
"rewrite": {
"port": 80,
"path": "/path2"
}
}
]'
```

修改后的 YAML：

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
annotations:
description: test
ingress.cloud.tencent.com/rewrite-support: "true"
kubernetes.io/ingress.class: qcloud
kubernetes.io/ingress.http-rules: '[{"path":"/path1","backend":{"serviceName":"path1","servicePort":"80"}},{"path":"/path2","back
end":{"serviceName":"path2","servicePort":"80"}},{"path":"/v1/path1","rewrite":{"port":80,"path":"/path1"}},{"path":"/v2/path
2","rewrite":{"port":80,"path":"/path2"}}]'
kubernetes.io/ingress.https-rules: "null"
kubernetes.io/ingress.rule-mix: "true"
name: test
namespace: default
spec:
rules:
- http:
paths:
- backend:
serviceName: path1
servicePort: 80
path: /path1
pathType: ImplementationSpecific
- http:
paths:
- backend:
serviceName: path2
servicePort: 80
path: /path2
pathType: ImplementationSpecific
status:
loadBalancer:
ingress:
- ip: 121.4.25.44
```

完整 Ingress Annotation 说明请参见 Ingress Annotation 说明 文档。

Ingress 混合使用 HTTP 及 HTTPS 协议

最近更新时间：2022-04-22 17:18:11

混合规则

默认场景下，当 Ingress 中不配置 TLS 时，服务将以 HTTP 协议的方式对外暴露。当 Ingress 配置 TLS 时，服务将以 HTTPS 协议的方式对外暴露。Ingress 描述的服务只能以其中一种协议暴露服务，基于此规则的局限性，腾讯云容器服务 TKE 提供了混合协议的支持。

用户需要同时暴露 HTTP 及 HTTPS 服务时，只需参考本文，开启混合协议并配置所有的转发规则到 kubernetes.io/ingress.http-rules 及 kubernetes.io/ingress.https-rules 注解中即可。

规则格式

kubernetes.io/ingress.http-rules 及 kubernetes.io/ingress.https-rules 的规则格式是一个 Json Array 。每个对象的格式如下：

```
{
"host": "<domain>",
"path": "<path>",
"backend": {
"serviceName": "<service name>",
"servicePort": "<service port>"
}
}
```

混合规则配置步骤

TKE Ingress Controller 支持混合配置 HTTP 及 HTTPS 规则，步骤如下：

1. **开启混合规则**
   在 Ingress 中添加注解 kubernetes.io/ingress.rule-mix，并设置为 true。
2. **规则匹配**
   将 Ingress 中的每条转发规则与 kubernetes.io/ingress.http-rules 及 kubernetes.io/ingress.https-rules 进行匹配，并添加到对应规则集中。若 Ingress 注解中的未找到对应规则，则默认添加到 HTTPS 规则集中。
3. **校验匹配项**
   匹配时请注意校验 Host、Path、ServiceName 及 ServicePort，其中 Host 默认为 VIP、Path 默认为 /。

> ⚠ 注意：
> IPv6 的负载均衡没有 IPv4 地址，不具备提供默认域名的功能。

示例

Ingress 示例：sample-ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
annotations:
kubernetes.io/ingress.http-rules: '[{"host":"www.tencent.com","path":"/","backend":{"serviceName":"sample-service","servicePort":"80"}}]'
kubernetes.io/ingress.https-rules: '[{"host":"www.tencent.com","path":"/","backend":{"serviceName":"sample-service","servicePort":"80"}}]'
kubernetes.io/ingress.rule-mix: "true"
name: sample-ingress
```

```
namespace: default
spec:
rules:
- host: www.tencent.com
http:
paths:
- backend:
serviceName: sample-service
servicePort: 80
path: /
tls:
- secretName: tencent-com-cert
```

该示例包含以下配置：

- 描述了默认证书，证书 ID 应该存在于名为 tencent-com-cert 的 Secret 资源中。
- 开启了混合协议，并在 kubernetes.io/ingress.http-rules 及 kubernetes.io/ingress.https-rules 中都描述了 ingress.spec.rule 中描述的转发规则。
- 此时负载均衡会同时在 HTTP、HTTPS 中配置转发规则对外暴露服务。

Ingress 优雅停机

最近更新时间：2022-04-22 17:11:10

简介

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

应用场景

> ⚠ 注意：
> 仅针对 直连场景 生效，请检查您的集群是否支持直连模式。

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

操作步骤

步骤1：使用 Annotation 标明使用优雅停机

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

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

步骤2：使用 preStop 和 terminationGracePeriodSeconds

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

容器终止流程

以下为容器在 Kubernetes 环境中的终止流程：

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

具体操作步骤

1. **使用 preStop**
   要实现优雅终止，务必在业务代码里处理 SIGTERM 信号。主要逻辑是不接受新的流量进入，继续处理存量流量，所有连接全部断开才退出。了解更多可参见示例。

若您的业务代码中未处理 SIGTERM 信号，或者您无法控制使用的第三方库或系统来增加优雅终止的逻辑，也可以尝试为 Pod 配置 preStop，在其实现优雅终止的逻辑，示例如下：

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

更多关于 preStop 的配置请参见 Kubernetes API 文档。

在某些极端情况下，Pod 被删除的一小段时间内，仍然可能有新连接被转发过来，因为 kubelet 与 kube-proxy 同时 watch 到 Pod 被删除，kubelet 有可能在 kube-proxy 同步完规则前就已停止容器，这时可能导致一些新的连接被转发到正在删除的 Pod，而通常情况下，当应用收到 SIGTERM 后都不再接受新连接，只保持存量连接继续处理，因此可能导致 Pod 删除的瞬间部分请求失败。

针对上述情况，可以利用 preStop 先 sleep 短暂时间，等待 kube-proxy 完成规则同步再开始停止容器内进程。示例如下：

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

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

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

```
exec:
command:
- sleep
- 5s
```

Ingress 证书配置

最近更新时间：2022-03-16 15:26:24

操作场景

本文档介绍 Ingress 证书使用相关的内容，您可在以下场景中进行 Ingress 证书配置：

- 创建 Ingress 选用 HTTPS 监听协议时，选用合适的服务器证书能够确保访问安全。
- 为所有的 HTTPS 域名绑定同一个证书，简化配置 Ingress 下所有 HTTPS 规则的证书，使更新操作更加便捷。
- 为不同的域名绑定不同的证书，改善服务器与客户端 SSL/TLS。

注意事项

- 需提前创建需配置的证书，详情请参见 通过控制台新建服务器证书。
- 需使用 Secret 形式来设置 Ingress 证书。腾讯云容器服务 TKE Ingress 会默认创建同名 Secret，其内容包含证书 ID。
- 若您需要更换证书，建议在证书平台新建一个证书，然后更新 Secret 的证书 ID。因为集群中组件的同步会以 Secret 的声明为准，若您直接在其他证书服务、负载均衡服务上更新的证书，将会被 Secret 里的内容还原。
- Secret 证书资源需和 Ingress 资源放置在同一个 Namespace 下。
- 由于控制台默认会创建同名 Secret 证书资源，若同名 Secret 资源已存在，则 Ingress 将无法创建。
- 通常情况下，在创建 Ingress 时，不会复用 Secret 关联的证书资源。但仍支持在创建 Ingress 复用 Secret 关联的证书资源，更新 Secret 时，会同步更新所有引用该 Secret 的 Ingress 的证书。
- 为域名增加匹配证书后，将同步开启负载均衡 CLB SNI 功能（不支持关闭）。若删除证书对应的域名，则该证书将默认匹配 Ingress 所对应的 HTTPS 域名。
- 传统型负载均衡不支持基于域名和 URL 的转发，由传统型负载均衡创建的 Ingress 不支持配置多证书。

示例

TKE 支持通过 Ingress 中的 spec.tls 的字段，为 Ingress 创建的 CLB HTTPS 监听器配置证书。其中，secretName 为包含腾讯云证书 ID 的 Kubernetes Secret 资源。示例如下：

Ingress

```
spec:
tls:
- hosts:
- www.abc.com
secretName: secret-tls-2
```

Secret

- 通过 YAML 进行创建：

```
apiVersion: v1
stringData:
qcloud_cert_id: Xxxxxxxx ## 配置证书 ID 为 Xxxxxxxx
kind: Secret
metadata:
name: tencent-com-cert
namespace: default
type: Opaque
```

- 通过**容器服务控制台**进行创建：
  操作详情可参考 创建 Secret。在“新建Secret” 页面，Secret 主要参数配置如下：

- **名称**：自定义，本文以 cos-secret 为例。
- **Secret类型**：选择 **Opaque**，该类型适用于保存密钥证书和配置文件，Value 将以 Base64 格式编码。
- **生效范围**：按需选择，需确保与 Ingress 在同一 Namespace 下。
- **内容**：变量名设置为 qcloud_cert_id，变量值配置为 qcloud_cert_id 所对应的证书 ID。

Ingress 证书配置行为

- 仅配置单个 spec.secretName 且未配置 hosts 的情况下，将会为所有的 HTTPS 的转发规则配置该证书。示例如下：

```
spec:
tls:
- secretName: secret-tls
```

- 支持配置一级泛域名统配。 示例如下：

```
spec:
tls:
- hosts:
- '*.abc.com'
secretName: secret-tls
```

- 若同时配置证书与泛域名证书，将优先选择一个证书。 示例如下，www.abc.com 将会使用 secret-tls-2 中描述的证书。

```
spec:
tls:
- hosts:
- '*.abc.com'
secretName: secret-tls-1
- hosts:
- www.abc.com
secretName: secret-tls-2
```

- 对已使用多个证书的 Ingress 进行更新时，TKE Ingress controller 将进行以下行为判断：
  - HTTPS 的 rules.host 无任何匹配时，若判断不通过，则不能提交更新。
  - HTTPS 的 rules.host 匹配中单个 TLS 时，可提交更新，并为该 host 配置对 Secret 中对应的证书。
  - 修改 TLS 的 SecretName 时仅校验 SecretName 的存在性，而不校验 Secret 内容， Secret 存在即可提交更新。

> **注意：**
> 请确保 Secret 中证书 ID 符合要求。

操作步骤

通过控制台新建服务器证书

> **说明：**
> 若您已具备需配置的证书，则请跳过此步骤。

1. 登录负载均衡控制台，选择左侧导航栏中的 证书管理。
2. 在“证书管理”页面中，单击**新建**。
3. 在弹出的“新建证书”窗口中，参考以下信息进行设置。
   - **证书名称**：自定义设置。

- **证书类型**：选择“服务器证书”。**服务器证书**即 SSL 证书（SSL Certificates）。基于 SSL 证书，可将站点由 HTTP（Hypertext Transfer Protocol）切换到 HTTPS（Hyper Text Transfer Protocol over Secure Socket Layer），即基于安全套接字层（SSL）进行安全数据传输的加密版 HTTP 协议。
- **证书内容**：根据实际情况填写证书内容，证书格式要求请参见文档 SSL 证书格式要求及格式转换说明。
- **密钥内容**：仅当证书类型选择为“服务器证书”时，该选项才会显示。请参考文档 SSL 证书格式要求及格式转换说明 添加相关密钥内容。

4. 单击**提交**即可完成创建。

创建使用证书的 Ingress 对象

注意事项：

- 当控制台创建的 Ingress 开启 HTTPS 服务，会先创建同名的 Secret 资源用于存放证书 ID，并在 Ingress 中使用并监听该 Secret。
- TLS 配置域名与证书的对应关系如下：
  - 可以使用一级泛域名统配。
    - 若域名匹配中多个不同的证书，将随机选择一个证书，不建议相同域名使用不同证书。
    - 需为所有 HTTPS 域名配置证书，否则会创建不通过。

操作步骤：

参考 创建 Ingress 完成 Ingress 新建，其中监听端口勾选 **Https:443**。

修改证书

注意事项：

- 如果您需要修改证书， 请确认所有使用该证书的 Ingress。如用户的多个 Ingress 配置使用同一个 Secret 资源，那么这些 Ingress 对应 CLB 的证书会同步变更。
- 证书需要通过修改 Secret 进行修改， Secret 内容中包含您使用的腾讯云证书的 ID。

操作步骤：

1. 执行以下命令，使用默认编辑器打开需修改的 Secret。其中，[secret-name] 需更换为需修改的 Secret 的名称。

```
kubectl edit secrets [secret-name]
```

2. 修改 Secret 资源，将 qcloud_cert_id 的值修改为新的证书 ID。
   与创建 Secret 相同，修改 Secret 证书 ID 需要进行 Base64 编码，请根据实际需求选择 Base64 手动编码或者指定 stringData 进行 Base64 自动编码。

更新 Ingress 对象

通过控制台更新

1. 登录 腾讯云容器服务控制台，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面，选择需修改 Ingress 的集群 ID。
3. 在集群详情页，选择左侧**服务与路由** > **Ingress**。如下图所示：

4. 单击目标 Ingress 所在行右侧的**更新转发配置**。

5. 在“更新转发配置”页面中，根据实际情况进行转发配置规则更新。
6. 单击**更新转发配置**即可完成更新操作。

通过 yaml 更新

执行以下命令，使用默认编辑器打开需修改的 ingress，修改 yaml 文件并保存即可完成更新操作。

```
kubectl edit ingress <ingressname> -n <namespaces>
```

Ingress Annotation 说明

最近更新时间：2022-03-25 10:59:57

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

注解使用方式

```
apiVersion:
kind: Ingress
metadata:
annotations:
kubernetes.io/ingress.class: "qcloud"
name: test
.....
```

Annotation 集合

kubernetes.io/ingress.class

说明：

配置 Ingress 类型。当前组件管理未配置该注解，或注解内容为 qcloud 的 Ingress 资源。

使用示例：

```
kubernetes.io/ingress.class: "qcloud"
```

kubernetes.io/ingress.qcloud-loadbalance-id

说明：

只读注解，组件提供当前 Ingress 引用的负载均衡 LoadBalanceId。

使用示例：

```
kubernetes.io/ingress.qcloud-loadbalance-id: "lb-3imskkfe"
```

ingress.cloud.tencent.com/loadbalance-nat-ipv6

说明：

只读注解，当用户配置或申请的为 NAT IPv6负载均衡时，提供 IPv6地址。

ingress.cloud.tencent.com/loadbalance-ipv6

说明：

只读注解，当用户配置或申请的为 FullStack IPv6负载均衡时，提供 IPv6地址。

kubernetes.io/ingress.internetChargeType

说明：

负载均衡的付费类型，当前仅在创建时支持配置，创建后不支持修改付费类型，创建后修改本注解无效。

指定创建负载均衡时，负载均衡的付费类型。请配合 kubernetes.io/ingress.internetMaxBandwidthOut 注解一起使用。

可选值：

- TRAFFIC_POSTPAID_BY_HOUR 按流量按小时后计费。
- BANDWIDTH_POSTPAID_BY_HOUR 按带宽按小时后计费。

使用示例：

kubernetes.io/ingress.internetChargeType: "TRAFFIC_POSTPAID_BY_HOUR"

kubernetes.io/ingress.internetMaxBandwidthOut

说明：

CLB 带宽设置，当前仅在创建时支持配置，创建后不支持修改带宽，创建后修改本注解无效。

指定创建负载均衡时，负载均衡的最大出带宽，仅对公网属性的 LB 生效。需配合 kubernetes.io/ingress.internetChargeType 注解一起使用。

可选值：

范围支持1到2048，单位 Mbps。

使用示例：

kubernetes.io/ingress.internetMaxBandwidthOut: "2048"

kubernetes.io/ingress.extensiveParameters

说明：

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

参考 创建负载均衡实例 为创建负载均衡追加自定义参数。

使用示例：

- 创建 NAT64 IPv6 实例：
  kubernetes.io/ingress.extensiveParameters: '{"AddressIPVersion":"IPV6"}'
- 创建 IPv6 实例：
  kubernetes.io/ingress.extensiveParameters: '{"AddressIPVersion":"IPv6FullChain"}'
- 购买电信负载均衡：
  kubernetes.io/ingress.extensiveParameters: '{"VipIsp":"CTCC"}'
- 指定可用区创建：
  kubernetes.io/ingress.extensiveParameters: '{"ZoneId":"ap-guangzhou-1"}'

kubernetes.io/ingress.subnetId

说明：

指定创建内网类型的负载均衡，并指定负载均衡所属子网。

使用示例：

kubernetes.io/ingress.subnetId: "subnet-3swgntkk"

kubernetes.io/ingress.existLbId

说明：

指定使用已有负载均衡作为接入层入口资源。

> ⚠ 注意：
> 使用已有负载均衡时，需要保证其不包含其他监听器。

使用示例：

kubernetes.io/ingress.existLbId: "lb-342wppll"

kubernetes.io/ingress.rule-mix:

kubernetes.io/ingress.http-rules:

kubernetes.io/ingress.https-rules:

说明：

支持配置混合协议，支持转发路径同时在 HTTP 和 HTTPS 上进行转发。支持手动配置重定向规则。

使用示例：

使用方式详情见 Ingress 混合使用 HTTP 及 HTTPS 协议。

ingress.cloud.tencent.com/direct-access

说明：

支持七层直连用户负载均衡。需要注意在各种不同的网络下，直连接入的服务依赖。

使用示例：

使用方式详情见 使用 LoadBalancer 直连 Pod 模式 Service。

ingress.cloud.tencent.com/tke-service-config

说明：

通过 tke-service-config 配置负载均衡相关配置，包括监听器、转发规则等。

使用示例：

ingress.cloud.tencent.com/tke-service-config: "nginx-config"，详情可参见 Ingress 使用 TkeServiceConfig 配置 CLB。

ingress.cloud.tencent.com/tke-service-config-auto

说明：

通过该注解可自动创建 TkeServiceConfig 资源，并提供配置的模板，用户可以按需进行配置。

使用示例：

ingress.cloud.tencent.com/tke-service-config-auto: "true"，详情可参见 Ingress 使用 TkeServiceConfig 配置 CLB。

ingress.cloud.tencent.com/rewrite-support

说明：

- 可以配合 kubernetes.io/ingress.http-rules、kubernetes.io/ingress.https-rules 实现手动重定向能力。
- 可以配合 ingress.cloud.tencent.com/auto-rewrite 实现自动重定向能力。

使用示例：

ingress.cloud.tencent.com/rewrite-support: "true"

ingress.cloud.tencent.com/auto-rewrite

说明：

为 HTTP 端口提供自动重定向能力，所有在 HTTPS 端口声明的转发规则都会创建对应的重定向规则。需要配合 ingress.cloud.tencent.com/rewrite-support 注解开启重定向的管理能力。

使用示例：

ingress.cloud.tencent.com/auto-rewrite: "true"

ingress.cloud.tencent.com/cross-region-id

说明：

Ingress 跨域绑定功能，指定需要从哪个地域接入。需要和 kubernetes.io/ingress.existLbId 或 ingress.cloud.tencent.com/cross-vpc-id 配合使用。

使用示例：

- **创建异地接入的负载均衡：**
  ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"
  ingress.cloud.tencent.com/cross-vpc-id: "vpc-646vhcjj"
- **选择已有负载均衡进行异地接入：**
  ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"
  kubernetes.io/ingress.existLbId: "lb-342wppll"

ingress.cloud.tencent.com/cross-vpc-id

说明：

Ingress 跨域绑定功能，指定需要接入的 VPC。可以和 ingress.cloud.tencent.com/cross-region-id 注解配合指定其他地域 VPC。

> **注意：**
> 适用于 TKE 创建并管理的负载均衡，对使用已有负载均衡的场景该注解无效。

使用示例：

创建异地接入的负载均衡：

ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"
ingress.cloud.tencent.com/cross-vpc-id: "vpc-646vhcjj"

ingress.cloud.tencent.com/enable-grace-shutdown

说明：

支持 CLB 直连模式的优雅停机。

使用示例：

仅在直连模式下支持，需要配合使用 ingress.cloud.tencent.com/direct-access ，使用方式详情见 Ingress 优雅停机。

API 网关类型 Ingress
API 网关 TKE 通道配置

最近更新时间：2022-06-09 14:53:00

操作场景

您可以通过 API 网关直接接入TKE 集群的 Pod，不需要经过 CLB。本文档指导您通过控制台创建 TKE 通道，并在 API 的后端中，配置后端类型为 TKE 通道，让 API 网关的请求，直接转到 TKE 通道的对应的 Pod 上。

功能优势

- API 网关直接连接 TKE 集群的 Pod，减少中间节点（例如 CLB）。
- 一个 TKE 通道可以同时对接多个 TKE 集群。

> 说明：
> 目前仅在**专享**类型的 API 网关上支持 TKE 通道。

前提条件

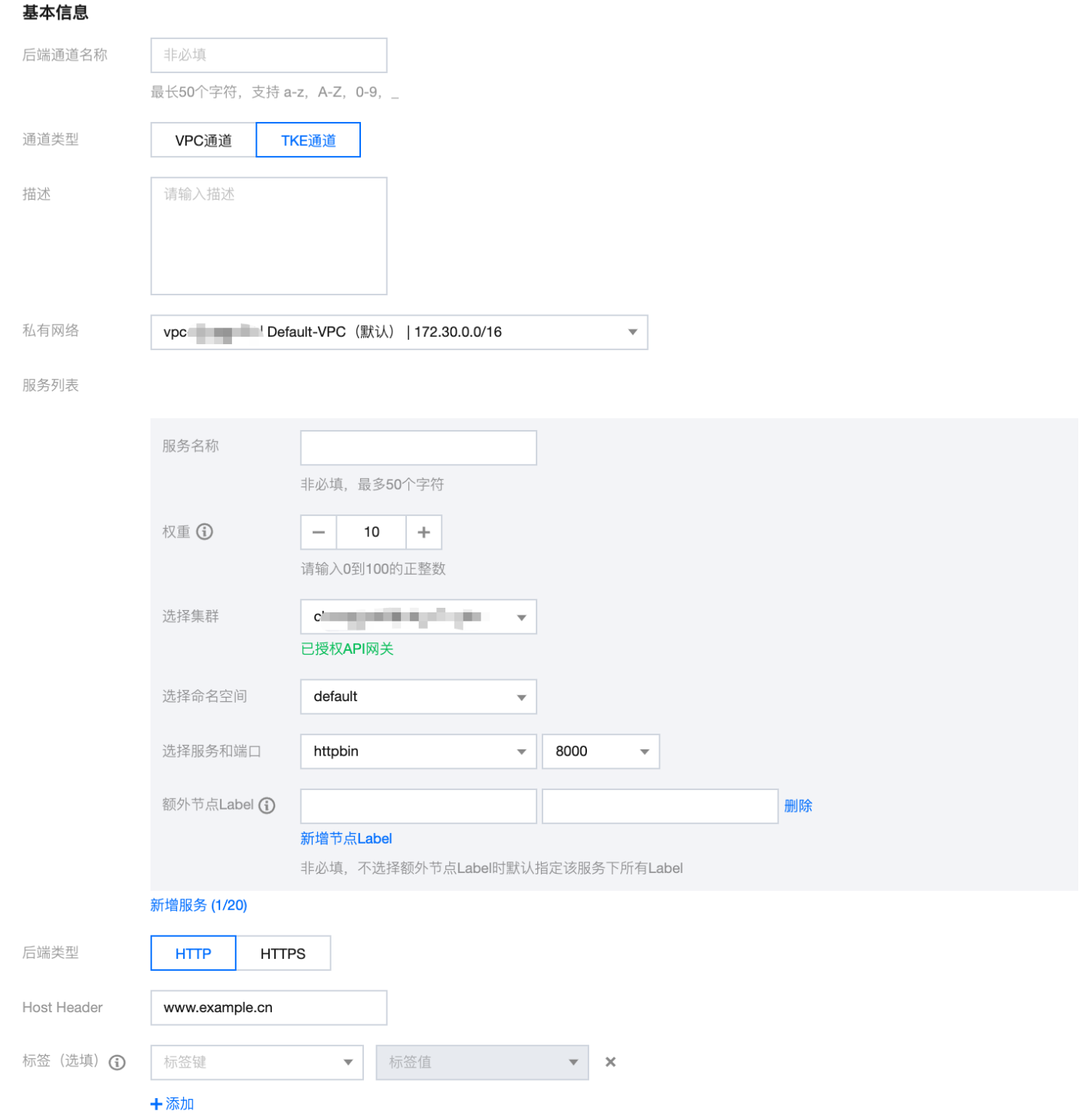
1. 已有专享型的服务。
2. 已有容器服务 TKE 的集群，并且已获取集群 admin 角色。

操作步骤

步骤1：创建 TKE 通道

1. 登录 API 网关控制台。
2. 在左侧导航栏选择**后端通道**，单击**新建**。
3. 在新建后端通道页面填写以下信息：
   - 后端通道名称：输入后端通道名称
   - 通道类型：选择**TKE通道**
   - 私有网络：选择私有网络 VPC
   - 服务列表：服务列表中配置多个服务，服务数量上限为20个，多个Pod之间采用加权轮训算法分配流量。单个服务配置的步骤如下：
     a. 填写服务的每个 Pod 的权重占比。
     b. 选择集群，如果集群还没授权，API 网关会请求授权。
     c. 选择集群内命名空间。
     d. 选择服务和服务的端口。
     e. 高级可选项：额外节点 Label。
   - 后端类型：选择 HTTP 或者 HTTPS。
   - Host Header：可选项，Host Header 是 API 网关访问后端服务时候，HTTP/HTTPS 请求中，携带的请求 HEADER 中 Host 的值。
   - 标签：可选项，标签用于从不同维度对资源分类管理。

一个完整的 TKE 通道配置如下：

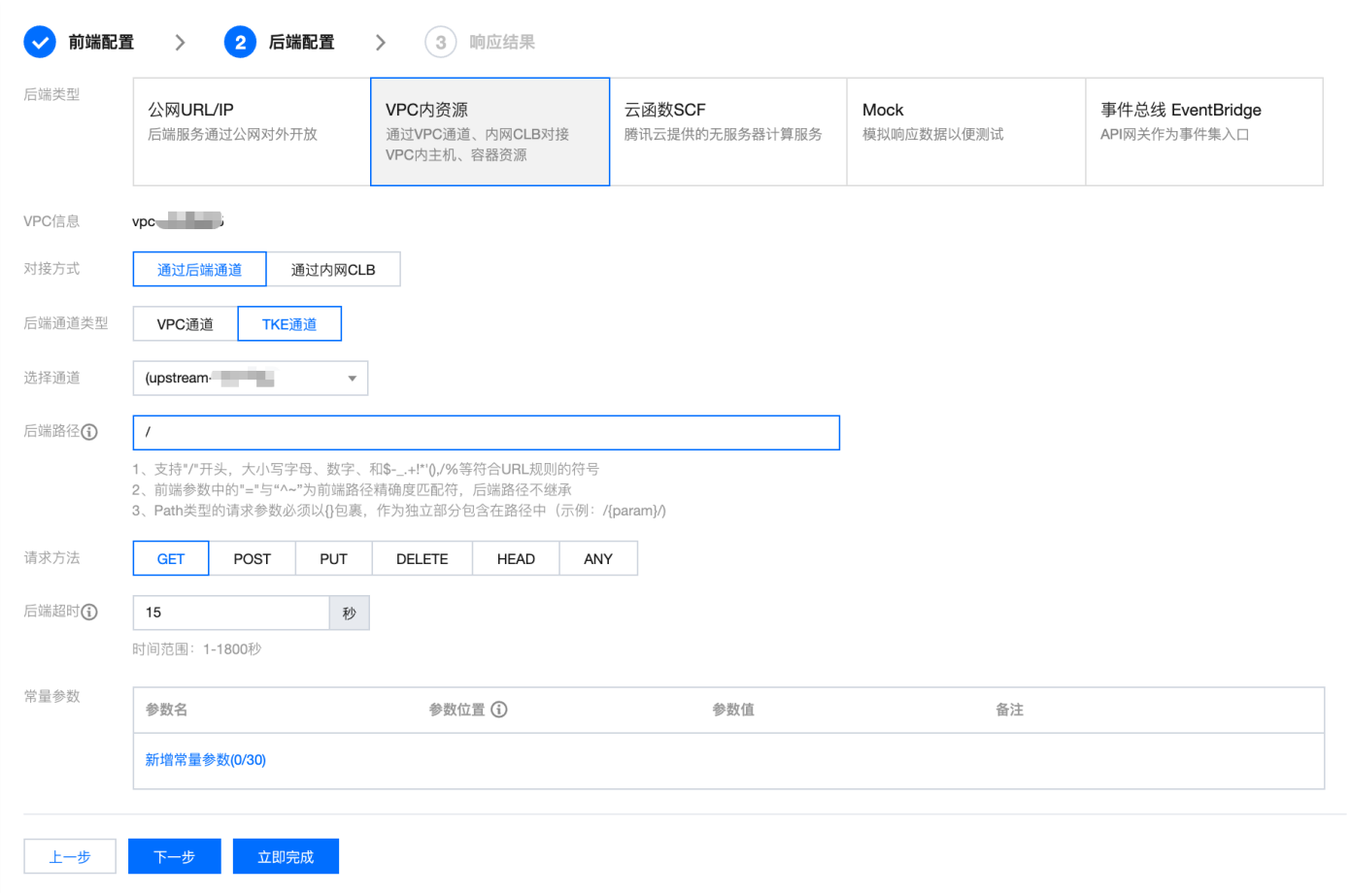


步骤2：API 后端对接 TKE 通道

1. 在 API 网关控制台的 服务页面，单击目标服务的“ID”，进入管理 API 页面。
2. 单击**新建**，创建通用 API。

3. 输入前端配置，然后单击**下一步**。

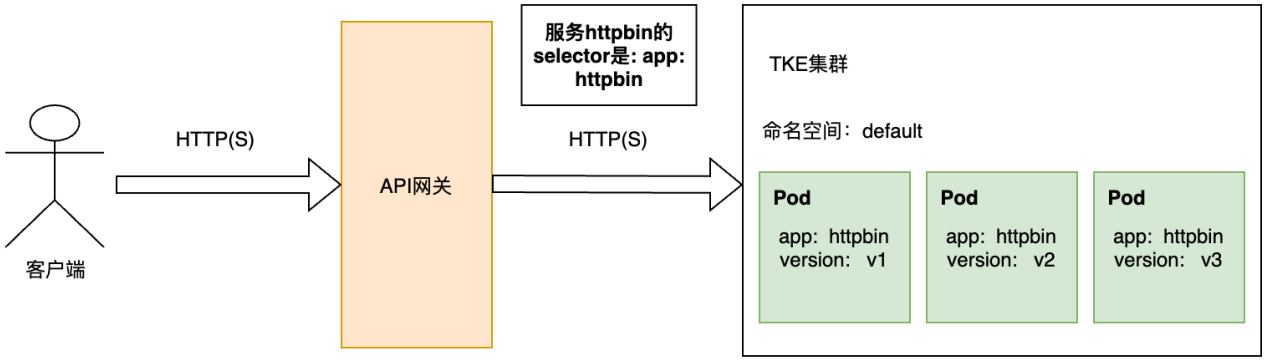
4. 选择后端类型为 **VPC内资源**，并且选择后端通道类型为 **TKE通道**，单击**下一步**。



5. 设置响应结果，并单击**完成**。

网络架构

TKE 通道被 API 绑定后，整个网络的架构如下：



API 网关直接访问 TKE 集群中的 Pod，不需要经过 CLB。因为在 TKE 集群中，httpbin 的服务配置文件 YAML 如下，其中 selector 中，表示选择带有标签键 app，标签值为 httpbin 的 Pod 作为 TKE 通道的节点。因此，version 为 v1/v2/v3 的 Pod 也都是 TKE 通道的节点。

```
apiVersion: v1
kind: Service
metadata:
name: httpbin
```

```
labels:
app: httpbin
service: httpbin
spec:
ports:
- name: http
port: 8000
targetPort: 80
selector:
app: httpbin
```

注意事项

- 一个 TKE 通道最多只能对接20个 TKE 服务。
- 用户需要拥有 TKE 集群的 admin 角色。
- TKE 通道和 API 网关专享在同一个 VPC 下才能使用，目前 API 网关暂时不支持直接跨 VPC。

API 网关获取 TKE 集群授权

最近更新时间：2022-06-09 14:53:12

操作场景

本文档会指导您如何授权 API 网关访问 TKE 集群的 API Server，并提供授权相关问题解决方案。最后通过 YAML文件描述 API 网关获取的权限列表。

前提条件

1. 已登录 API 网关控制台。
2. 已有容器服务 TKE 的集群，并且已获取集群 admin 角色。

操作步骤

- 在 API 网关的 TKE 通道配置中，如果是首次引用某个 TKE 集群，需授予 API 网关访问该 TKE 集群 API Server 的权限，并且需要保证 TKE集群已经开启了内网访问。

  授权操作，是在配置 TKE 通道时候，系统会自动识别集群是否已经授权，如果没有授权，API 网关会提示用户授权。

- 如果集群已经授权API网关访问，则会显示**已授权API网关**。每个集群只需要在 API 网关授权一次，后面使用不需要重复授权。

原理说明

API 网关获取用户授权的流程如下：

1. 在命名空间 kube-system 下，通过创建名为 apigw-ingress 的 ServiceAccount 和名为 apigw-ingress-clusterrole 的 ClusterRole。
2. 把 apigw-ingress 和 apigw-ingress-clusterrole 通过 ClusterRoleBinding 绑定在一起。接着 apigw-ingress 这个 ServiceAccount 的权限就被 API 网关获取到，用来访问集群的 APIServer。

   其中名为 apigw-ingress 的 ServiceAccount 权限，是保存在以 apigw-ingress-token- 为前缀的 Secret 中。

如果您想了解 API 网关获取的权限明细和具体方式，可以查看我们创建相关资源的 YAML 文件。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: apigw-ingress-clusterrole
rules:
- apiGroups:
- ""
resources:
- services
- namespaces
- endpoints
- nodes
- pods
verbs:
- get
- list
- watch
- apiGroups:
- apps
```

```
resources:
- deployments
- replicasets
verbs:
- get
- list
- watch
- apiGroups:
- ""
resources:
- configmaps
- secrets
verbs:
- "*"
- apiGroups:
- extensions
resources:
- ingresses
- ingresses/status
verbs:
- "*"
- apiGroups:
- ""
resources:
- events
verbs:
- create
- patch
- list
- update
- apiGroups:
- apiextensions.k8s.io
resources:
- customresourcedefinitions
verbs:
- "*"
- apiGroups:
- cloud.tencent.com
resources:
- tkeserviceconfigs
verbs:
- "*"
---
apiVersion: v1
kind: ServiceAccount
metadata:
namespace: kube-system
name: apigw-ingress
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
name: apigw-ingress-clusterrole-binding
roleRef:
```

```
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
name: apigw-ingress-clusterrole
subjects:
- kind: ServiceAccount
name: apigw-ingress
namespace: kube-system
```

注意事项

用户在成功授权 API 网关 TKE 集群的访问权限后，就不能修改 API 网关保留使用的资源，资源列表如下：

- kube-system 命名空间下，名为 apigw-ingress 的 ServiceAccount。
- kube-system 命名空间下，名为 apigw-ingress-clusterrole 的 ClusterRole。
- kube-system 命名空间下，名为 apigw-ingress-clusterrole-binding 的 ClusterRoleBinding。
- kube-system 命名空间下，以 apigw-ingress-token- 为前缀的 Secret。

常见问题

问题描述：授权时发现，TKE 集群没有开启内网访问功能。

解决方法：主动 开启 TKE 集群内网访问功能，然后单击重试。

额外节点 Label 的使用

最近更新时间：2021-12-20 16:43:30

使用场景

通过额外节点 Label 的使用，您可以直接将请求转发到某个服务下的具有指定 Label 的 Pod，精细需要控制转发的 Pod。

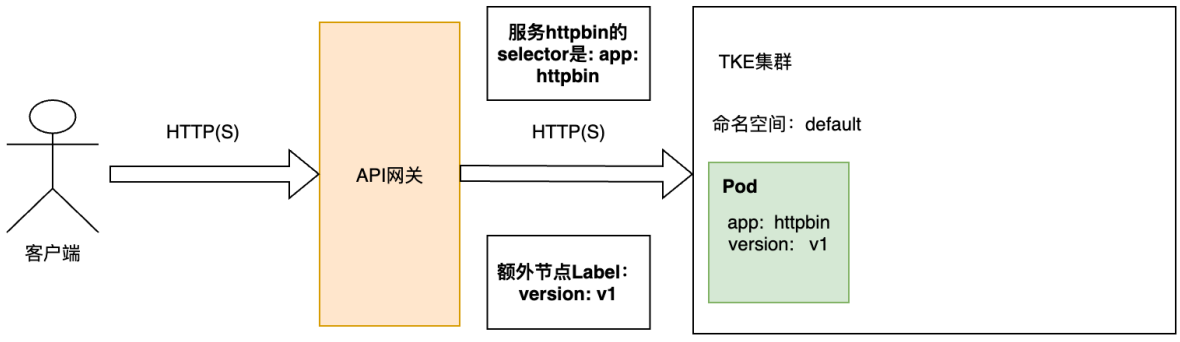
例如：default 命名空间下，存在 Label 为 app: httpbin 和 version: v1 的 Pod， 也存在 app: httpbin 和 version: v2 的 Pod，存在一个 httpbin 服务（selector 选择的是 app:httpbin）。如果希望 API 网关只转发到 Label 为 app: httpbin 和 version: v1 的 Pod， 可以通过额外节点 Label，加上 version: v1的配置，就可以实现。

操作步骤

1. 在 配置 TKE 通道 的服务前提下，再手动输入额外节点 Label。效果如下：

2. 单击**保存**，新建或修改 TKE 通道。

最终转发的效果如下：



原理说明

TKE 集群中，服务本身是有 selector 的配置。例如：httpbin 服务中，selector 的配置是 app: httpbin，但是 API 网关提供的额外节点Label 会与 httpbin 服务中的 selector 合并起来，组合成的 Label 是：app: httpbin 和 version: v1。因此，改 TKE 通道节点，只会出现 version: v1的 http 的 Pod。

如果在额外节点 Label 中输入在 httpbin 服务中已经存在的 Label 的键，那么额外节点中输入的该 Label 会被忽略，以 selector 中存在的 Label 的值为准。例如：额外 Label 中输入 app: not-httpbin，这个 Label 与服务 httpbin 的 selector 发生了冲突，app: not-httpbin 将会被忽略。

httpbin 服务的 YAML 如下：

```
apiVersion: v1
kind: Service
metadata:
name: httpbin
labels:
app: httpbin
service: httpbin
spec:
ports:
- name: http
port: 8000
targetPort: 80
selector:
app: httpbin
```

注意事项

- 额外节点 Label 是高级功能，需要用户输入的时候确认 Label 的存在。如果输入错误的 Label，会导致 TKE 通道的节点数量变为0.
- 如果服务的 selector 和额外节点 Label 出现同一个键的时候，会以 selector 中的配置为准。
- 如果服务的端口（port）发生更改（例如从80改为8080），需要在 API 网关中同步修改；如果端口（port）没有修改，仅仅修改了目标端口（target port），API 网关会自动同步，不需要在 API 网关修改。

Nginx 类型 Ingress
概述

最近更新时间：2022-04-02 17:40:46

> 说明：
> 如果您需要安装 NginxIngress 组件，可通过 提交工单 来联系我们。

Nginx-ingress 介绍

Nginx 可以用作反向代理、负载平衡器和 HTTP 缓存。

Nginx-ingress 是使用 Nginx 作为反向代理和负载平衡器的 Kubernetes 的 Ingress 控制器。您可以部署 Nginx-ingress 组件，在集群中使用 Nginx-ingress。容器服务 TKE 提供了产品化的能力，帮助您在集群内安装和使用 Nginx-ingress。

Nginx-ingress 名词解释

- **Nginx-ingress 组件**：在 TKE 中使用 Nginx-ingress 的入口，您可以在集群的组件页面一键安装部署 Nginx-ingress。
- **Nginx-ingress 实例**：一个集群中可部署多个 Nginx-ingress（例如一个用于公网，一个用于内网）。在 Kubernetes 中对应一个 CRD，创建一个 Nginx-ingress 实例会在集群中自动创建 Nginx-ingress-controller、service、configmap 等 Kubernetes 资源。
- **Nginx-ingress-controller**：实际 Nginx 负载，同时 controller 会 watch kubernetes ingress 对象的变化更新在集群中，Nginx 负载的转发配置即 nginx.conf 文件。

Nginx-ingress 相关操作

Nginx-Ingress 相关操作及功能如下，您可参考以下文档进一步了解：

- Nginx-ingress 安装
- 使用 Nginx-ingress 对象接入集群外部流量
- Nginx-ingress 监控配置
- Nginx-ingress 日志配置

安装 Nginx-ingress 实例

最近更新时间：2022-01-10 11:03:58

安装 NginxIngress 组件

> 说明：
> 如果您需要安装 NginxIngress 组件，可通过 提交工单 来联系我们。

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 NginxIngress。
5. 单击**完成**即可安装组件。

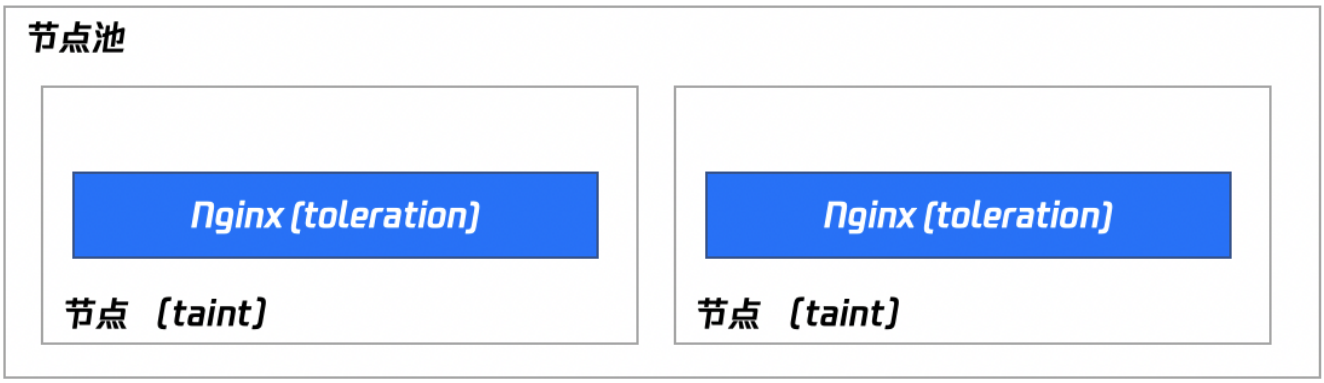
安装方式

您可以根据不同的业务场景需求，使用以下几种安装方案在容器服务 TKE 中安装 Nginx-ingress。

- 通过 DaemonSet 形式在指定节点池部署
- 通过 Deployment + HPA 形式并指定调度规则部署
- Nginx 前端接入 LB 部署方式

通过 DaemonSet 形式在指定节点池部署（推荐）

Nginx 作为关键的流量接入网关，不建议您将 Nginx 与其他业务部署在相同的节点内。推荐您使用指定的节点池来部署 Nginx-ingress。部署架构如下图所示：



请参考以下步骤进行安装：

> 说明：
> 使用此安装方式，您可以完整享有节点池快速扩缩容的能力，后续您只要调整节点池的数量，即可扩缩容 Nginx 的副本。

1. 准备用于部署 Nginx-ingress 的节点池，同时设置污点 taint（防止其他 Pod 调度到该节点池）。部署节点池详情可参见 节点池相关说明。
2. 在集群中 安装 NginxIngress 组件。

3. 在新建的 Nginx Ingress 组件详情页，单击**新增Nginx Ingress实例**（一个集群内可以同时存在多个 Nginx）。

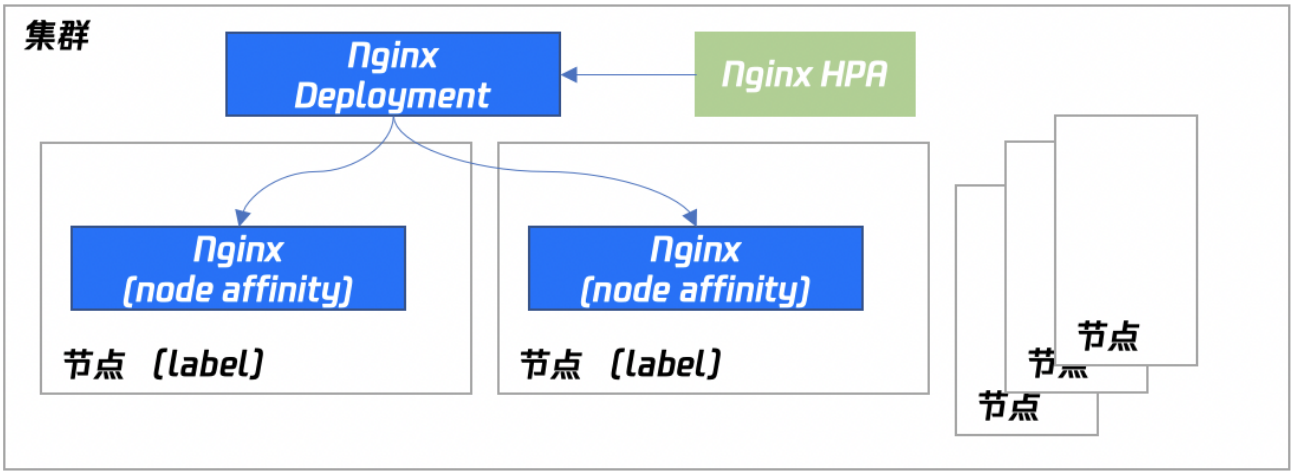
4. 在弹出的窗口中，选择部署选项中的**指定DaemonSet节点池部署**，并按需设置其他参数。如下图所示：

- 节点池：配置节点池。
- Nginx 配置：Requst 需设置比节点池的机型配置小（节点本身有资源预留）。Limit 可不设置。

5. 单击**确定**即可完成安装。

通过 Deployment + HPA 形式并指定调度规则部署

使用 Deployment + HPA 的形式部署 Nginx-ingress，您可以根据业务需要配置污点和容忍将 Nginx 和业务 Pod 分散部署。同时搭配 HPA，可设置 Nginx 根据 CPU / 内存等指标进行弹性伸缩。部署架构如下图所示：



安装步骤

1. 在集群中设置即将部署 Nginx 的节点的 Label，设置步骤可参见 设置节点 Label。
2. 在集群中 安装 NginxIngress 组件。
3. 在新建的 Nginx Ingress 组件详情页，单击**新增Nginx Ingress实例**（一个集群内可以同时存在多个 Nginx）。

4. 在弹出的窗口中，选择部署选项中的**自定义Deployment+HPA 部署**，并按需设置其他参数。如下图所示：

- **节点调度策略**：需自行指定。
- Nginx 配置：Requst 需设置比节点池的机型配置小（节点本身有资源预留）。Limit 可不设置。

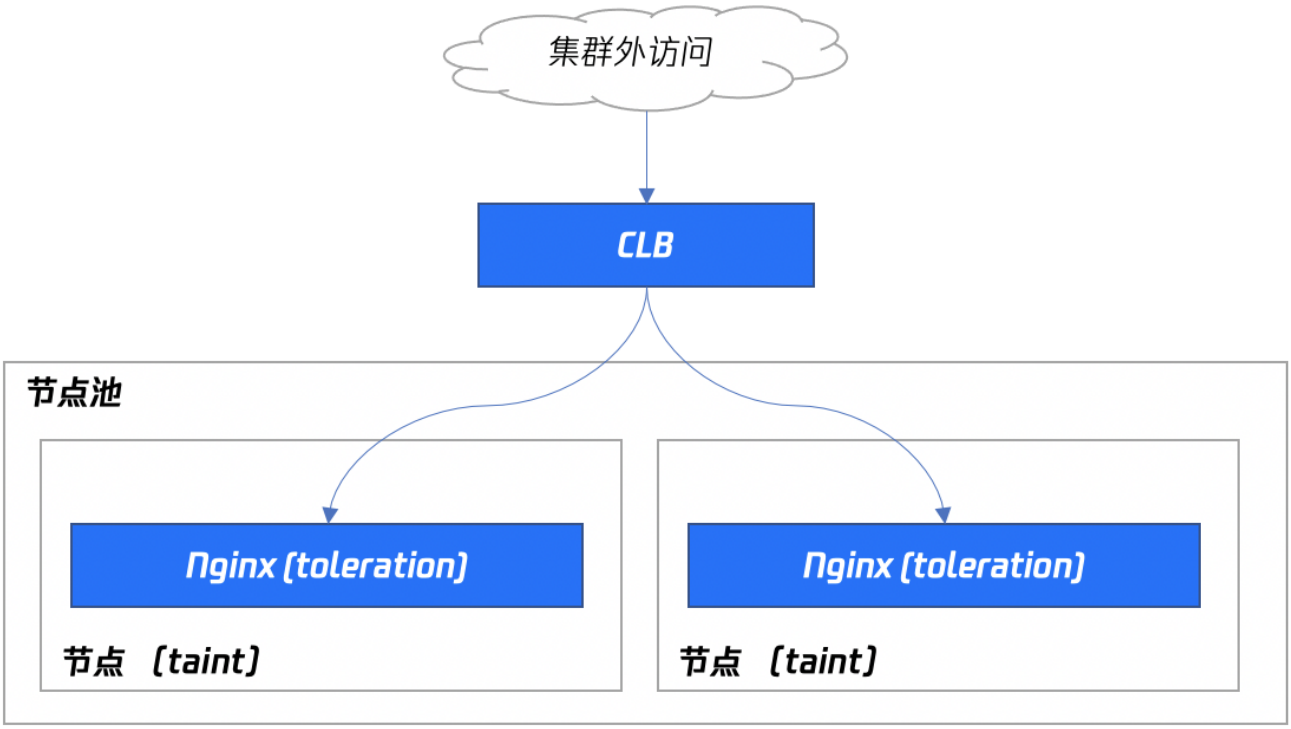
5. 单击**确定**即可完成安装。

Nginx 前端接入 LB 部署方式

仅部署 Nginx 在集群内将无法接收外部流量，还需配置 Nginx 的前端 LB。TKE 现已提供产品化的安装能力，您也可以根据业务需要选择不同的部署模式。

VPC-CNI 模式集群使用 CLB 直通 Nginx 的 Serivce（推荐）

如果您的集群是 VPC-CNI 模式的集群，推荐您使用 CLB 直通 Nginx 的 Serivce。下图为以节点池部署的负载示例。



当前方案性能好，而且不需要手动维护 CLB，是最理想的方案。需要集群支持 VPC-CNI，如果您的集群已配置 VPC-CNI 网络插件，或者已配置 Global Router 网络插件并开启了 VPC-CNI 的支持（两种模式混用），建议使用此方案。

Globalrouter 模式集群使用普通 Loadbalancer 模式的 Service

如果您的集群不支持 VPC-CNI 模式网络，您也可以通过常规的 Loadbalancer 模式 Service 接入流量。
当前 TKE 上 LoadBalancer 类型的 Service 默认实现是基于 NodePort，CLB 会绑定各节点的 NodePort 作为后端 RS，将流量转发到节点的 NodePort，然后节点上再通过 iptables 或 ipvs 将请求路由到 Service 对应的后端 Pod。这种方案是最简单的方案，但流量会经过一层 NodePort，会多一层转发。可能存在以下问题：

- 转发路径较长，流量到了 NodePort 还会再经过 k8s 内部负载均衡，通过 iptables 或 ipvs 转发到 Nginx，会增加一点网络耗时。
- 经过 NodePort 必然发生 SNAT，如果流量过于集中容易导致源端口耗尽或者 conntrack 插入冲突导致丢包，引发部分流量异常。
- 每个节点的 NodePort 也充当一个负载均衡器，CLB 如果绑定大量节点的 NodePort，负载均衡的状态会分散在每个节点上，容器导致全局负载不均。
- CLB 会对 NodePort 进行健康探测，探测包最终会被转发到 nginx ingress 的 Pod，如果 CLB 绑定的节点多，Nginx-ingress 的 Pod 少，会导致探测包对 Nginx-ingress 造成较大的压力。

使用 HostNetwork + LB 模式

控制台暂不支持，您可以手动修改 Nginx 工作负载的 Yaml 配置网络模式为 HostNetwork，手动创建 CLB 绑定 Nginx 暴露的节点端口。
需要注意使用 hostNetwork 时，为避免端口监听冲突，Nginx-ingress 的 Pod 不能被调度到同一节点。

TKE 安装 Nginx-ingress 默认参数

Nginx-ingress 参数设置方式

您可以在 Nginx-ingress 组件详情页，Ningx 参数 tab 中选择的 Nginx-ingress 实例进行 YAML 编辑。

> ⚠ 注意：
> 默认情况下配置参数不会重启 Nginx，生效时间有细微延迟。

1. 登录 容器服务控制台，选择左侧导航栏中的**集群**。

2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 单击需要设置参数的组件右侧的**更新Nginx配置**，进入“Nginx配置”页面。
5. 选择 Nginx Ingress 实例，并单击**编辑YAML**。
6. 在“更新ConfigMap”页面进行编辑，单击**完成**即可配置参数。

配置参数示例

```
apiVersion: v1
kind: ConfigMap
metadata:
name: alpha-ingress-nginx-controller
namespace: kube-system
data:
access-log-path: /var/log/nginx/nginx_access.log
error-log-path: /var/log/nginx/nginx_error.log
log-format-upstream: $remote_addr - $remote_user [$time_iso8601] $msec "$request" $status $body_bytes_sent "$http_referer" "$http_user_agent" $request_length $request_time [$proxy_upstream_name] [$proxy_alternative_upstream_name] [$upstream_addr] [$upstream_response_length] [$upstream_response_time] [$upstream_status] $req_id
keep-alive-requests: "10000"
max-worker-connections: "65536"
upstream-keepalive-connections: "200"
```

⚠ 注意：

- **请勿修改** access-log-path 、error-log-path、log-format-upstream。**若修改则会对 CLS 日志采集造成影响。**
- **若您需要根据业务配置不同的参数，可参见** 官方文档。

使用 Nginx-ingress 对象接入集群外部流量

最近更新时间：2022-04-18 11:43:48

前提条件

- 已登录 容器服务控制台 。
- 集群内已 部署 NginxIngress 组件。
- 已安装并创建业务需要的 Nginx-ingress 实例。

使用方法

Nginx-ingress 控制台操作指引

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，单击**集群**，进入集群管理页面。
3. 单击已安装 Nginx-ingress 组件的集群 ID，进入集群管理页面。
4. 选择**服务与路由** > **Ingress**，进入 Ingress 信息页面。
5. 单击**新建**，进入“新建Ingress”页面。
6. 根据实际需求，设置 Ingress 参数。如下图所示：

   - Ingress类型：选择**IngressController**。
   - 转发规则：需自行设置。
   - Annotation：设置注解，可配置的注解可参见 为 Nginx 类型 Ingress 对象配置注解。
7. 单击**创建Ingress**即可。

Kubectl 操作 Nginx-ingress 指引

在 Kubernetes 中引入 IngressClass 资源和 ingressClassName 字段之前，Ingress 类由 Ingress 中的 kubernetes.io/ingress.class 注解指定。示例如下：

```
metadata:
name:
annotations:
kubernetes.io/ingress.class: "nginx-pulic". ## 对应 TKE 集群 Nginx-ingress 组件中的 Nginx-ingress 实例名称
```

相关操作

为 Nginx 类型 Ingress 对象可配置注解，详情可参见 官方文档。

Nginx-ingress 对象使用模型

当多个 Ingress 对象作用于一个 Nginx 实体时：

- 按 CreationTimestamp 字段对 Ingress 规则排序，即先按旧规则。
- 如果在多个 Ingress 中为同一主机定义了相同路径，则最早的规则将获胜。
- 如果多个 Ingress 包含同一主机的 TLS 部分，则最早的规则将获胜。
- 如果多个 Ingress 定义了一个影响 Server 块配置的注释，则最早的规则将获胜。
- 按每个 hostname 创建 NGINX Server。
- 如果多个 Ingress 为同一 host 定义了不同的路径，则 ingress-controller 合并这些定义。
- 多个 Ingress 可以定义不同的注释。这些定义在 Ingress 之间不共享。
- Ingress 的注释将应用于 Ingress 中的所有路径。

触发更新 nginx.conf 机制

以下内容描述了需要重新加载 nginx.conf 的情况：

- 创建新的 ingress 对象。
- 为 Ingress 添加新的 TLS。
- Ingress 注解的更改不仅影响上游配置，而且影响更大。例如 load-balance 注释不需要重新加载。
- 为 Ingress 添加/删除路径。
- 删除 Ingress、Ingress 的 Service、Secret。
- Ingress 关联的对象状态不可知，例如 Service 或 Secret。
- 更新 Secret。

Nginx-ingress 日志配置

最近更新时间：2022-04-18 11:44:30

容器服务 TKE 通过集成日志服务 CLS，提供了全套完整的产品化能力，实现 Nginx-ingress 日志采集、消费能力。

Nginx-ingress 日志基础

Nginx Controller 需要搜集以下日志并提供给用户：

- **Nginx Controller 日志**：重要。控制面日志，记录了 Nginx Controller 控制面的修改。主要用于控制面排障，例如用户错误配置 Ingress 模板导致同步未进行等。
- **AccessLog 日志**：重要。用户数据面日志，记录了用户的七层请求相关信息。主要用于提供给用户进行数据分析、审计、业务排障等。
- **ErrorLog 日志**：一般。Nginx 的内部错误日志。

默认配置下，AccessLog 和 Nginx Controller 日志会混合到标准输出流，日志采集将遇到困难。本文向您介绍如何对日志路径进行区分后分别收集日志。

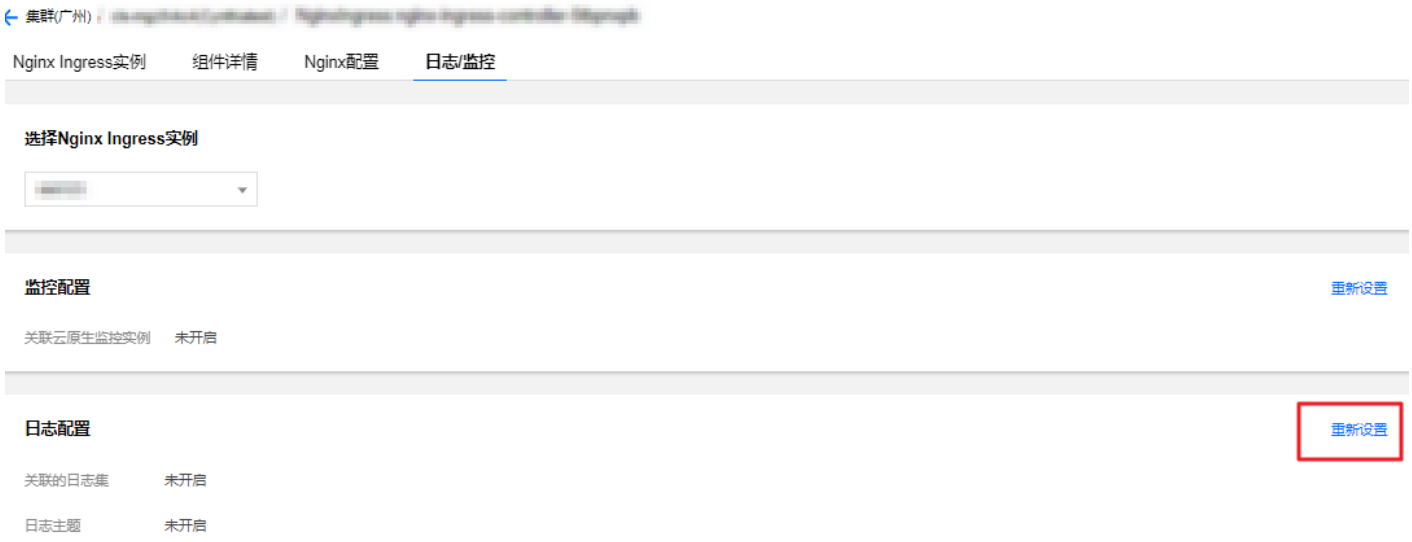
前提条件

已在容器服务控制台 的 **功能管理**中开启日志采集，详情参见 开启日志采集。

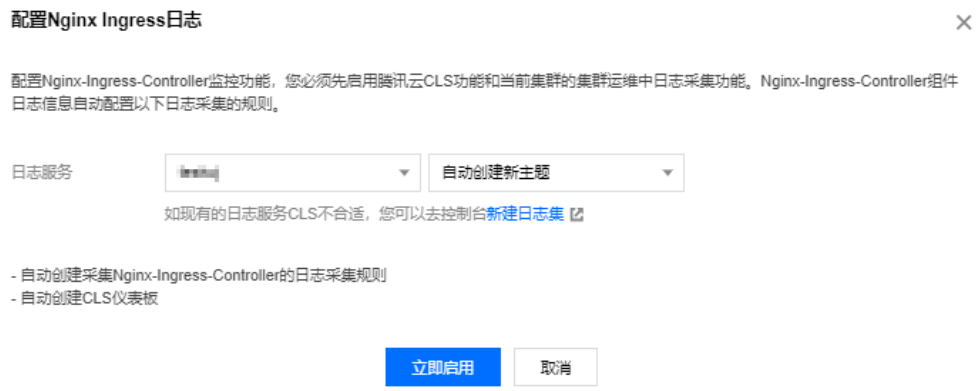
TKE Nginx-ingress 采集日志

采集日志步骤

1. 为目标集群 安装 Nginx-ingress 组件。
2. 在“组件管理”页面选择已安装的组件名称，进入组件详情页。
3. 在**日志监控**页面中，选择日志配置右侧的**重新设置**。如下图所示：



4. 在弹出的窗口中选择指定的日志集，如不制定将创建新的日志集。如下图所示：



5. 单击**立即启用**即可完成日志采集配置。

> **注意：**
> 日志服务具体计费规则和收费标准请参见 CLS 计费概述 。

采集日志指标

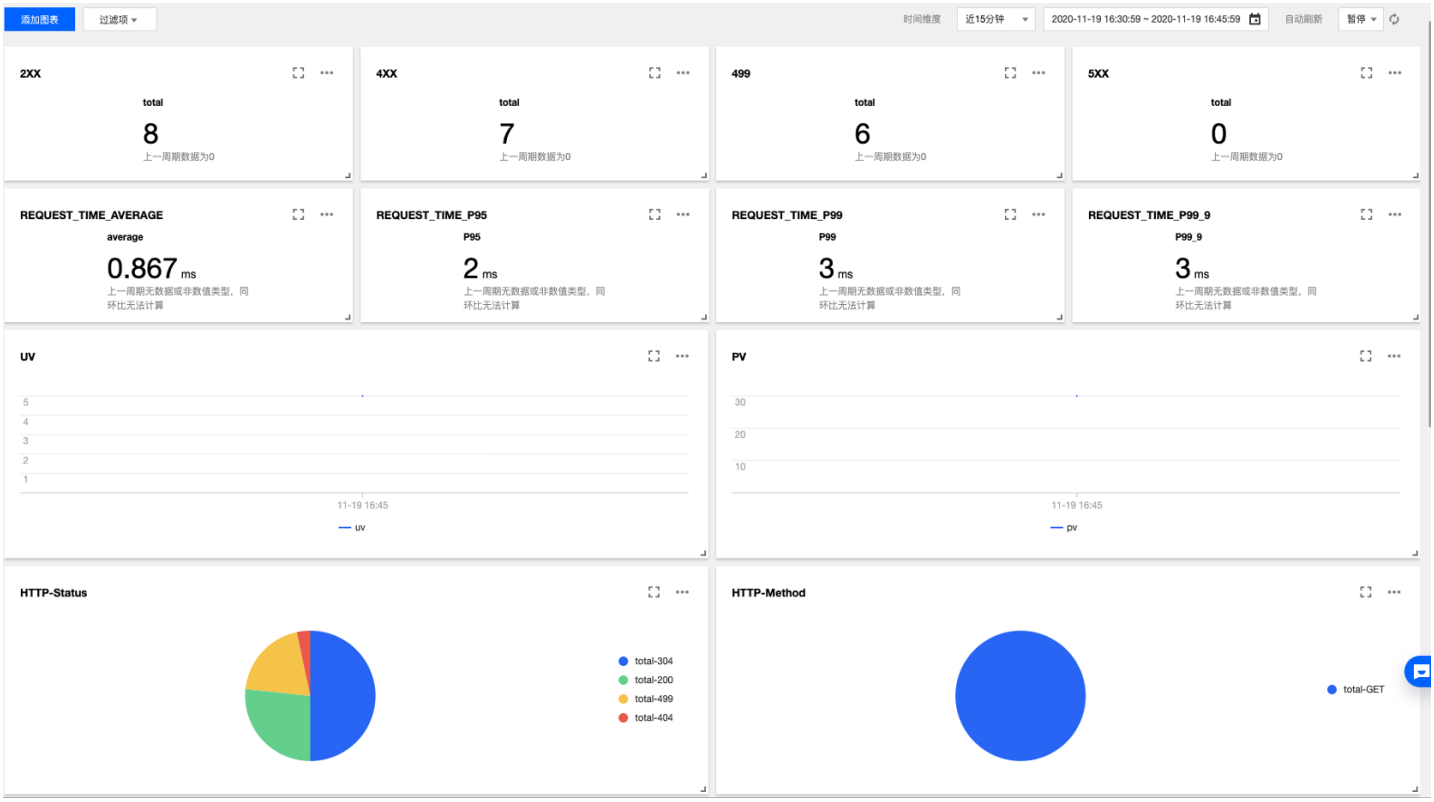
采集日志的指标如下所示：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
name: nginx-ingress-test
resourceVersion: "7169042"
selfLink: /apis/cls.cloud.tencent.com/v1/logconfigs/nginx-ingress-test
uid: 67c96f86-4160-****-****-f6faf8d544dc
spec:
clsDetail:
extractRule:
beginningRegex: (\S+)\s-\s(\S+)\s\[(\S+)\]\s(\S+)\s\"(\w+)\s(\S+)\s([^\"]+)\"\s(\S+)\s(\S+)\s\"([^"]*)\"\s\"([^"]*)\"\s(\S+)\s(\S+)\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s(\S+)
keys:
- remote_addr
- remote_user
- time_local
- timestamp
- method
- url
- version
- status
- body_bytes_sent
- http_referer
- http_user_agent
- request_length
- request_time
- proxy_upstream_name
- proxy_alternative_upstream_name
- upstream_addr
- upstream_response_length
- upstream_response_time
```

```
- upstream_status
- req_id
logRegex: (\S+)\s-\s(\S+)\s\[(\S+)\]\s(\S+)\s\"(\w+)\s(\S+)\s([^\"]+)\"\s(\S+)\s(\S+)\s\"([^"]*)\"\s\"([^"]*)\"\s(\S+)\s(\S+)\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s(\S+)
logType: fullregex_log
topicId: 56766bad-368e-****-****-ed77ebcdefa8
inputDetail:
containerFile:
container: controller
filePattern: nginx_access.log
logPath: /var/log/nginx
namespace: default
workload:
kind: deployment
name: nginx-ingress-nginx-controller
type: container_file
```

Nginx-ingress 日志仪表盘

TKE Nginx-ingress 开启日志采集功能将会自动为您创建一个标准的日志仪表盘，您也可以根据业务需要自行在 CLS 控制台配置图表。如下图所示：



Nginx-ingress 监控配置

最近更新时间：2022-01-19 14:47:07

TKE Nginx-ingress 监控介绍

Nginx Controller 现已提供组件运行状态相关的监控数据，您可以通过配置 Nginx-ingress 监控，开启 Nginx-ingress 监控能力。

前提条件

- 集群已关联云原生监控 Prometheus，操作详情可参见 关联集群。
- 云原生监控 Prometheus 需要与 Nginx 在同一个网络平面。

采集指标

TKE Nginx-ingress 自动配置以下采集指标：

- **Nginx 状态**
  - nginx_ingress_controller_connections_total
  - nginx_ingress_controller_requests_total
  - nginx_ingress_controller_connections
- **进程相关**
  - nginx_ingress_controller_num_procs
  - nginx_ingress_controller_cpu_seconds_total
  - nginx_ingress_controller_read_bytes_total
  - nginx_ingress_controller_write_bytes_total
  - nginx_ingress_controller_resident_memory_bytes
  - nginx_ingress_controller_virtual_memory_bytes
  - nginx_ingress_controller_oldest_start_time_seconds
- **Socket 相关**
  - nginx_ingress_controller_request_duration_seconds
  - nginx_ingress_controller_request_size
  - nginx_ingress_controller_response_duration_seconds
  - nginx_ingress_controller_response_size
  - nginx_ingress_controller_bytes_sent
  - nginx_ingress_controller_ingress_upstream_latency_seconds

您也可以根据业务需要自行配置监控采集指标，指标详情可参见 官方文档。

Nginx-ingress 监控 Grafana 面板

TKE Nginx-ingress 开启监控功能后将关联云原生监控 Prometheus，云原生监控 Prometheus 自带一个 Grafana，您可以在 Nginx-ingress 组件页面直接跳转到对应的 Grafana 面板，如下图所示：


NGINX Ingress controller
clusterId
Namespace
All
Controller Class
All
Controller
All
Ingress
All
Config Reloads
Last 1 hour
5s
Controller Request Volume
0 ops
Controller Connections
4.0
Controller Success Rate (non-4|5xx responses)
N/A
Config Reloads
0
Last Config Failed
N/A
Ingress Request Volume
1.0 reqps
0.5 reqps
0 reqps
-0.5 reqps
-1.0 reqps
avg
18:20
18:30
18:40
18:50
19:00
19:10
Ingress Success Rate (non-4|5xx responses)
100.0%
50.0%
0%
-50.0%
-100.0%
avg
Network I/O pressure
1.0 B/s
0.5 B/s
0 B/s
-0.5 B/s
-1.0 B/s
Average Memory Usage
147.3 MiB
146.9 MiB
146.4 MiB
145.9 MiB
145.4 MiB
145.0 MiB
Average CPU Usage
0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
cores
Ingress Percentile Response Times and Transfer Rates
Ingress
P50 Latency
P90 Latency
P99 Latency
IN
OUT
nginx-ingress-test
NaN
NaN
NaN
0 B/s
0 B/s
Ingress Certificate Expiry

存储管理
概述

最近更新时间：2022-01-19 14:24:26

集群的存储管理是保存业务数据的重要组件。目前，腾讯云容器服务（Tencent Kubernetes Engine，TKE）支持多种类型的存储。

存储类型

存储类型	说明	使用方法
腾讯云硬盘（CBS）	CBS 提供数据块级别的持久性存储，通常用作需要频繁更新、细粒度更新的数据（如文件系统、数据库等）的主存储设备，具有高可用、高可靠和高性能的特点。	TKE 支持通过创建 PV/PVC，并为工作负载挂载动（静）态数据卷的方式使用云硬盘 CBS。详情参见 使用云硬盘 CBS。
腾讯云文件存储（CFS）	CFS 提供了标准的 NFS 及 CIFS/SMB 文件系统访问协议，为多个 CVM 实例或其他计算服务提供共享的数据源，支持弹性容量和性能的扩展，是一种高可用、高可靠的分布式文件系统，适合于大数据分析、媒体处理和内容管理等场景。	TKE 支持通过创建 PV/PVC，并为工作负载挂载动（静）态数据卷的方式使用文件存储 CFS。详情参见 使用文件存储 CFS。
腾讯云对象存储（COS）	COS 是腾讯云提供的一种存储海量文件的分布式存储服务，通过 COS 可以进行多格式文件的上传、下载和管理。	TKE 支持通过创建 PV/PVC，并为工作负载挂载静态数据卷的方式使用对象存储 COS。详情参见 使用对象存储 COS。
其他类型	-	在创建工作负载时，TKE 还支持使用以下类型的本地存储，如使用主机路径、NFS 盘、配置项（ConfigMap）、密钥（Secret）等。详情参见 使用其他存储卷。

> 说明：
> 建议使用云存储服务，否则当节点异常无法恢复时，本地存储的数据同样不能恢复。

相关概念

- **PersistentVolume（PV）**：集群内的存储资源。PV 独立于 Pod 的生命周期，可根据不同的 StorageClass 类型创建不同类型的 PV。
- **PersistentVolumeClaim（PVC）**：集群内的存储请求。例如，PV 是 Pod 使用的节点资源，PVC 则声明使用 PV 资源。当 PV 资源不足时，PVC 可动态创建 PV。

使用对象存储 COS

最近更新时间：2022-02-16 10:48:15

操作场景

腾讯云容器服务 TKE 支持通过创建 PersistentVolume（PV）/PersistentVolumeClaim（PVC），并为工作负载挂载数据卷的方式使用腾讯云对象存储 COS。本文介绍如何在 TKE 集群中为工作负载挂载对象存储。

准备工作

安装对象存储扩展组件

> 说明：
> 若您的集群已安装 COS-CSI 扩展组件，则请跳过此步骤。

1. 登录 容器服务控制台。
2. 选择左侧导航栏中的**集群**，进入集群管理界面。
3. 选择需新建组件的集群 ID，单击集群详情页左侧栏中的**组件管理**。
4. 在“组件管理”页面，单击**新建**，进入“新建组件”页面。
5. 勾选** COS（腾讯云对象存储）**并单击完成即可。

创建访问密钥

> 注意：
> 为避免主账号密钥泄露造成您的云上资产损失，建议您参照 安全设置策略 停止使用主账号登录控制台或者使用主账号密钥访问云 API，并使用已授予相关管理权限的子账号/协作者进行相关资源操作。
>
> 本文以已授予访问管理相关权限的子用户创建或查看访问密钥为例，关于如何创建子用户并实现访问管理权限请参考文档 自定义创建子用户。

1. 使用子账号用户登录 访问管理控制台 ，单击左侧导航栏中的**访问密钥** > **API 密钥管理**，进入 “API 密钥管理”管理界面。
2. 单击**新建密钥**等待新建完成即可。

> 说明：
> - 一个子用户最多可以创建两个 API 密钥。
> - API 密钥是构建腾讯云 API 请求的重要凭证，为了您的财产和服务安全，请妥善保存和定期更换密钥。当您更换密钥后，请及时删除旧密钥。

创建存储桶

> 注意：
> 根据相关法规和政策要求，使用腾讯云对象存储服务前需要完成 实名认证。

1. 登录 对象存储控制台，单击左侧导航中**存储桶列表**，进入“存储桶列表”页面。

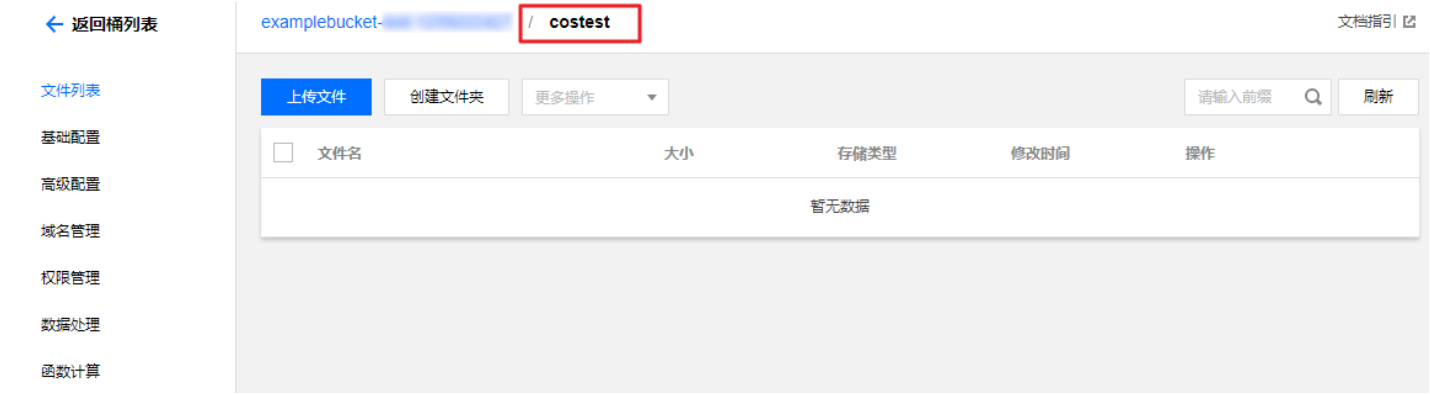
2. 单击**创建存储桶**，在弹出的“创建存储桶”窗口，参考以下信息进行创建。如下图所示：

- **所属地域**：请选择本文中目标集群所在地域，设置后不可修改。详情请参见 地域和访问域名。
- **名称**：存储桶名称由 [自定义名称]-[开发商 APPID] 构成。请输入自定义名称，设置后不可修改。命名说明请参见存储桶的 命名规范。
- **访问权限**：存储桶默认提供**私有读写**、**公有读私有写**和**公有读写**三种访问权限，设置后仍可修改。
  - **私有读写**：仅该存储桶的创建者及有授权的账号具备该存储桶对象的读写权限。存储桶访问权限默认为私有读写，推荐使用。
  - **公有读私有写**：任何人（包括匿名访问者）都具备该存储桶对象的读权限，但仅有存储桶创建者及有授权的账号具备该存储桶对象的写权限。
  - **公有读写**：任何人（包括匿名访问者）都具备该存储桶对象的读权限和写权限，不推荐使用。
- **版本控制**：该功能支持在相同存储桶中保留对象的多个版本。
- **日志存储**：该功能支持记录跟存储桶操作相关的各种请求日志。
- **存储桶标签**：存储桶标签是一个键值对（key = value），是用于管理存储桶的标识，便于分组管理存储桶。详情请参见 设置存储桶标签。
- **服务端加密**：支持**不加密**和**SSE-COS 加密**（即由对象存储托管密钥的服务端加密）两种方式。
  - **SSE-COS 加密**：对象存储托管密钥的服务端加密，由对象存储托管主密钥和管理数据，用户可通过对象存储直接对数据进行管理和加密。详情请参见 服务端加密概述。

3. 确认信息无误后单击**创建**即可。创建完成后，即可在存储桶列表中进行查看。

获取存储桶子目录

1. 在“存储桶列表”页面，选择已创建的存储桶名称，进入该存储桶名称的详情页。
2. 在存储桶详情页面，选择需要挂载的子文件夹，进入该文件夹详情页。在页面右上角获取子目录路径 /costest。如下图所示：

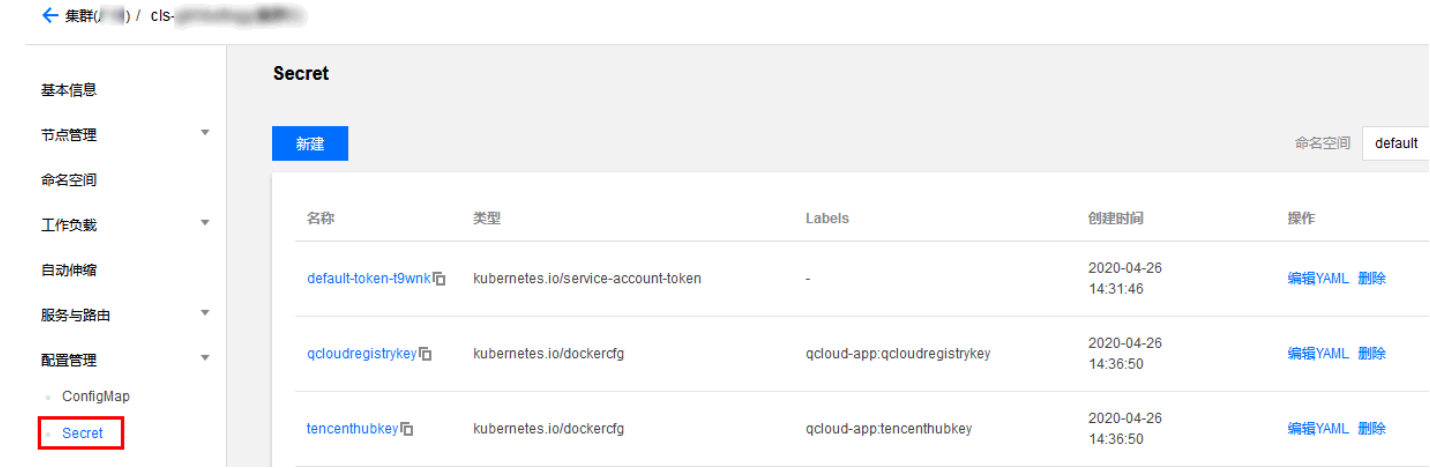


操作步骤

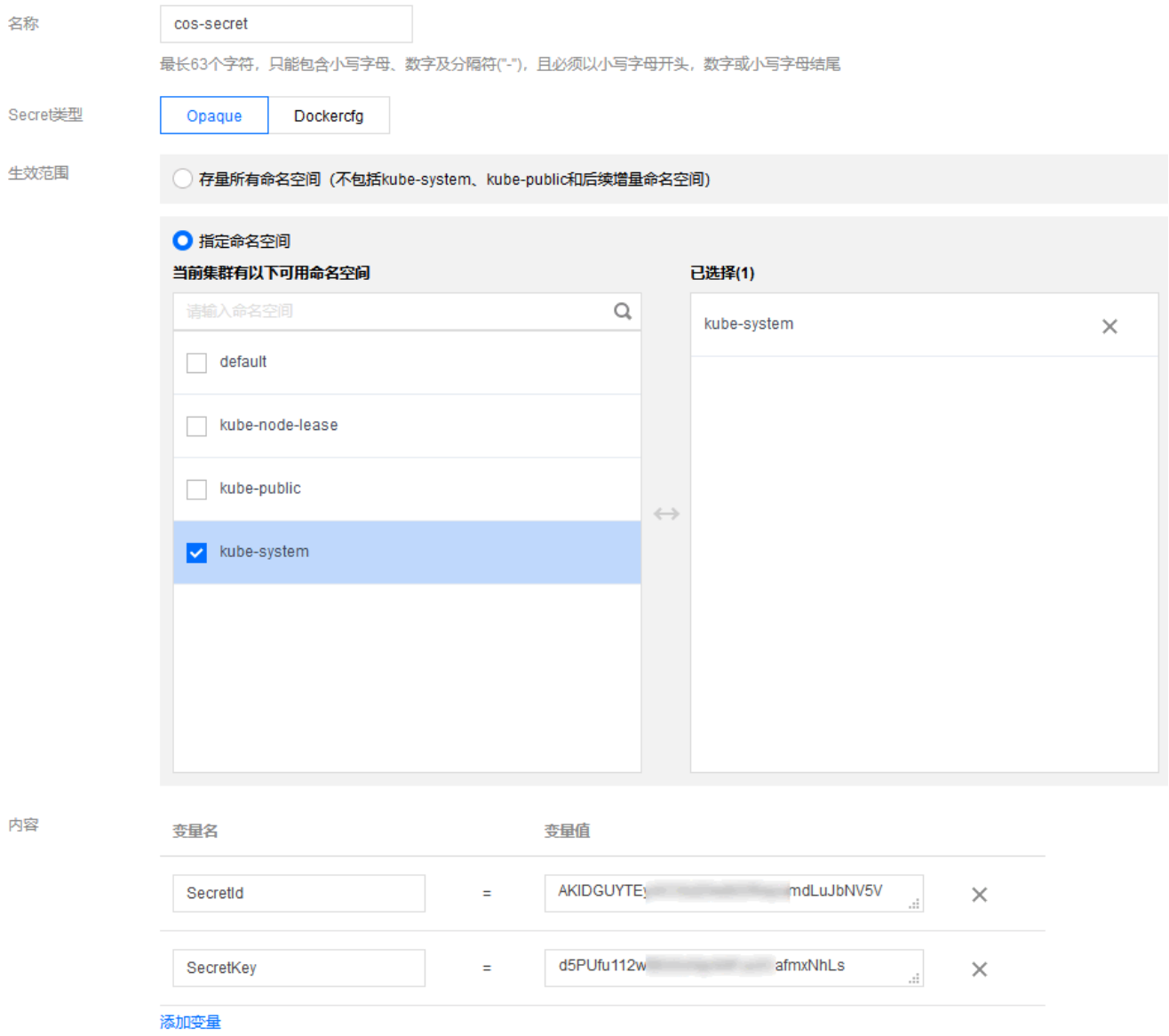
通过控制台使用对象存储

创建可以访问对象存储的 Secret

1. 单击左侧导航栏中的**集群**，进入集群管理界面。
2. 选择目标集群 ID，进入集群详情页面。
3. 在集群详情页面，选择左侧菜单栏中的**配置管理** > **Secret**，进入 “Secret” 页面。如下图所示：



4. 单击**新建**进入“新建 Secret” 页面，根据以下信息进行设置。如下图所示：



- **名称**：自定义，本文以 cos-secret 为例。
- **Secret 类型**：选择** Opaque**，该类型适用于保存密钥证书和配置文件，Value 将以 Base64 格式编码。
- **生效范围**：选择**指定命名空间**，请确保 Secret 创建在 kube-system 命名空间下。
- **内容**：此处用于设置 Secret 访问存储桶（Bucket）所需的访问密钥，需包含变量名 SecretId 和 SecretKey 及其分别所对应的变量值。请参考 创建访问密钥 完成创建，并前往 API 密钥管理 页面获取访问密钥。

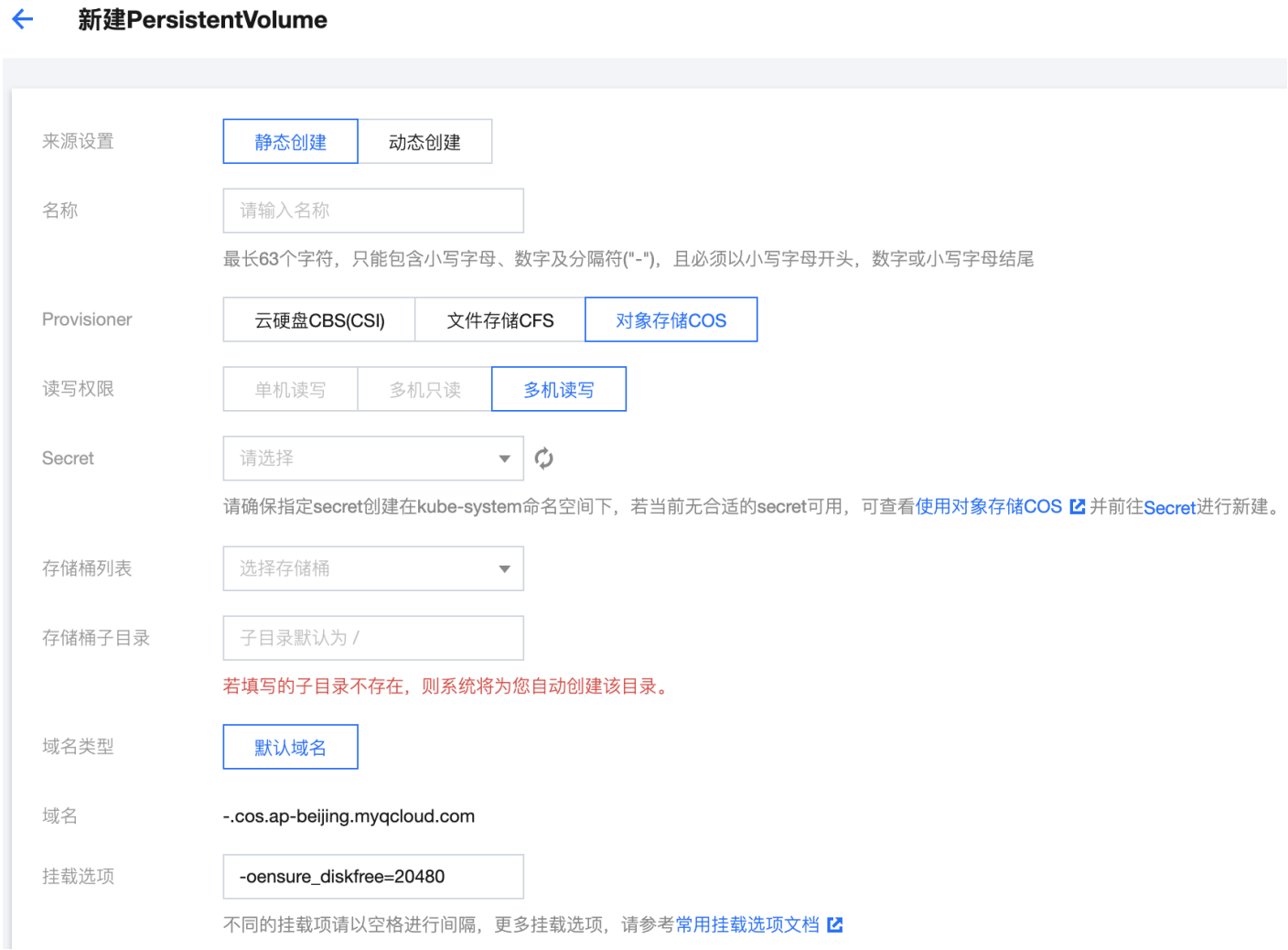
5. 单击**创建 Secret** 即可。

创建支持 COS-CSI 动态配置的 PV

> ⚠ 注意：
> 本步骤需使用存储桶，若当前地域无可用存储桶，则请参考 创建存储桶 进行创建。

1. 在目标集群详情页面，选择左侧菜单栏中的**存储** > **PersistentVolume**，进入 “PersistentVolume” 页面。

2. 单击**新建**进入“新建 PersistentVolume” 页面，参考以下信息创建 PV。如下图所示：



主要参数信息如下：

- **来源设置：选择静态创建。**
- **名称：自定义，本文以** cos-pv **为例。**
- **Provisioner：选择为对象存储 COS。**
- **读写权限：对象存储仅支持多机读写。**
- **Secret：选择已在** 步骤 1 **创建的 Secret，本文以** cos-secret **为例（请确保 Secret 创建在** kube-system **命名空间下）。**
- **存储桶列表：用于保存对象存储中的对象，按需选择可用存储桶即可。**
- **存储桶子目录：填写已在** 获取存储桶子目录 **中获取的存储桶子目录，本文以** /costest **为例。若填写的子目录不存在，则系统将为您自动创建。**
- **域名：展示为默认域名，您可以使用该域名对存储桶进行访问。**
- **挂载选项：COSFS 工具支持将存储桶挂载到本地，挂载后可直接操作对象存储中的对象，此项用于设置相关限制条件。本例中挂载选项** -oensure_diskfree=20480 **表示当缓存文件所在磁盘剩余空间不足 20480MB 时，COSFS 将启动失败。**

> **说明：**
> **不同的挂载项请以空格进行间隔，更多挂载选项请参见** 常用挂载选项文档 **。**

3. 单击**创建 PersistentVolume** 即可。

创建 PVC 绑定 PV

> **注意：**
> **请勿绑定状态为 Bound 的 PV。**

1. 在目标集群详情页，选择左侧菜单栏中的**存储** > **PersistentVolumeClaim**，进入 “PersistentVolumeClaim” 页面。
2. 单击**新建**进入 “新建 PersistentVolumeClaim” 页面，参考以下信息创建 PVC。如下图所示：

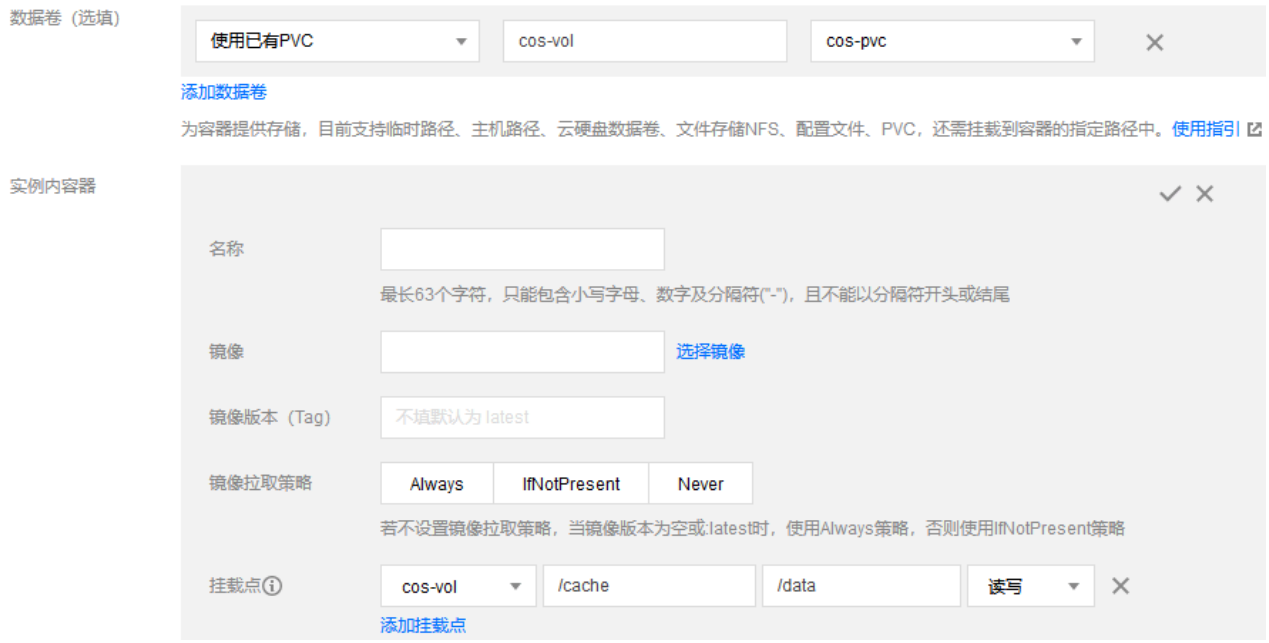
- **名称**：自定义，本文以 cos-pvc 为例。
- **命名空间**：选择为 kube-system。
- **Provisioner**：选择**对象存储 COS**。
- **读写权限**：对象存储仅支持多机读写。
- **PersistentVolume**：选择在 步骤 2 中已创建的 PV，本文以 cos-pv 为例。

3. 单击**创建 PersistentVolumeClaim** 即可。

创建 Pod 使用的 PVC

> **说明：**
> 本步骤以创建工作负载 Deployment 为例。

1. 在目标集群详情页，选择左侧菜单栏中的**工作负载** > **Deployment**，进入 “Deployment” 页面。
2. 单击**新建**进入 “新建 Workload” 页面，参考 创建 Deployment 进行创建，并设置数据卷挂载。如下图所示：



- **数据卷（选填）**：
  - **挂载方式**：选择**使用已有 PVC**。
  - **数据卷名称**：自定义，本文以 cos-vol 为例。

- **选择 PVC**：选择已在 步骤 3 中创建的 PVC，本文以选择 cos-pvc 为例。
- **实例内容器**：单击**添加挂载点**，进行挂载点设置。
  - **数据卷**：选择为该步骤中所添加的数据卷 “cos-vol”。
  - **目标路径**：填写目标路径，本文以 /cache 为例。
  - **挂载子路径**：仅挂载选中数据卷中的子路径或单一文件。例如，./data 或 data。

3. 单击**创建 Workload** 即可。

通过 YAML 文件使用对象存储

创建可以访问对象存储的 Secret

可通过 YAML 创建可以访问对象存储的 Secret，模板如下：

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
name: cos-secret
# Replaced by your secret namespace.
namespace: kube-system
data:
# Replaced by your temporary secret file content. You can generate a temporary secret key with these docs:
# Note: The value must be encoded by base64.
SecretId: VWVEJxRk5Fb0JGbDA4M...(base64 encode)
SecretKey: Qa3p4ZTVCMFlQek...(base64 encode)
```

创建支持 COS-CSI 动态配置的 PV

可通过 YAML 创建 PV 以支持 COS-CSI 动态配置，模板如下：

```
apiVersion: v1
kind: PersistentVolume
metadata:
name: cos-pv
spec:
accessModes:
- ReadWriteMany
capacity:
storage: 10Gi
csi:
driver: com.tencent.cloud.csi.cosfs
nodePublishSecretRef:
name: cos-secret
namespace: kube-system
volumeAttributes:
# Replaced by the url of your region.
url: http://cos.ap-XXX.myqcloud.com
# Replaced by the bucket name you want to use.
bucket: XXX-1251707795
# You can specify sub-directory of bucket in cosfs command in here.
path: /costest
# You can specify any other options used by the cosfs command in here.
# additional_args: "-oallow_other"# Specify a unique volumeHandle like bucket name.(this value must different from other pv's volumeH
andle)
volumeHandle: XXX
```

```
persistentVolumeReclaimPolicy: Retain
volumeMode: Filesystem
```

创建 PVC 绑定 PV

可通过 YAML 创建绑定上述 PV 的 PVC，模板如下：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: cos-pvc
spec:
accessModes:
- ReadWriteMany
resources:
requests:
storage: 1Gi
# You can specify the pv name manually or just let kubernetes to bind the pv and pvc.
# volumeName: cos-pv
# Currently cos only supports static provisioning, the StorageClass name should be empty.
storageClassName: ""
```

创建 Pod 使用 PVC

可通过 YAML 创建 Pod，模板如下：

```
apiVersion: v1
kind: Pod
metadata:
name: pod-cos
spec:
containers:
- name: pod-cos
command: ["tail", "-f", "/etc/hosts"]
image: "centos:latest"
volumeMounts:
- mountPath: /data
name: cos
resources:
requests:
memory: "128Mi"
cpu: "0.1"
volumes:
- name: cos
persistentVolumeClaim:
# Replaced by your pvc name.
claimName: cos-pvc
```

相关信息

更多关于如何使用对象存储的信息请参见 README_COSFS.md。

使用文件存储 CFS
文件存储使用说明

最近更新时间：2022-01-07 15:39:08

操作场景

腾讯云容器服务 TKE 支持通过创建 PV/PVC，并为工作负载挂载数据卷的方式使用腾讯云文件存储 CFS。本文介绍如何通过以下两种方式在集群中为工作负载挂载文件存储：

- 方式1：动态创建文件存储
- 方式2：使用已有的文件存储

准备工作

安装文件存储扩展组件

> 说明：
> 若您的集群已安装 CFS-CSI 的扩展组件，则请跳过此步骤。

1. 登录 容器服务控制台。
2. 选择左侧导航栏中的**集群**，进入集群管理界面。
3. 选择需新建组件的集群 ID，单击集群详情页左侧栏中的**组件管理**。
4. 在“组件管理”页面，单击**新建**，进入“新建组件”页面。
5. 勾选**CFS（腾讯云文件存储）**并单击完成即可。

操作步骤

动态创建文件存储

当您需要动态创建文件存储时，可以按照以下步骤进行操作：

1. 创建文件存储类型的 StorageClass，定义需使用的文件存储模板。
2. 通过 StorageClass 创建 PVC，进一步定义所需的文件存储参数。
3. 创建工作负载数据卷时选择已创建的 PVC，并设置容器挂载点。
   详细操作步骤请参见 StorageClass 管理文件存储模板。

使用已有的文件存储

当您需要使用已有文件存储时，可以按照以下步骤进行操作：

1. 通过已有的文件存储创建 PV。
2. 创建 PVC 时设置与上述创建的 PV 相同的 StorageClass 和容量。
3. 创建工作负载时，选择上述 PVC。
   详细操作步骤请参见 PV 和 PVC 管理文件存储。

相关信息

更多关于如何使用文件存储的信息请参见 README_CFS.md。

StorageClass 管理文件存储模板

最近更新时间：2022-02-16 10:48:11

操作场景

集群管理员可使用 StorageClass 为容器服务集群定义不同的存储类型。容器服务已默认提供块存储类型的 StorageClass，您可通过 StorageClass 配合 PersistentVolumeClaim 动态创建需要的存储资源。

本文介绍通过控制台、Kubectl 两种方式创建文件存储 CFS 类型的 StorageClass，自定义文件存储使用所需的模板。

准备工作

安装文件存储扩展组件

> 说明：
> 若您的集群已安装 CFS-CSI 的扩展组件，则请跳过此步骤。

1. 登录 容器服务控制台。
2. 单击左侧导航栏中的**集群**，进入**集群管理**页面。
3. 选择需新建组件的集群 ID，进入**集群详情**页面。
4. 在“集群详情页”，选择**组件管理** > **新建**，进入**新建组件**页面。
5. 在“新建组件”页面，勾选**CFS（腾讯云文件存储）**并单击**完成**即可。

创建子网

创建 StorageClass 过程中，需设置文件存储归属子网，为确保文件存储所处私有网络下每一个可用区均拥有合适子网，建议您提前进行子网创建。

1. 登录 私有网络控制台，单击左侧导航栏中的**子网**。
2. 在“子网”列表页面单击**+新建**，在弹出的“创建子网”窗口中设置子网名称、VPC 网段、CIDR、可用区和关联路由表等基本信息。如下图所示：

3. （可选）单击**+新增一行**，可以同时创建多个子网。
4. 单击**创建**即可。

创建权限组并添加权限组规则

创建 StorageClass 过程中，需为文件系统配置权限组，为确保具备合适的权限组，建议您提前进行权限组创建。

1. 登录 文件存储控制台 ，选择左侧导航栏中的**权限组**。
2. 在“权限组”页面单击**新建**，在弹出的“创建权限组”窗口中配置权限组名称和备注。
3. 单击**确定**，完成创建即可在“权限组”页面进行查看。
4. 单击该权限组 ID，进入其详情页，可进行权限组规则的添加、编辑或删除操作。如果权限组中没有添加规则，则会允许全部。详情请参见 添加权限组规则。

获取文件系统 FSID

1. 在 文件系统控制台，单击需获取 FSID 的文件系统 ID，进入该文件系统详情页。

2. 选择**挂载点信息**页签，从 “Linux 下挂载” 获取该文件系统的 FSID。如下图所示， a43qadkl为该文件系统的 FSID。

说明：

为了获取更好的稳定性，在通过 YAML 创建 PV 并使用 NFSV3 协议挂载时，需要指定待挂载文件系统对应的 FSID。

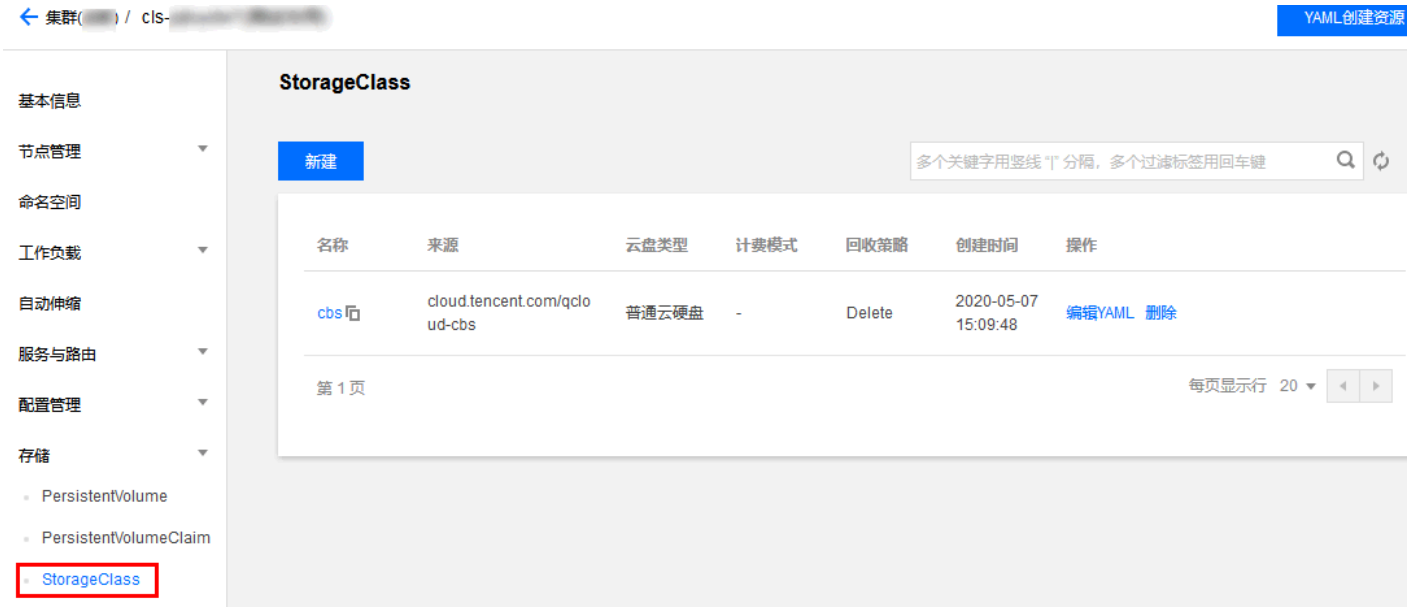
操作步骤

控制台操作指引

通过控制台创建 StorageClass

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。

3. 选择左侧菜单栏中的**存储** > **StorageClass**，进入 “StorageClass” 页面。如下所示：



4. 单击**新建**，进入 “新建 StorageClass” 页面，参考以下信息进行创建。如下所示：

主要参数信息如下：

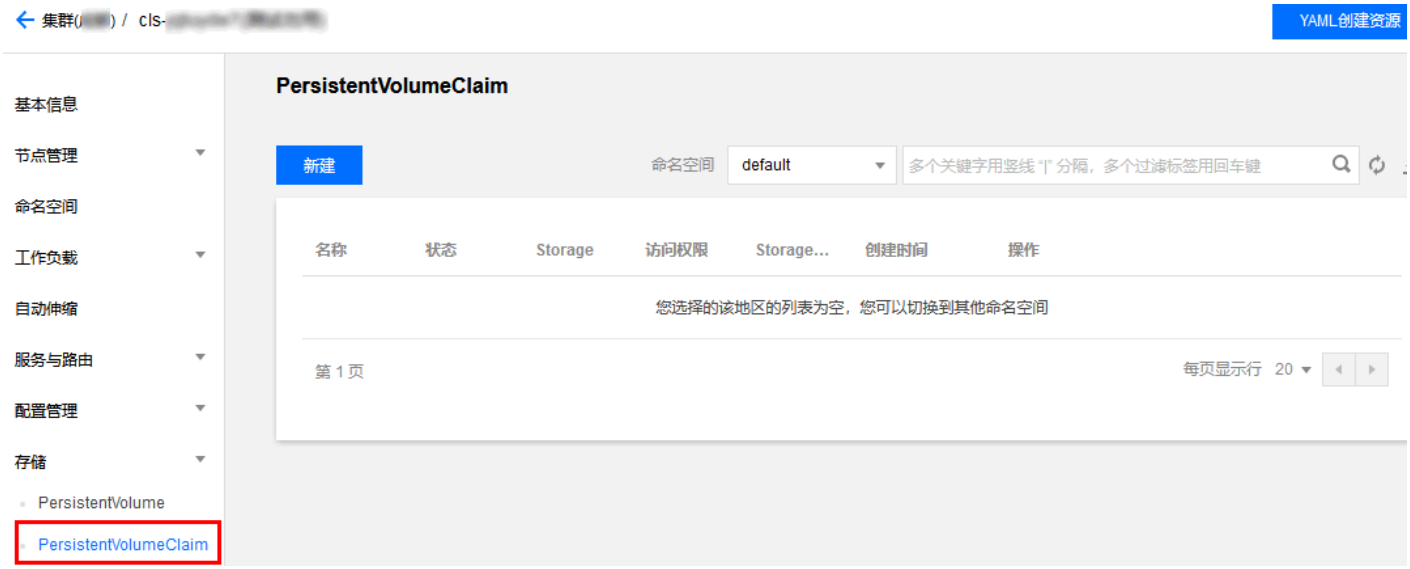
- **名称：自定义，本文以** cfs-storageclass **为例。**

- **Provisioner**：选择**文件存储 CFS**。
- **可用区**：表示当前地域下支持使用文件存储的可用区，每个地域下不同可用区所适用的存储类型不完全一致，请参考 可用地域 进行选择。
- **CFS 归属子网**：设置当前可用区下文件系统的所属子网范围，请按需选择。
- **存储类型**：文件存储提供**标准存储**和**性能存储**两种类型的文件系统，每个地域下不同可用区所适用的存储类型不完全一致，请结合控制台实际情况进行选择。
  - **标准存储**：低成本、大容量，适用于成本敏感及大容量的业务。例如数据备份、文件共享、日志存储等场景。
  - **性能存储**：高吞吐、高 IOPS，适用于 IO 密集型工作负载。例如高性能计算、媒资渲染、机器学习、DevOps、办公 OA 等场景。
- **文件服务协议**：默认为 NFS 协议，允许透明访问服务器上的文件和文件系统。
- **协议版本**：推荐使用 NFSV3 协议挂载获得更好的性能，如果您的应用依赖文件锁（即需要使用多台 CVM 同时编辑一个文件）请使用 NFSV4 协议挂载。
- **权限组**：为文件系统配置权限组，便于进一步管理与文件系统处于同一网络下的来访客户端的访问权限及读写权限。请根据实际需求选择合适的权限组，如不具备，请前往 权限组 页面进行创建。
- **回收策略**：提供**删除**和**保留**两种回收策略，出于数据安全考虑，推荐使用保留回收策略。
  - **删除**：通过 PVC 动态创建的 PV，在 PVC 销毁时，与其绑定的 PV 和存储实例也会自动销毁。
  - **保留**：通过 PVC 动态创建的 PV，在 PVC 销毁时，与其绑定的 PV 和存储实例会被保留。
- **标签**：选择 CFS 实例需要绑定的云标签。该标签将由 StorageClass 动态创建的 CFS 实例自动继承，StorageClass 创建后其绑定的标签参数不支持修改。如现有标签不符合您的要求，请前往 标签控制台 操作。

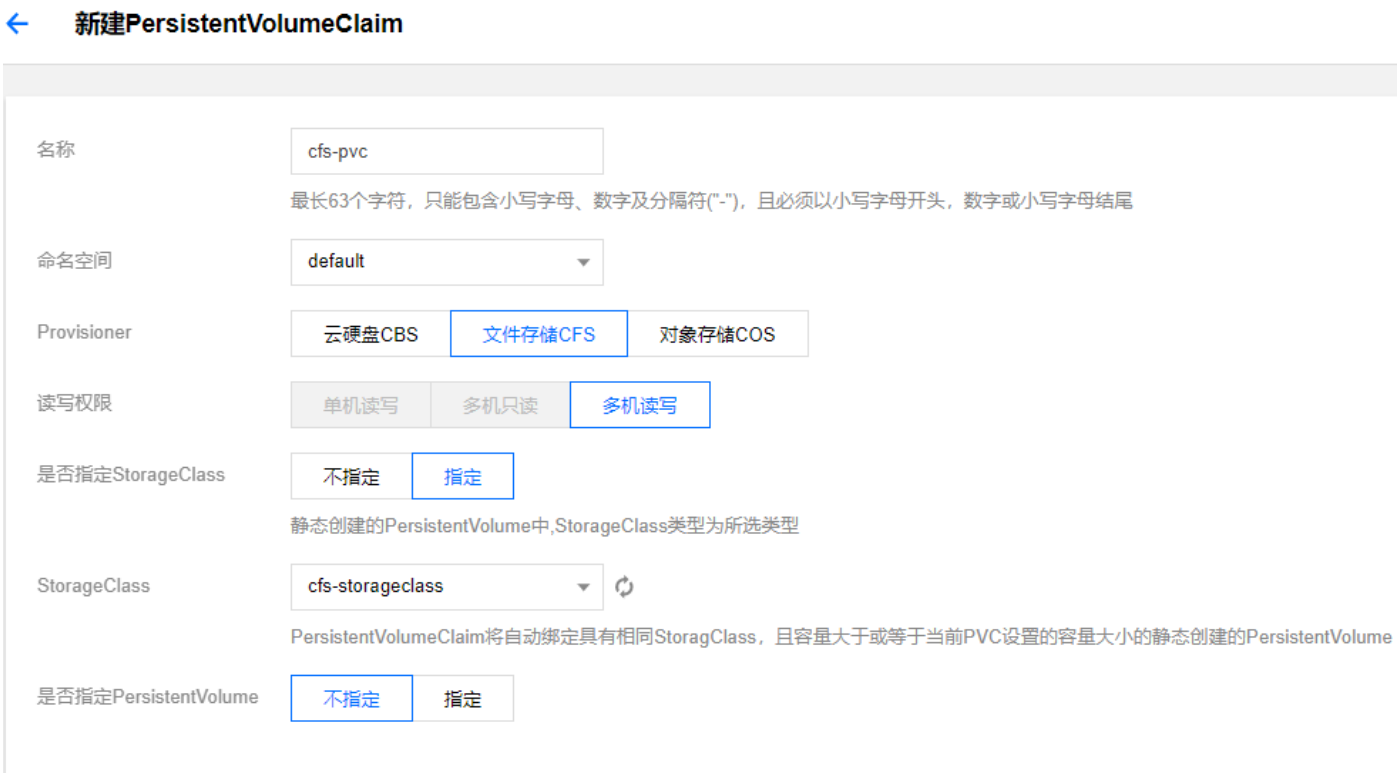
5. 单击 **新建 StorageClass** 即可。

使用指定 StorageClass 创建 PVC

1. 在“集群管理”页，选择需创建 PVC 的集群 ID。
2. 在集群详情页，选择左侧菜单栏中的**存储** > **PersistentVolumeClaim**，进入 “PersistentVolumeClaim” 信息页面。如下图所示：



3. 选择**新建**进入“新建 PersistentVolumeClaim” 页面，参考以下信息设置 PVC 关键参数。如下图所示：



主要参数信息如下：

- **名称**：自定义，本文以 cfs-pvc 为例。
- **命名空间**：选择 “default ”。
- **Provisioner**：选择**文件存储 CFS**。
- **读写权限**：文件存储仅支持多机读写。
- **StorageClass**：按需指定 StorageClass，本文选择以在 创建 StorageClass 步骤中创建的 cfs-storageclass 为例。

> 说明：
>
> - PVC 和 PV 会绑定在同一个 StorageClass 下。
> - 不指定 StorageClass 意味着该 PVC 对应的 StorageClass 取值为空，对应 YAML 文件中的 storageClassName 字段取值为空字符串。

- **PersistVolume**：按需指定 PersistentVolume，本文以不指定 PersistentVolume 为例。

> 说明：
>
> - 系统首先会筛选当前集群内是否存在符合绑定规则的 PV，若没有则根据 PVC 和所选 StorageClass 的参数动态创建 PV 与之绑定。
> - 系统不允许在不指定 StorageClass 的情况下同时选择不指定 PersistVolume。
> - 不指定 PersistVolume。详情请参见 查看 PV 和 PVC 的绑定规则。

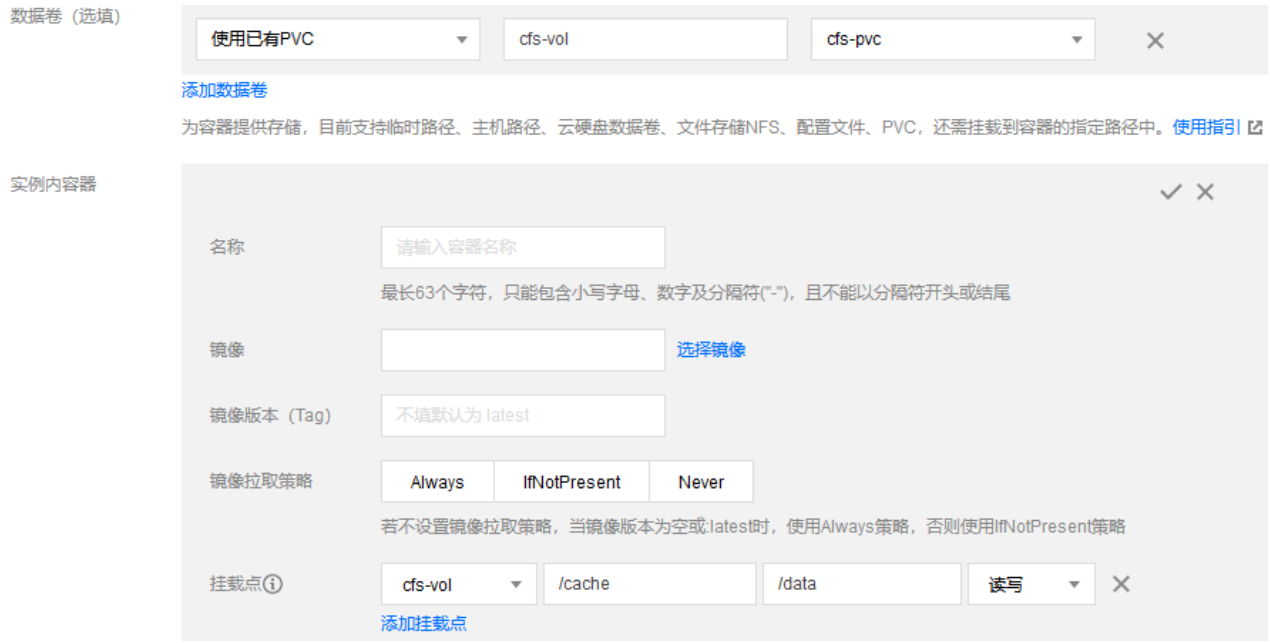
4. 单击**创建 PersistentVolumeClaim**，即可完成创建。

创建 Workload 使用 PVC 数据卷

> 说明：
> 该步骤以创建工作负载 Deployment 为例。

1. 在“集群管理”页面，选择目标集群 ID，进入待部署 Workload 的集群的 “Deployment” 页面。

2. 选择**新建**，进入“新建 Workload” 页面，参考 创建 Deployment 进行创建，并参考以下信息进行数据卷挂载。如下图所示：



- **数据卷（选填）：**
  - **挂载方式：**选择“使用已有 PVC”。
  - **数据卷名称：**自定义，本文以 cfs-vol 为例。
  - **选择 PVC：**选择在步骤 创建 PVC 中已创建的 “cfs-pvc”。
- **实例内容器：**单击**添加挂载点**，进行挂载点设置。
  - **数据卷：**选择该步骤中已添加的数据卷“cfs-vol”。
  - **目标路径：**填写目标路径，本文以 /cache 为例。
  - **挂载子路径：**仅挂载选中数据卷中的子路径或单一文件。例如， /data 或 /test.txt。

3. 单击**创建 Workload**，完成创建。

> **注意：**
> 如使用 CFS 的 PVC 挂载模式，数据卷支持挂载到多台 Node 主机上。

Kubectl 操作指引

创建 StorageClass

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: cfs
parameters:
vpcid: vpc-xxxxxxxx
subnetid: subnet-xxxxxxxx
vers: "3"
resourcetags: ""
provisioner: com.tencent.cloud.csi.cfs
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

parameters 支持参数如下：

参数	是否可选	描述
zone	否	设置文件存储所在的地域。
pgroupid	否	设置文件存储所归属的权限组。
storagetype	否	默认为标准存储 SD，可取值及描述如下：SD：标准型存储 HP：性能存储
vpcid	是	创建的文件存储所在的私有网络 ID。
subnetid	是	创建的文件存储所在的子网 ID。
vers	是	插件连接文件系统时所使用的协议版本，动态生成的 PV 会继承该参数，目前支持的版本有 "3" 和 "4"。
resourcetags	是	文件系统云标签，生成的文件系统上会打上对应腾讯云标签，多个标签由英文逗号隔开，例如 "a:b,c:d"。

创建 PVC

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: cfs
namespace: default
spec:
accessModes:
- ReadWriteMany
resources:
requests:
storage: 10Gi
storageClassName: cfs
volumeMode: Filesystem
volumeName: XXX
```

参数	是否可选	描述
spec.accessModes	否	cfs 存储支持多读多写
spec.resources.requests.storage	是	无实际意义，具体存储大小只与文件系统种类有关。

说明：

1. CFS 文件存储系统支持根据文件容量大小自动扩展文件系统存储容量，扩展过程不会中断请求和应用。默认创建的 CFS 实例容量大小为 10Gi，容量上限与产品类型相关，详情根据请参考 系统限制。
2. 通过 PVC 动态创建的 PV 将自动继承 StorageClass 中设定的参数，该参数由存储插件自动生成。

PV 和 PVC 管理文件存储

最近更新时间：2022-02-16 10:48:04

操作场景

腾讯云容器服务支持通过创建 PV/PVC，并在创建工作负载添加数据卷时使用已有 PVC，实现通过 PV 和 PVC 管理文件系统。

> 注意：
> 不同地域所支持的文件存储能力有一定差异，请按需选择。详情请参见 文件存储类型和性能规格。

准备工作

安装文件存储扩展组件

> 说明：
> 若您的集群已安装 CFS-CSI 的扩展组件，则请跳过此步骤。

1. 登录 容器服务控制台。
2. 单击左侧导航栏中的**集群**，进入**集群管理**页面。
3. 选择需新建组件的集群 ID，进入**集群详情**页面。
4. 在“集群详情页”，选择**组件管理** > **新建**，进入**新建组件**页面。
5. 在“新建组件”页面，勾选**CFS（腾讯云文件存储）**并单击**完成**即可。

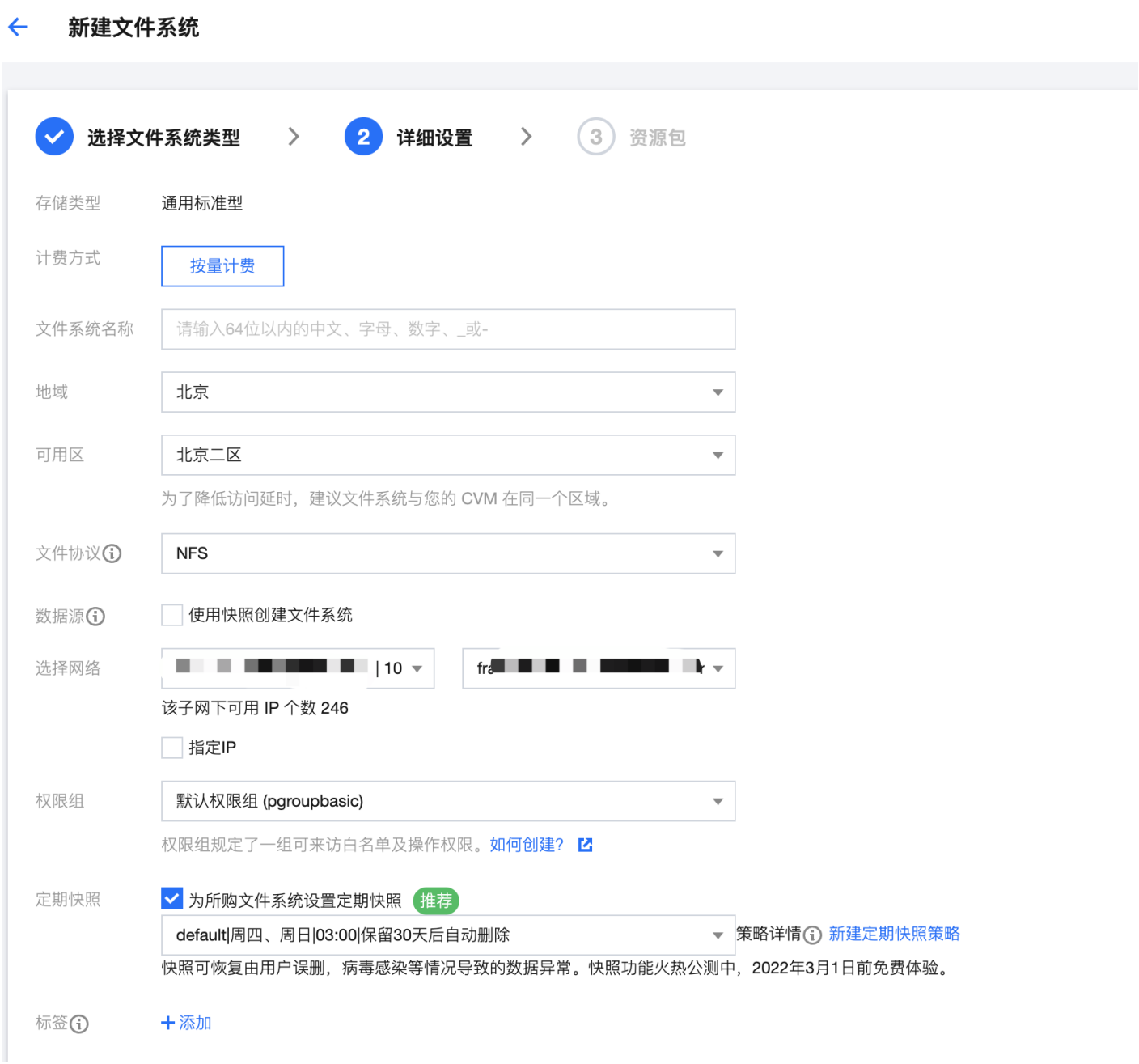
通过控制台创建 StorageClass

由于静态创建文件存储类型的 PV 时，需要绑定同类型可用 StorageClass，请参考 通过控制台创建 StorageClass 完成创建。

创建文件存储

1. 登录 文件存储控制台，进入“文件系统”页面。

2. 单击**新建**，首先选择文件系统类型：提供**标准存储**和**性能存储**两种类型，不同可用区支持类型有一定差异，详情请参见 可用地域；然后进入详细设置：



- **名称**：自定义，本文以 cfs-test 为例。
- **地域**：选择所需要创建文件系统的地域，需确保与集群在同一地域。
- **可用区**：选择所需要创建文件系统的可用区。
- **文件服务协议**：选择文件系统的协议类型，NFS或CIFS/SMB。
  - NFS 协议：更适合于 Linux/Unix 客户端。
  - CIFS/SMB 协议：更适合于 Windows 客户端。
- **数据源**：支持使用快照创建文件系统。
- **选择网络**：需确保与使用该文件系统的集群处于同一私有网络下。
- **权限组**：每个文件系统必须绑定一个权限组，权限组规定了一组可来访白名单及读、写操作权限。
- **标签**：
  - 若已拥有标签，可在此处为新建文件系统添加。
  - 若未拥有标签，则可前往 标签控制台 创建所需要的标签，再为文件系统绑定标签。或可在文件系统创建完成后，再为文件系统添加标签。

3. 单击**立即购买**，等待创建成功即可。

获取文件系统子目录

1. 在“文件系统”页面，单击需获取子目标路径的文件系统 ID，进入该文件系统详情页。
2. 选择**挂载点信息**页签，从 “Linux 下挂载” 获取该文件系统子目录路径 /subfolder。如下图所示：

- localfolder：**指用户本地自己创建的目录。**
- subfolder：**指用户在文件存储的文件系统里创建的子目录，则该文件系统子目录路径即为** /subfolder。

获取文件系统 fsid

> **说明：**
> **为了获取更好的稳定性，在使用 NFSV3 协议挂载时，需要指定待挂载文件系统对应的 FSID。**

1. 在 文件系统控制台，**单击需获取 FSID 的文件系统 ID，进入该文件系统详情页。**

2. 选择**挂载点信息**页签，从 “Linux 下挂载” 获取该文件系统的 FSID。如下图所示， a43qadkl为该文件系统的 FSID。

操作步骤

静态创建 PV

> **说明：**
> **静态创建 PV 适用于已有存量的文件存储，并在集群内使用的场景。**

1. 登录容器服务控制台，选择左侧导航栏中的 集群。
2. 在“集群管理”页面，选择需创建 PV 的集群 ID，进入待创建 PV 的集群管理页面。
3. 选择左侧菜单栏中的**存储** > **PersistentVolume**，进入 “PersistentVolume” 页面。如下图所示：

4. 单击**新建**进入“新建 PersistentVolume” 页面，参考以下信息设置 PV 参数。如下图所示：

- **来源设置**：选择**静态创建**。
- **名称**：自定义，本文以 cfs-pv 为例。
- **Provisioner**：选择**文件存储 CFS**。
- **读写权限**：文件存储仅支持多机读写。
- **StorageClass**：按需选择合适的 StorageClass。本文以选择在 通过控制台创建 StorageClass 步骤中创建的 cfs-storageclass 为例。

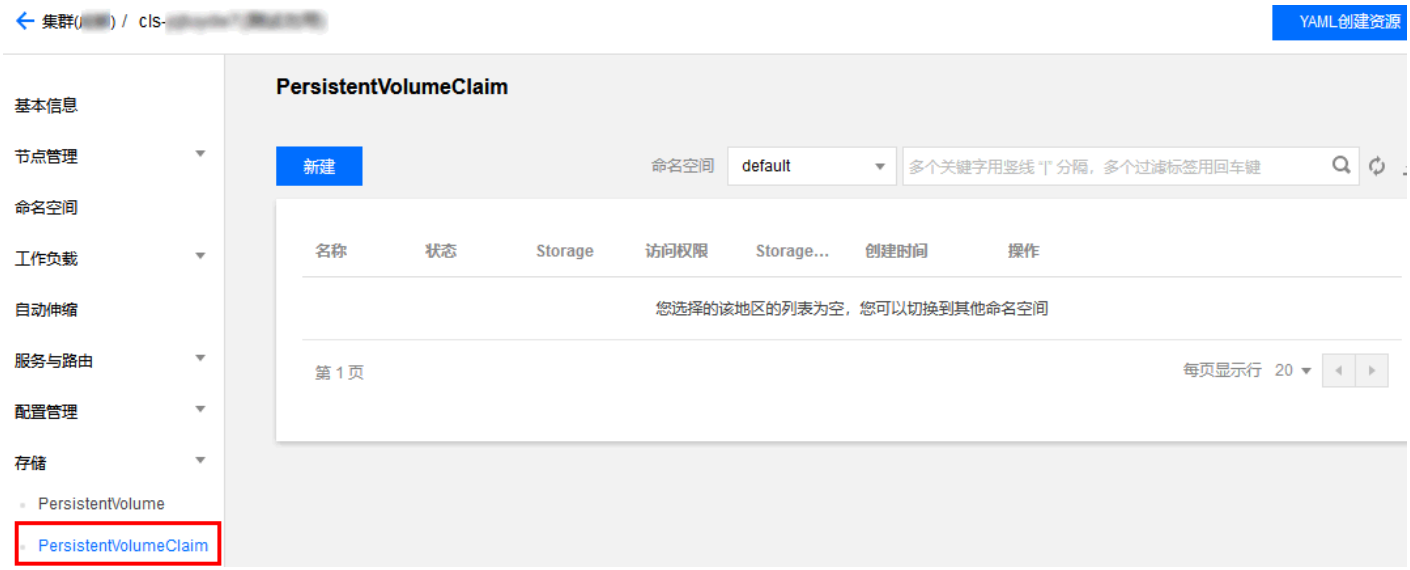
> **说明**：
>
> - PVC 和 PV 会绑定在同一个 StorageClass 下。
> - 不指定 StorageClass 意味着该 PV 对应的 StorageClass 取值为空，对应 YAML 文件中的 storageClassName 字段取值为空字符串。

- **选择 CFS**：需确保文件存储与当前集群处于同一私有网络下，本文以选择在 创建文件存储 步骤中创建的 cfs-test 为例。
- **CFS 子目录**：填写已在步骤 获取文件系统子目录 中获取的文件系统子路径，本文以 /subfolder 为例。

5. 单击**创建 PersistentVolume**，即可完成创建。

创建 PVC

1. 在目标集群详情页，选择左侧菜单栏中的**存储** > **PersistentVolumeClaim**，进入 “PersistentVolumeClaim” 页面。如下图所示：



2. 选择**新建**进入“新建 PersistentVolumeClaim” 页面，参考以下信息设置 PVC 关键参数。如下图所示：

- **名称**：自定义，本文以 cfs-pvc 为例。
- **命名空间**：选择 “default”。
- **Provisioner**：选择**文件存储 CFS**。
- **读写权限**：文件存储仅支持多机读写。
- **StorageClass**：按需选择合适的 StorageClass。本文以选择在 通过控制台创建 StorageClass 步骤中创建的 cfs-storageclass 为例。

> **说明：**
>
> - PVC 和 PV 会绑定在同一个 StorageClass 下。
> - 不指定意味着该 PVC 对应的 StorageClass 取值为空，对应 YAML 文件中的 storageClassName 字段取值为空字符串。

- **PersistVolume**：按需指定 PersistentVolume，本文选择以在 静态创建 PV 步骤中创建的 cfs-pv 为例。

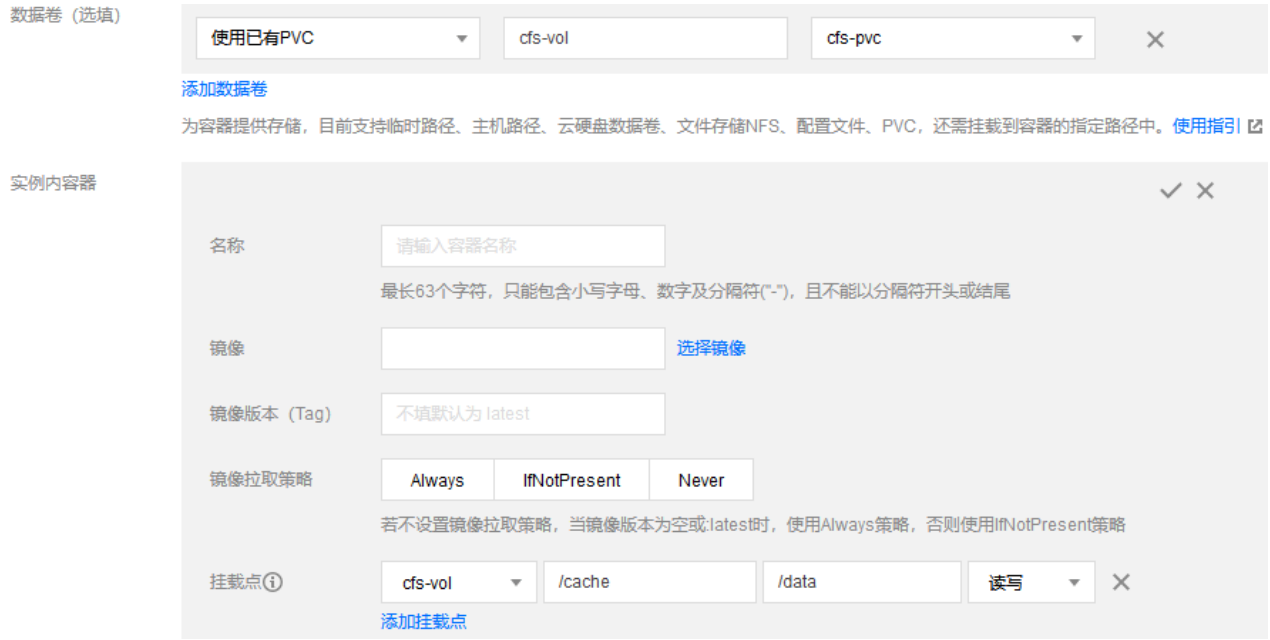
> 说明：
>
> - 只有与指定的 StorageClass 相同并且状态为 Available 和 Released 的 PV 为可选状态，如果当前集群内没有满足条件的 PV 可选，请选择“不指定”PersistVolume。
> - 如果选择的 PV 状态为 Released，还需手动删除该 PV 对应 YAML 配置文件中的 claimRef 字段，该 PV 才能顺利与 PVC 绑定。详情请参见 查看 PV 和 PVC 的绑定规则。

3. 选择**创建 PersistentVolumeClaim**，即可完成创建。

创建 Workload 使用 PVC 数据卷

> 说明：
> 该步骤以创建工作负载 Deployment 为例。

1. 在“集群管理”页面，选择目标集群 ID，进入待部署 Workload 的集群的 “Deployment” 页面。
2. 单击**新建**，进入“新建 Workload” 页面，参考 创建 Deployment 进行创建，并参考以下信息进行数据卷挂载。如下图所示：



- **数据卷（选填）**：
  - **挂载方式**：选择“使用已有 PVC”。
  - **数据卷名称**：自定义，本文以 cfs-vol 为例。
  - **选择 PVC**：选择在步骤 创建 PVC 中已创建的 “cfs-pvc”。
- **实例内容器**：单击**添加挂载点**，进行挂载点设置。
  - **数据卷**：选择该步骤中已添加的数据卷“cfs-vol”。
  - **目标路径**：填写目标路径，本文以 /cache 为例。
  - **挂载子路径**：仅挂载选中数据卷中的子路径或单一文件。例如， /data 或 /test.txt。
3. 单击**创建 Workload**，完成创建。

> 注意：
> 如使用 CFS 的 PVC 挂载模式，数据卷支持挂载到多台 Node 主机上。

Kubectl 操作指引

创建 PV

```
apiVersion: v1
kind: PersistentVolume
metadata:
name: cfs
spec:
accessModes:
- ReadWriteMany
capacity:
storage: 10Gi
csi:
driver: com.tencent.cloud.csi.cfs
volumeAttributes:
fsid: XXXXXX
host: 192.168.XX.XX
path: /
vers: "3"
volumeHandle: cfs
persistentVolumeReclaimPolicy: Retain
storageClassName: XXX
volumeMode: Filesystem
```

参数	是否可选	描述
fsid	是	文件系统 fsid（非文件系统 id），可在文件系统挂载点信息中查看。
host	是	文件系统 ip 地址，可在文件系统挂载点信息中查看。
path	是	文件系统子目录，挂载后 workload 将无法访问到该子目录的上层目录。
vers	是	插件连接文件系统时所使用的协议版本，目前支持的版本有 "3" 和 "4"。

说明：

如果您在静态 PV 的 YAML 中指定协议版本为 vers: "3"**，则还需要指定待挂载文件系统的 fsid 参数（获取方式请参考** 获取文件系统 fsid**），否则会存在挂载失败的情况；** vers: "4" **则无需指定 fsid。**

使用云硬盘 CBS
云硬盘使用说明

最近更新时间：2022-01-07 15:41:58

操作场景

腾讯云容器服务支持通过创建 PV/PVC，并为工作负载挂载数据卷的方式使用云硬盘 CBS。本文介绍如何通过以下两种方式在集群中为工作负载挂载云硬盘：

> 说明：
> 通过 PV 和 PVC 使用云硬盘 CBS 时，一个云硬盘仅支持创建一个 PV，同时只能被一个集群节点挂载。

操作步骤

动态创建云硬盘

动态创建云硬盘时，通常包含以下几个步骤：

1. 创建云硬盘类型的 StorageClass，定义需使用的云硬盘模板。

   > 说明：
   > - 容器服务默认提供名称为 cbs 的 StorageClass。配置为：高性能云硬盘、随机选择可用区、按量计费。
   > - 您可按需自行定义 StorageClass。

2. 通过 StorageClass 创建 PVC， 进一步定义所需的云硬盘参数。
3. 创建工作负载数据卷时选择已创建的 PVC，并设置容器挂载点。
   详细操作步骤请参见 StorageClass 管理云硬盘模板。

使用已有的云硬盘

可通过以下步骤使用已有云硬盘：

1. 使用已有云硬盘创建 PV。
2. 创建 PVC 时，设置与已有 PV 相同的 StorageClass 和容量。
3. 创建工作负载时，选择 PVC。
   详细操作步骤请参见 PV 和 PVC 管理云硬盘。

StorageClass 管理云硬盘模板

最近更新时间：2021-12-31 16:37:30

操作场景

集群管理员可使用 StorageClass 为容器服务集群定义不同的存储类型。容器服务已默认提供块存储类型的 StorageClass，您可通过 StorageClass 配合 PersistentVolumeClaim 动态创建需要的存储资源。

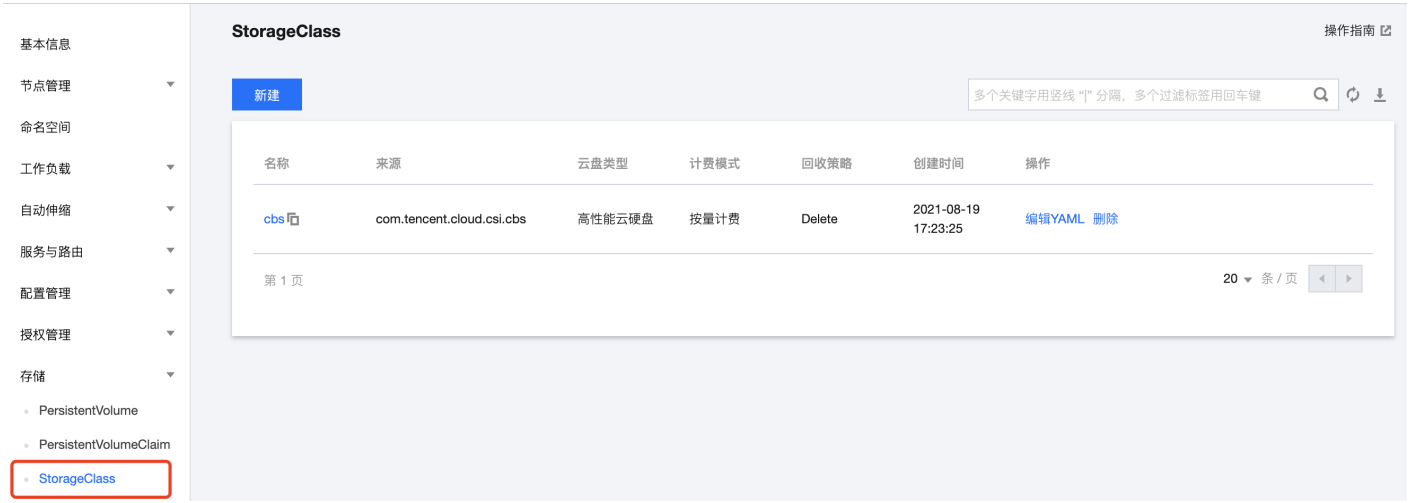
本文介绍通过控制台、Kubectl 两种方式创建云硬盘 CBS 类型的 StorageClass，自定义云硬盘使用所需的模板。

操作步骤

控制台操作指引

创建 StorageClass

1. 登录 容器服务控制台 ，选择左侧栏中的**集群**。
2. 在“集群管理”页中，单击需创建 StorageClass 的集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**存储** > **StorageClass**。如下图所示：



4. 单击**新建**进入“新建StorageClass” 页面，参考以下信息进行创建。如下图所示：

主要参数信息如下：

- **名称**：自定义，本文以 cbs-test 为例。
- **Provisioner**：选择**云硬盘CBS**。
- **地域**：当前集群所在地域。
- **可用区**：表示当前地域下支持使用云硬盘的可用区，请按需选择。
- **计费模式**：提供**按量计费**和**包年包月**两种计费模式，不同计费模式所支持的回收策略不同，请参考以下信息进行选择：
  - **按量计费**：一种弹性计费模式，支持随时开通/销毁实例，按实例的实际使用量付费。支持删除和保留的回收策略。
  - **包年包月**：一种预付费模式，提前一次性支付一个月的存储费用，支持按月自动续费。仅支持保留的回收策略。

> 说明
> - 如需购买包年包月云硬盘，则需前往 角色 页面，为 TKE_QCSRole 角色添加策略 QcloudCVMFinanceAccess 配置支付权限，否则可能会因支付权限问题导致创建基于包年包月 StorageClass 的 PVC 失败。
> - 仅计费模式为包年包月的云硬盘可执行续费操作，自动续费功能默认按月续费。用户可前往所创建的PVC详情页，打开/关闭自动续费功能。更多计费信息参见 云硬盘计费问题。

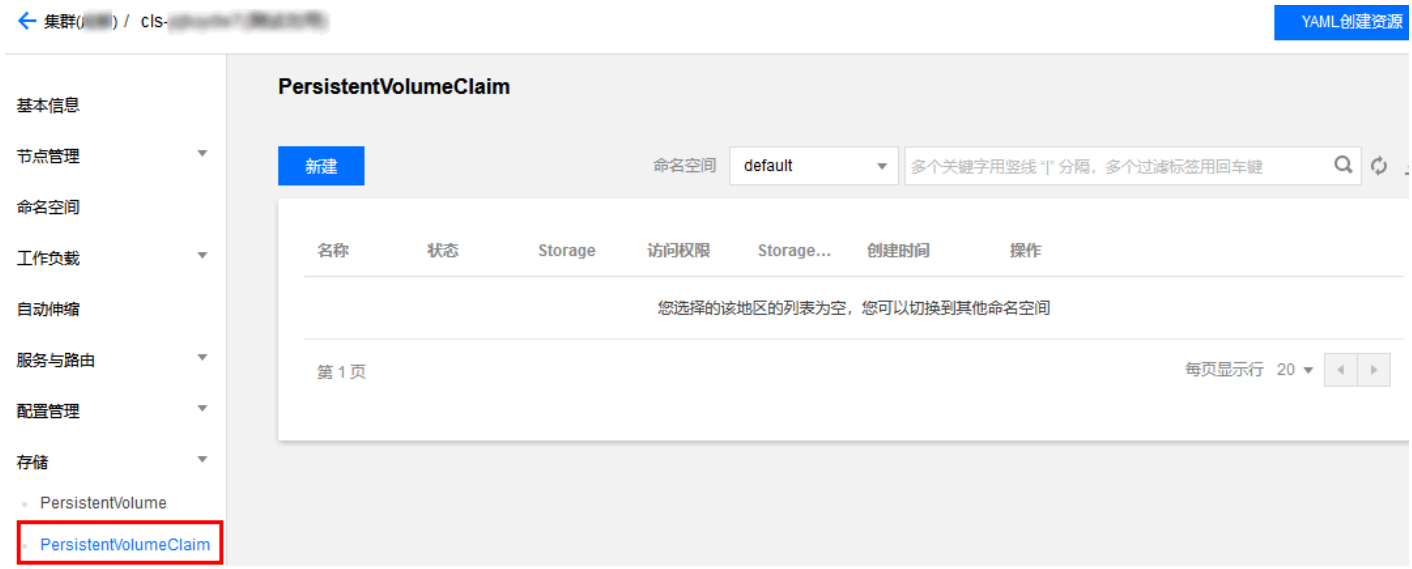
- **云盘类型**：通常提供**高性能云硬盘**、**SSD云硬盘**和**增强型SSD云硬盘**三种类型，不同可用区下提供情况有一定差异，详情请参见 云硬盘类型说明 并结合控制台提示进行选择。
- **回收策略**：云盘的回收策略，通常提供**删除**和**保留**两种回收策略，具体选择情况与所选计费模式相关。出于数据安全考虑，推荐使用保留回收策略。
- **卷绑定模式**：提供**立即绑定**和**等待调度**两种卷绑定模式，不同模式所支持的卷绑定策略不同，请参考以下信息进行选择：
  - **立即绑定**：通过该 storageclass 创建的 PVC 将直接进行 PV 的绑定和分配。
  - **等待调度**：通过该 storageclass 创建的 PVC 将延迟与 PV 的绑定和分配，直到使用该 PVC 的 Pod 被创建。
- **定期备份**：设置定期备份可有效保护数据安全，备份数据将产生额外费用，详情请见 快照概述。

> 说明：
> 容器服务默认提供的 default-policy 备份策略的配置包括：执行备份的日期、执行备份的时间点和备份保留的时长。

5. 单击**新建StorageClass**即可完成创建。

使用指定 StorageClass 创建 PVC

1. 在“集群管理”页面，选择需创建 PVC 的集群 ID。
2. 在集群详情页面，选择左侧菜单栏中的**存储** > **PersistentVolumeClaim**，进入 “PersistentVolumeClaim” 信息页面。如下图所示：



3. 单击**新建**进入“新建PersistentVolumeClaim” 页面，参考以下信息设置 PVC 关键参数。如下图所示：

主要参数信息如下：

- **名称**：自定义，本文以 cbs-pvc 为例。
- **命名空间**：选择 “default”。
- **Provisioner**：选择**云硬盘CBS**。
- **读写权限**：云硬盘仅支持单机读写。
- **StorageClass**：按需指定 StorageClass，本文选择已在 创建 StorageClass 步骤中创建的 cbs-test 为例。

> **说明：**
>
> - PVC 和 PV 会绑定在同一个 StorageClass 下。
> - 不指定 StorageClass 意味着该 PVC 对应的 StorageClass 取值为空，对应 YAML 文件中的 storageClassName 字段取值为空字符串。

- **PersistVolume**：按需指定 PersistentVolume，本文以不指定 PersistentVolume 为例。

> 说明：
> - 系统首先会筛选当前集群内是否存在符合绑定规则的 PV，如果没有则根据 PVC 和所选 StorageClass 的参数动态创建 PV 与之绑定。
> - 系统不允许在不指定 StorageClass 的情况下同时选择不指定 PersistVolume。
> - 不指定 PersistentVolume。详情请参见 查看 PV 和 PVC 的绑定规则。

- **云盘类型**：根据所选的 StorageClass 展示所选的云盘类型为**高性能云硬盘、SSD云硬盘和增强型SSD云硬盘**。
- **容量**：在不指定 PersistentVolume 时，需提供期望的云硬盘容量（云硬盘大小必须为10的倍数。高性能云硬盘最小为10GB；SSD 和增强型 SSD 云硬盘最小为20GB）。
- **费用**：根据上述参数计算创建对应云盘的所需费用，详情参考 计费模式。

4. 单击**创建PersistentVolumeClaim**，即可完成创建。

创建 StatefulSet 挂载 PVC 类型数据卷

> 说明：
> 该步骤以创建工作负载 StatefulSet 为例。

1. 在目标集群详情页，选择左侧菜单栏中的**工作负载** > **StatefulSet**，进入 “StatefulSet” 页面。
2. 单击**新建**进入“新建Workload” 页面，参考 创建 StatefulSet 进行创建，并参考以下信息进行数据卷挂载。如下图所示：

- **数据卷（选填）**：
  - **挂载方式**：选择“使用已有PVC”。
  - **数据卷名称**：自定义，本文以 cbs-vol 为例。
  - **选择 PVC**：选择已有 PVC，本文以选择在 使用指定 StorageClass 创建 PVC 步骤中创建的 cbs-pvc 为例。
- **实例内容器**：单击**添加挂载点**，进行挂载点设置。
  - **数据卷**：选择该步骤中已添加的数据卷 “cbs-vol”。
  - **目标路径**：填写目标路径，本文以 /cache 为例。
  - **挂载子路径**：仅挂载选中数据卷中的子路径或单一文件。例如，/data 或 /test.txt。

3. 单击**创建Workload**，即可完成创建。

> 注意：
> 如使用 CBS 的 PVC 挂载模式，则数据卷只能挂载到一台 Node 主机上。

Kubectl 操作指引

您可参考本文提供的示例模板，使用 Kubectl 进行 StorageClass 创建操作。

创建 StorageClass

以下 YAML 文件示例为集群内默认存在 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_PREMIUM,CLOUD_SSD,CLOUD_HSSD 如果不识别则当做 CLOUD_PREMIUM
# renewflag: NOTIFY_AND_AUTO_RENEW
# renewflag为云硬盘的续费模式，NOTIFY_AND_AUTO_RENEW模式支持通知过期且按月自动续费，NOTIFY_AND_MANUAL_RENEW模式支持通知过期但不支持自动续费，DISABLE_NOTIFY_AND_MANUAL_RENEW模式支持不通知过期也不自动续费。不指定该字段则默认为NOTIFY_AND_MANUAL_RENEW模式。
# 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 POSTPAID（按量计费：支持 Retain 保留和 Delete 删除策略，Retain 仅在高于1.8的集群版本生效）
# aspid: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
```

支持参数如下表：

参数	描述
type	包括 CLOUD_PREMIUM（高性能云硬盘）和 CLOUD_SSD（SSD 云硬盘）、CLOUD_HSSD（增强型 SSD 云硬盘）。
zone	用于指定可用区。如果指定，则云硬盘将创建到此可用区。如果不指定，则拉取所有 Node 的可用区信息，进行随机选取。 腾讯云各地域标识符请参见 地域和可用区。
paymode	云硬盘的计费模式，默认设置为 POSTPAID 模式，即按量计费，支持 Retain 保留和 Delete 删除策略，Retain 仅在高于1.8的集群版本生效。还可设置为 PREPAID 模式，即包年包月，仅支持 Retain 保留策略。
renewflag	云硬盘的续费模式。默认为 NOTIFY_AND_MANUAL_RENEW 模式。 • NOTIFY_AND_AUTO_RENEW 模式代表所创建的云硬盘支持通知过期且按月自动续费。 • NOTIFY_AND_MANUAL_RENEW 模式代表所创建的云硬盘支持通知过期但不自动续费。 • DISABLE_NOTIFY_AND_MANUAL_RENEW 模式则代表所创建的云硬盘不通知过期也不自动续费。
aspid	指定快照 ID，创建云硬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

使用云硬盘创建多实例 StatefulSet，YAML 文件示例如下：

> **说明**
> 资源对象的 apiVersion 可能因为您集群的 Kubernetes 版本不同而不同，您可通过 kubectl api-versions 命令查看当前资源对象的 apiVersion。

```
apiVersion: apps/v1
kind: StatefulSet
```

```
metadata:
name: web
spec:
selector:
matchLabels:
app: nginx
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
```

PV 和 PVC 管理云硬盘

最近更新时间：2021-12-31 16:37:03

操作场景

腾讯云容器服务支持通过创建 PV/PVC，并在创建工作负载添加数据卷时使用已有 PVC，实现通过 PV 和 PVC 管理云硬盘。本文介绍如何通过控制台、Kubectl 两种方式实现 PV 和 PVC 管理云硬盘。

> 注意：
>
> - 云硬盘不支持跨可用区挂载。若挂载云硬盘类型 PV 的 Pod 迁移至其他可用区，将会导致挂载失败。
> - 容器服务控制台不支持云硬盘扩容，可前往 云硬盘控制台 进行扩容操作。详情请参见 扩容云硬盘。

操作步骤

控制台操作指引

通过控制台创建 StorageClass

由于静态创建云硬盘类型的 PV 时，需要绑定同类型可用 StorageClass，请参考 创建 StorageClass 完成创建。

静态创建 PV

> 说明：
> 静态创建 PV 适用于已有存量云盘，并在集群内使用的场景。

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 选择需创建 PV 的集群 ID，进入该集群详情页面。
3. 选择左侧菜单栏中的**存储** > **PersistentVolume**，进入 “PersistentVolume” 页面。如下图所示：

4. 选择**新建**进入“新建PersistentVolume” 页面，参考以下信息进行创建。如下图所示：

主要参数信息如下：

- **来源设置**：选择**静态创建**。
- **名称**：自定义，本文以 cbs-pv 为例。
- **Provisioner**：选择**云硬盘CBS**。
- **读写权限**：云硬盘仅支持单机读写。
- **StorageClass**：按需选择合适的 StorageClass。本文以选择在 通过控制台创建 StorageClass 步骤中创建的 cbs-test 为例。

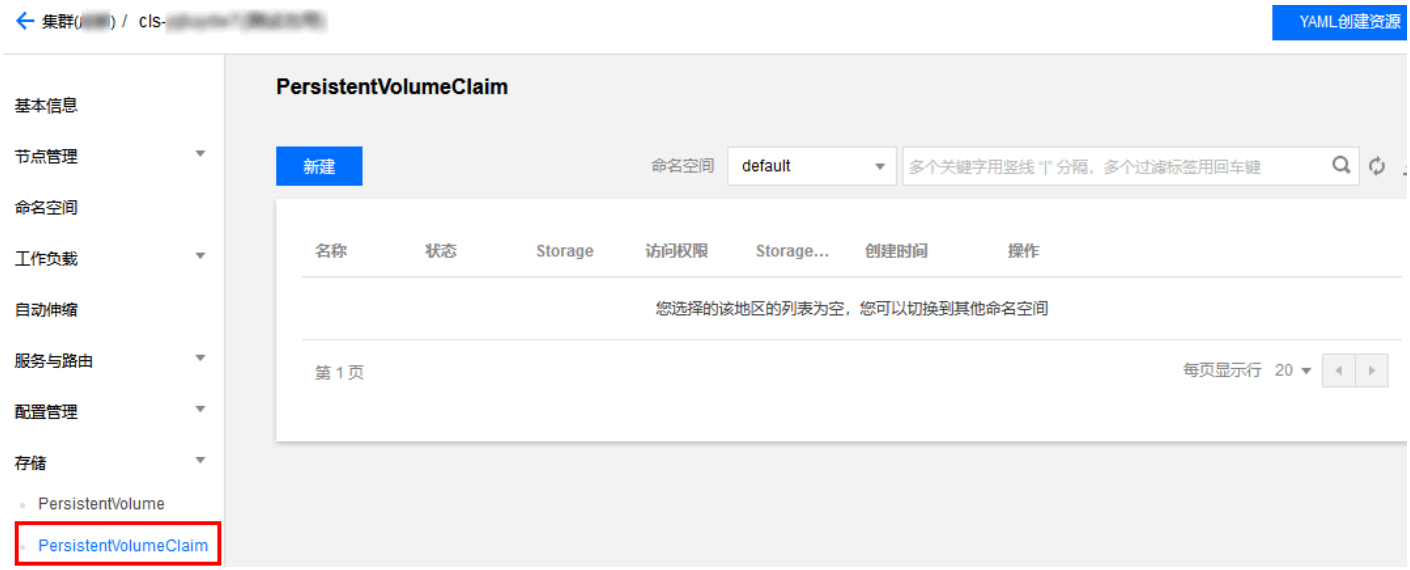
> 说明：
>
> - PVC 和 PV 会绑定在同一个 StorageClass 下。
> - 不指定意味着该 PV 对应的 StorageClass 取值为空，对应 YAML 文件中的 storageClassName 字段取值为空字符串。

- **云盘**：选择已经创建好的云硬盘。
- **文件系统**：默认为 ext4。

5. 单击**创建PersistentVolume**即可完成创建。

创建 PVC

1. 在集群详情页，选择左侧菜单栏中的**存储** > **PersistentVolumeClaim**，进入 “PersistentVolumeClaim” 页面。如下图所示：



2. 选择**新建**进入 “新建PersistentVolumeClaim” 页面，参考以下信息进行创建。如下图所示：

主要参数信息如下：

- **名称**：自定义，本文以 cbs-pvc 为例。
- **命名空间**：选择 “default”。
- **Provisioner**：选择**云硬盘CBS**。
- **读写权限**：云硬盘只支持单机读写。
- **StorageClass**：按需选择合适的 StorageClass。本文以选择在 通过控制台创建 StorageClass 步骤中创建的 cbs-test 为例。

> **说明：**
>
> - PVC 和 PV 会绑定在同一个 StorageClass 下。
> - 不指定意味着该 PVC 对应的 StorageClass 取值为空，对应 YAML 文件中的 storageClassName 字段取值为空字符串。

- **PersistVolume**：按需指定 PersistentVolume，本文选择以在 静态创建PV 步骤中创建的 cbs-pv 为例。

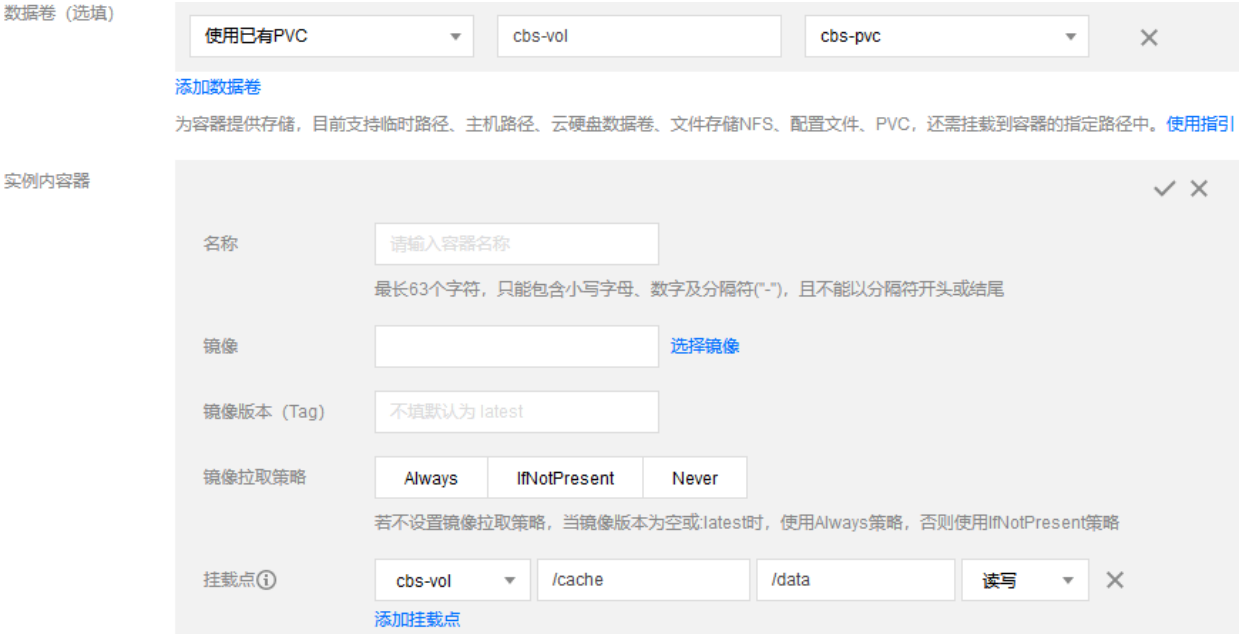
> 说明：
>
> - 只有与指定的 StorageClass 相同并且状态为 Available 和 Released 的 PV 为可选状态，如果当前集群内没有满足条件的 PV 可选，请选择“不指定”PersistVolume。
> - 如果选择的 PV 状态为 Released，还需手动删除该 PV 对应 YAML 配置文件中的 claimRef 字段，该 PV 才能顺利与 PVC 绑定。详情请参见 查看 PV 和 PVC 的绑定规则。

3. 单击**创建PersistentVolumeClaim**，即可完成创建。

创建 Workload 使用 PVC 数据卷

> 说明：
> 该步骤以创建工作负载 Deployment 为例。

1. 在“集群管理”页面，选择目标集群 ID，进入待部署 Workload 的集群的 “Deployment” 页面。
2. 单击**新建**，进入“新建Workload” 页面，参考 创建 Deployment 进行创建，并参考以下信息进行数据卷挂载。如下图所示：



- **数据卷（选填）**：
  - **挂载方式**：选择“使用已有PVC”。
  - **数据卷名称**：自定义，本文以 cbs-vol 为例。
  - **选择 PVC**：选择在步骤 创建 PVC 中已创建的 “cbs-pvc”。
- **实例内容器**：单击**添加挂载点**，进行挂载点设置。
  - **数据卷**：选择该步骤中已添加的数据卷 “cbs-vol”。
  - **目标路径**：填写目标路径，本文以 /cache 为例。
  - **挂载子路径**：仅挂载选中数据卷中的子路径或单一文件。例如，/data 或 /test.txt。

3. 单击**创建Workload**即可完成创建。

> 注意：
> 如使用 CBS 的 PVC 挂载模式，则数据卷只能挂载到一台 Node 主机上。

Kubectl 操作指引

您可通过以下 YAML 示例文件，使用 Kubectl 进行创建操作。

（可选）创建 PV

可以通过已有云硬盘创建 PV，也可以直接 创建 PVC ，系统将自动创建对应的 PV。YAML 文件示例如下：

```
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，则在创建 PVC 时，系统将自动创建对应的 PV。YAML 文件示例如下：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: nginx-pv-claim
spec:
storageClassName: cbs
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 10Gi
```

- 云硬盘大小必须为10的倍数。
- 高性能云硬盘最小为10GB，SSD 和增强型 SSD 云硬盘最小为20GB，详情见 创建云硬盘。

使用 PVC

可通过创建 Workload 使用 PVC 数据卷。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
```

其他存储卷使用说明

最近更新时间：2022-03-18 11:49:13

简介

数据卷类型

数据卷类型	描述
使用临时路径	/
使用主机路径	将容器所在宿主机的文件目录挂载到容器的指定路径中（即对应 Kubernetes 的 HostPath）。您可以根据业务需求，不设置源路径（即对应 Kubernetes 的 EmptyDir）。如果不设置源路径，系统将分配主机的临时目录挂载到容器的挂载点。**指定源路径的本地硬盘数据卷适用于将数据持久化存储到容器所在宿主机，EmptyDir 适用于容器的临时存储。**
使用 NFS 盘	只需填写 NFS 路径，您可以使用腾讯云的 文件存储 CFS，也可使用自建的文件存储 NFS。**使用 NFS 数据卷适用于多读多写的持久化存储，也适用于大数据分析、媒体处理、内容管理等场景。**
使用已有 PersistentVolumeClaim	使用已有 PersistentVolumeClaim 声明工作负载的存储，自动分配或新建 PersistentVolume 挂载到对应的 Pod 下。主要适用于 StatefulSet 创建的有状态应用。
使用 ConfigMap	ConfigMap 以文件系统的形式挂载到 Pod 上，支持自定义 ConfigMap 条目挂载到特定的路径。更多详情请参见 ConfigMap 管理。
使用 Secret	Secret 以文件系统的形式挂载到 Pod 上，支持自定义 Secret 条目挂载到特定的路径。更多详情请参见 Secret 管理。

数据卷的注意事项

- **创建数据卷后，需在实例内容器模块设置容器的挂载点。**
- 同一个服务下，数据卷的名称和容器设置的挂载点不能重复。
- 本地硬盘数据卷源路径为空时，系统将分配 /var/lib/kubelet/pods/pod_name/volumes/kubernetes.io~empty-dir **临时目录，且使用临时的数据卷生命周期与实例的生命周期保持一致。**
- 数据卷挂载未设置权限，默认设置为读写权限。

Volume 控制台操作指引

创建工作负载挂载数据卷

1. 登录容器服务控制台，并选择左侧导航栏中的 **集群**。
2. 在“集群管理”页面，单击需要部署 Workload 的集群 ID，进入待部署 Workload 的集群管理页面。
3. 在**工作负载**下，任意选择 Workload 类型，进入对应的信息页面。
   例如，选择**工作负载** > **DaemonSet**，进入 DaemonSet 信息页面。如下图所示：

4. 单击**新建**，进入 “新建Workload” 页面。
5. 根据页面信息，设置工作负载名、命名空间等信息。并在“数据卷”中，单击**添加数据卷**添加数据卷。
6. 根据实际需求，选择数据卷的存储方式，本文以**使用腾讯云硬盘**为例。
7. 在“实例内容器”的**挂载点**配置挂载点。如下图所示：
   在 步骤5 中选择**添加数据卷**后，才可进行挂载点配置。

8. 其余选项请按需设置，并单击**创建Workload**即可完成创建。

各类数据卷挂载配置

该表展示了不同数据卷的使用细节，**当您在创建工作负载时，并选择“添加数据卷”后**，可参照以下内容进行数据卷的添加以及挂载点的设置：

数据卷			挂载点		
类型	名称	其他	目标路径	挂载子路径	读写权限
临时路径	自定义	/	请根据实际需求进行填写，示例如 /cache。	仅挂载选中数据卷中的子路径或单一文件，示例如 /data 或 /test.txt。	请根据实际需求进行选择。 • 只读：只允许读取该容器路径数据卷，数据修改只允许在宿主机上操作。 • 读写：允许读取以及将修改保存到该容器路径数据卷。
主机路径		设置主机路径。 • 主机路径：该路径不能为空，例如当该容器需要访问 Docker 时主机路径可设置为/var/lib/docker。 • 检查类型：TKE 为您提供 NoChecks、DirectoryOrCreate 等多种检查类型，请仔细查阅控制台上每种类型介绍，并根据实际需求进行选择。			
NFS 盘		NFS 路径：填写文件系统 CFS 或自建 NFS 地址。 • 如需创建文件系统，请参看创建文件系统及挂载点。 • NFS 路径示例如10.0.0.161:/。该路径可登录 文件系统控制台，单击目标文件系统 ID ，在**挂载点信息**页签的“Linux 下挂载目录”中获取。			
已有 PVC		请选择 PVC：根据实际需求进行选择。			
ConfigMap		• 选择 ConfigMap：根据实际需求进行。 • 选项：提供“全部”和“指定部分Key”两种选择。 • Items：当选择“指定部分Key” 选项时，可以通过添加 item 向特定路径挂载，如挂载点是 /data/config，子路径是 dev，最终会存储在 /data/config/dev 下。			
Secret					

Kubectl 操作 Volume 指引

仅提供示例文件，您可直接通过 Kubectl 进行创建操作。

Pod 挂载 Volume YAML 示例

```
apiVersion: v1
kind: Pod
metadata:
```

```
name: test-pd
spec:
containers:
- image: k8s.gcr.io/test-webserver
name: test-container
volumeMounts:
- mountPath: /cache
name: cache-volume
volumes:
- name: cache-volume
emptyDir: {}
```

- **spec.volumes**：设置数据卷名称、类型、数据卷的参数。
  - **spec.volumes.emptyDir**：设置临时路径。
  - **spec.volumes.hostPath**：设置主机路径。
  - **spec.volumes.nfs**：设置 NFS 盘。
  - **spec.volumes.persistentVolumeClaim**：设置已有 PersistentVolumeClaim
- **spec.volumeClaimTemplates**：若使用该声明，将根据内容自动创建 PersistentVolumeClaim 和 PersistentVolume。
- **spec.containers.volumeMounts**：填写数据卷的挂载点。

PV 和 PVC 的绑定规则

最近更新时间：2022-01-19 14:37:20

PV 状态介绍

PV 状态	描述
Avaliable	创建好的 PV 在没有和 PVC 绑定的时候处于 Available 状态。
Bound	当一个 PVC 与 PV 绑定之后，PVC 就会进入 Bound 的状态。
Released	一个回收策略为 Retain 的 PV，当其绑定的 PVC 被删除，该 PV 会由 Bound 状态转变为 Released 状态。 **注意：**Released 状态的 PV 需要手动删除 YAML 配置文件中的 claimRef 字段才能与 PVC 成功绑定。

PVC 状态介绍

PVC 状态	描述
Pending	没有满足条件的 PV 能与 PVC 绑定时，PVC 将处于 Pending 状态。
Bound	当一个 PV 与 PVC 绑定之后，PVC 会进入 Bound 的状态。

绑定规则

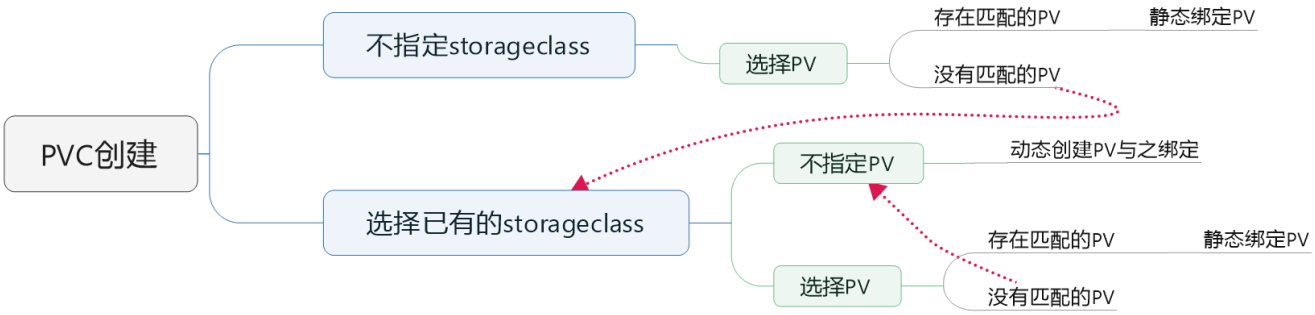
当 PVC 绑定 PV 时，需考虑以下参数来筛选当前集群内是否存在满足条件的 PV。

参数	描述
VolumeMode	主要定义 volume 是文件系统（FileSystem）类型还是块（Block）类型，PV 与 PVC 的 VolumeMode 标签必须相匹配。
Storageclass	PV 与 PVC 的 storageclass 类名必须相同（或同时为空）。
AccessMode	主要定义 volume 的访问模式，PV 与 PVC 的 AccessMode 必须相同。
Size	主要定义 volume 的存储容量，PVC 中声明的容量必须小于等于 PV，如果存在多个满足条件的 PV，则选择最小的 PV 与 PVC 绑定。

> **说明：**
> PVC 创建后，系统会根据上述参数筛选满足条件的 PV 进行绑定。如果当前集群内的 PV 资源不足，系统会动态创建一个满足绑定条件的 PV 与 PVC 进行绑定。

StorageClass 的选择和 PV/PVC 的绑定关系

容器服务 TKE 的平台操作中，StorageClass 的选择与 PV/PVC 之间的绑定关系见下图：



组件管理
扩展组件概述

最近更新时间：2022-02-28 17:15:32

扩展组件是腾讯云容器服务 TKE 提供的扩展功能包，您可以根据业务诉求选择部署所需的扩展组件。扩展组件可帮助您管理集群的 Kubernetes 组件，包括组件部署、升级、更新配置和卸载等。

扩展组件类型

扩展组件分为基础组件和增强组件两种类型。

基础组件

基础组件是 TKE 功能依赖的软件包。例如，负载均衡组件 Service-controller、CLB-ingress-controller 及容器网络插件 tke-cni-agent 等。

> 说明：
>
> - 基础组件的升级、配置管理将由 TKE 统一进行管理维护，不建议您修改基础组件。
> - 基础组件的更新发布动态将通过邮件、短信等形式进行通知。

增强组件

增强组件是 TKE 提供的非必需部署的组件，您可以通过部署增强组件来使用 TKE 支持的增强功能，增强组件类型如下表所示：

组件名称	使用场景	组件介绍
OOMGuard（内存溢出守护）	监控	该组件在用户态降低了由于 cgroup 内存回收失败而产生的各种内核故障的发生几率。
NodeProblemDetectorPlus（节点异常检测 Plus）	监控	该组件可以实时检测节点上的各种异常情况，并将检测结果报告给 kube-apiserver。
NodeLocalDNSCache（本地 DNS 缓存组件）	DNS	该组件通过在集群节点上作为 DaemonSet 运行 DNS 缓存代理来提高集群 DNS 性能。
DNSAutoscaler（DNS 水平伸缩组件）	DNS	该组件通过 deployment 获取集群的节点数和核数，并可以根据预设的伸缩策略，自动水平伸缩 DNS 的副本数。
COS-CSI（腾讯云对象存储）	存储	该组件实现了 CSI 接口，可帮助容器集群使用腾讯云对象存储。
CFS-CSI（腾讯云文件存储）	存储	该组件实现了 CSI 接口，可帮助容器集群使用腾讯云文件存储。
CBS-CSI（腾讯云硬盘存储）	存储	该组件实现了 CSI 接口，支持 TKE 集群通过控制台快捷选择存储类型，并创建对应块存储云硬盘类型的 PV 和 PVC。
TCR（容器镜像服务插件）	镜像	该组件自动为集群配置指定 TCR 实例的域名内网解析及集群专属访问凭证，可用于内网，免密拉取容器镜像。
P2P（容器镜像加速分发）	镜像	该组件基于 P2P 技术，可应用于大规模 TKE 集群快速拉取 GB 级容器镜像，支持上千节点的并发拉取。
Dynamic Scheduler（动态调度组件）	调度	Dynamic Scheduler 是容器服务 TKE 基于 Kubernetes 原生 Kube-scheduler Extender 机制实现的动态调度器插件，可基于 Node 真实负载进行预选和优选。安装该组件后可以有效避免原生调度器基于 request 和 limit 调度机制带来的节点负载不均问题。
Descheduler（重调度组件）	调度	在 TKE 集群中安装该插件后，该插件会和 Kube-scheduler 协同生效，实时监控集群中高负载节点并驱逐低优先级 Pod。建议您搭配 TKE Dynamic Scheduler（动态调度器扩展组件）一起使用，多维度保障集群负载均衡。

组件名称	使用场景	组件介绍
NetworkPolicy Controller（网络策略控制器组件）	其他	Network Policy 是 Kubernetes 提供的一种资源，本组件提供了针对该资源的 Controller 实现。
Nginx-Ingress（社区 Ingress 组件）	其他	Nginx 可以用作反向代理、负载平衡器和 HTTP 缓存。Nginx-ingress 组件是使用 Nginx 作为反向代理和负载平衡器的 Kubernetes 的 Ingress 控制器。
OLM（Operator 生命周期管理）	其他	OLM（Operator Lifecycle Manager）作为 Operator Framework 的一部分，可以帮助用户进行 Operator 的自动安装，升级及生命周期的管理。
HPC（定时修改副本数）	其他	HPC（HorizontalPodCronscaler）是一种可以对 K8S workload 副本数进行定时修改的自研组件，配合 HPC CRD 使用，最小支持秒级的定时任务。

组件版本维护说明

最近更新时间：2022-07-01 14:39:48

版本更新说明

腾讯云容器服务提供了涵盖网络、存储、监控、镜像、调度、GPU相关场景下的增强组件来扩展集群功能，您可在容器集群详情中的组件管理页面查看当前组件版本，并支持对组件版本进行手动升级操作。

升级须知

1. 升级属于不可逆操作。
2. 仅支持向上升级容器服务提供的组件版本，在满足集群 kubernetes 版本限制的情况下，默认升级至当前最新版本。
3. **针对已废弃的组件版本，容器团队将不再提供技术支持**，建议您及时升级。

版本迭代记录

2022年6月

组件名称	发布时间	版本号	变更内容	限制和影响
DeScheduler（重调度器插件）	2022-06-07	v1.0.1	TMP 认证支持：prom-probe 中添加 auth 认证、descheduler 和 init container 中传入 token/appid 等环境变量，并进行解码、descheduler 中增加 prometheus client 认证功能。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。
qGPU（GPU 隔离组件）	2022-06-08	v1.0.3	• qgpu manager 镜像更新为 tkeimages/elastic-gpu-agent:v1.0.2。 • qgpu scheduler 镜像更新为 tkeimages/elastic-gpu-scheduler:v1.0.2。 • 支持使用 GPU CRD 管理 GPU 资源。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。

2022年5月

组件名称	发布时间	版本号	变更内容	限制和影响
CBS-CSI（腾讯云硬盘存储）	2022-05-06	v1.0.3	• 插件支持配置污点容忍。 • 插件新增 type 启动参数	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。
COS-CSI（腾讯云对象存储）	2022-05-06	v1.0.1	插件支持配置污点容忍。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。
CFS-CSI（腾讯云文件存储）	2022-05-06	v1.0.4	插件 umount 幂等性支持。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。
CFS-CSI（腾讯云文件存储）	2022-05-24	v1.0.5	支持 EKS cfs provisoner。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。
CBS-CSI（腾讯云硬盘存储）	2022-05-31	v1.0.4	• 优化插件启动逻辑。 • csi-attacher 默认并发数调大至 50。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。

2022年4月

组件名称	发布时间	版本号	变更内容	限制和影响
CFS-CSI（腾讯云文件存储）	2022-04-12	v1.0.2	插件 umount 幂等性支持。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。
CFS-CSI（腾讯云文件存储）	2022-04-19	v1.0.3	• tcfs crd 增加资源标签字段。 • 1.12 及以下 k8s 版本不安装 tcfs 相关的资源。 • cfs-csi startServer 注册启动优化。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。
qGPU（GPU 隔离组件）	2022-04-21	v1.0.2	• 更新了 qgpu manager 镜像版本，支持自动所在节点设置 gpu 驱动版本以及其他信息。 • 更新了 clusterrole qgpu-manager，增加了对 nodes 的操作权限。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。
CBS-CSI（腾讯云硬盘存储）	2022-04-24	v1.0.2	• 取消插件 NodeUnpublishVolume 接口中的目录清理逻辑。 • 插件支持通过 Serial 获取盘符。 • 插件删除时保留对应 crd 资源。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。

2022年3月

组件名称	发布时间	版本号	变更内容	限制和影响
CBS-CSI（腾讯云硬盘存储）	2022-03-16	v1.0.1	支持使用了 intree cbs 的业务负载在集群从 1.18 升级到 1.20 时原地无损迁移到 csi。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。
CFS-CSI（腾讯云文件存储）	2022-03-24	v1.0.1	支持动态创建时共享存储实例，通过自动生成的子目录进行数据隔离。	此次升级不会对已有业务造成影响，升级过程中可能存在组件不可用情况，建议业务低峰期升级。

组件的生命周期管理

最近更新时间：2022-07-01 14:40:01

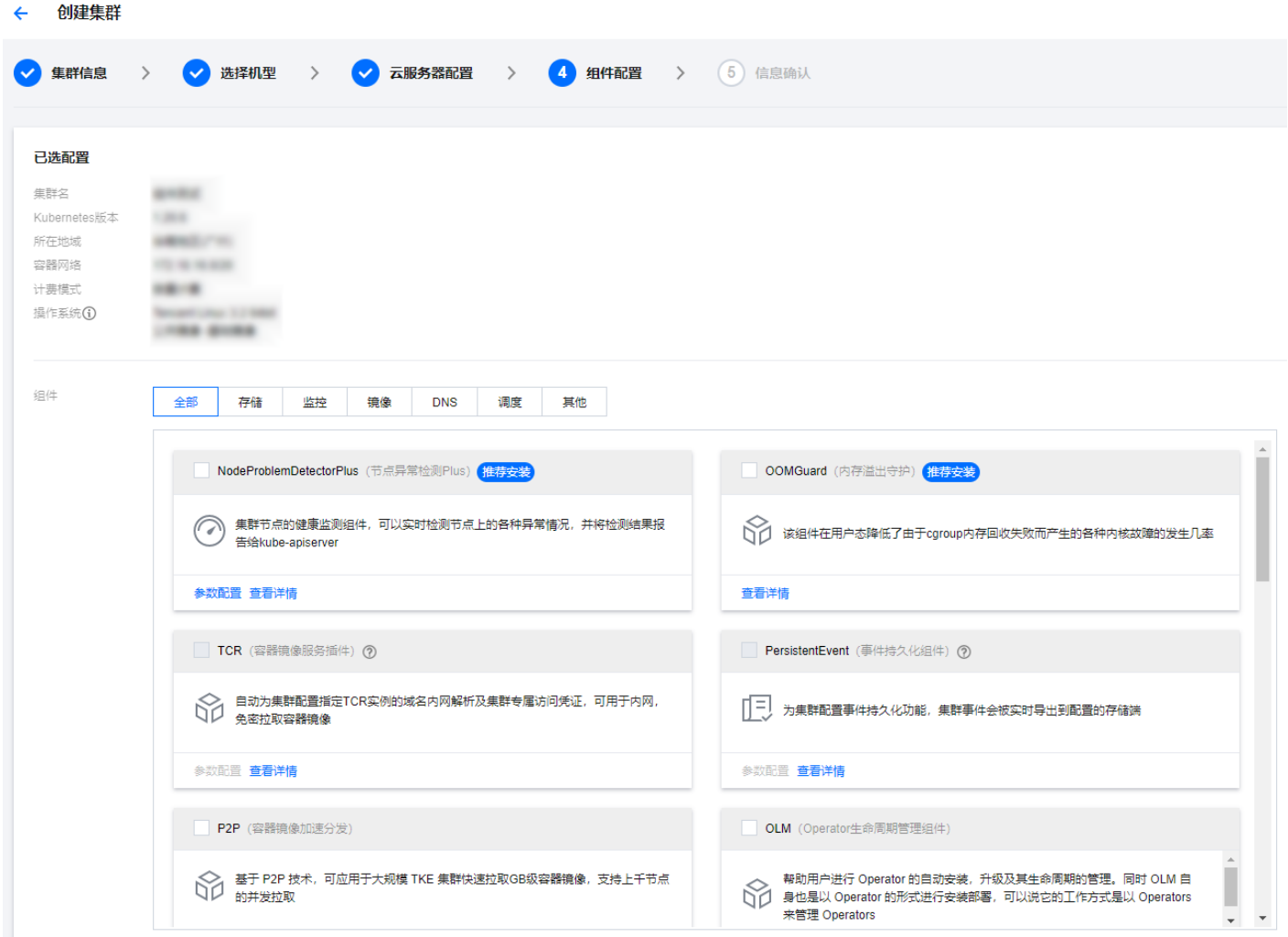
组件安装

您可以通过以下两种方式安装增强组件：

- 通过集群创建页安装
- 通过组件管理页安装

通过集群创建页安装

1. 登录 容器服务控制台，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面，单击集群列表上方的**新建**。
3. 在“创建集群”页面，依次填写集群的**集群信息**、**选择机型**、**云服务器配置**及**组件配置**。如下图所示：



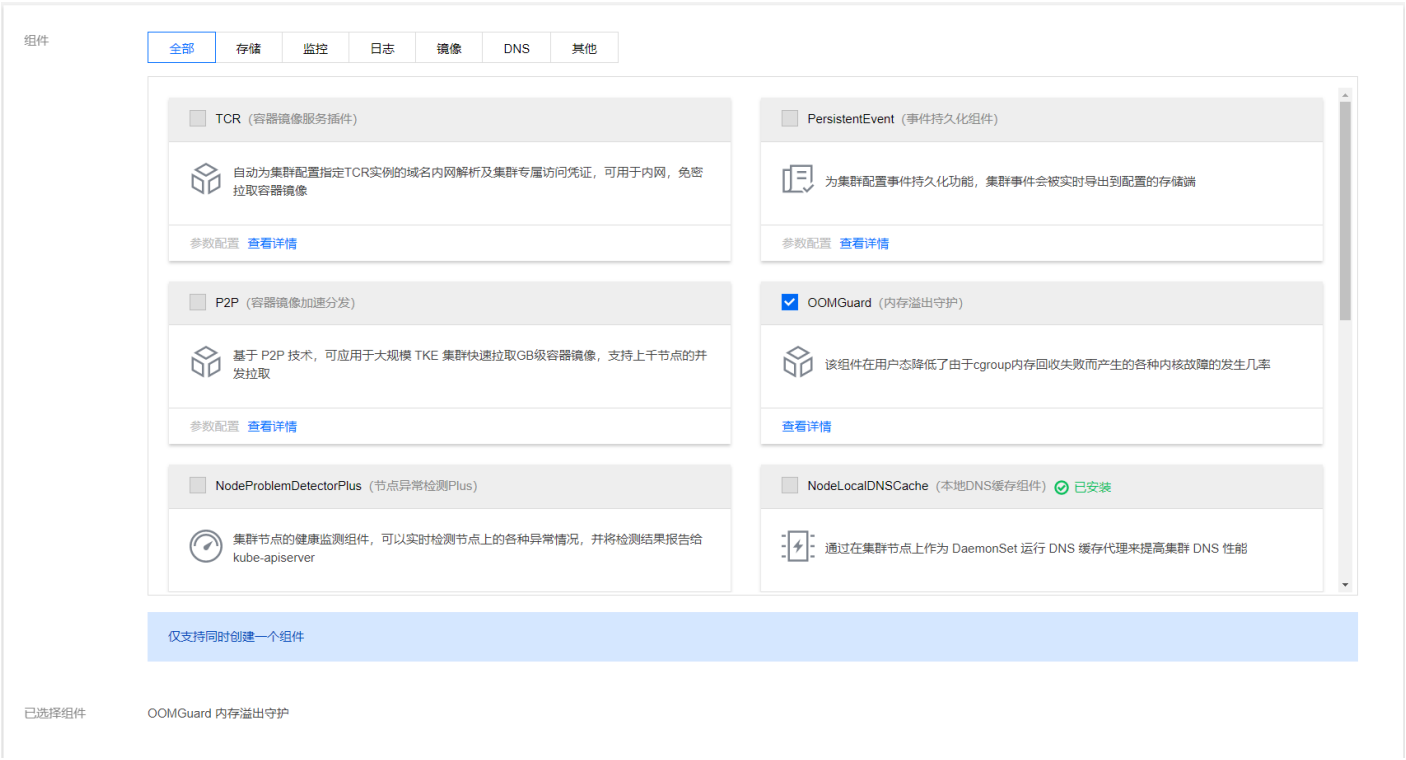
您可以根据业务部署情况，按需选择合适的组件安装。单击每个组件卡片的**查看详情**可以查看该组件的介绍，部分组件需要您先完成**参数配置**。

> **说明：**
>
> - **组件安装为集群创建的非关键路径，安装失败不会影响集群的创建。**
> - **组件安装需要占用集群的一定资源，不同组件的资源占用情况不同，单击查看详情查看每个组件的详细信息。**

4. 单击**下一步**，检查并确认集群配置信息。
5. 单击**完成**，即可完成创建。

通过组件管理页安装

1. 登录 容器服务控制台，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，进入组件安装页面，如下图所示：



5. 选择需要安装的组件并单击**完成**即可。

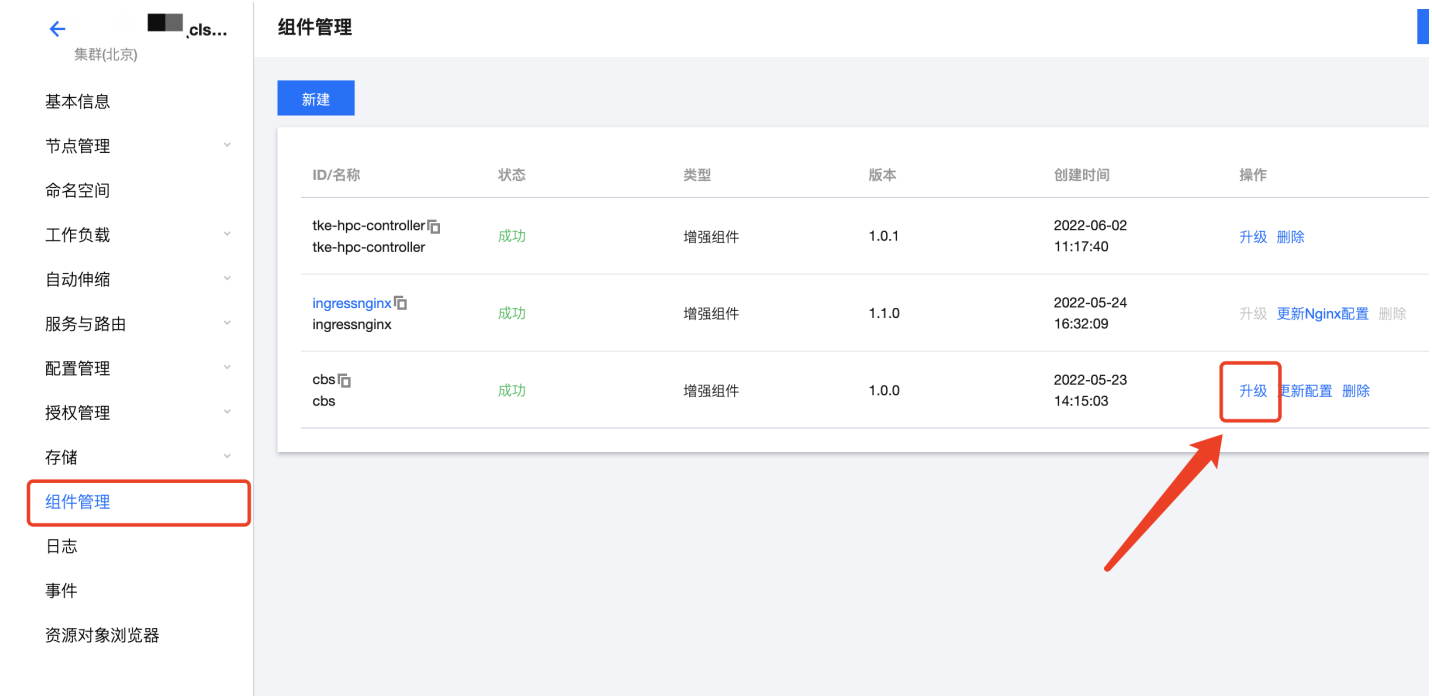
组件卸载

1. 登录 容器服务控制台，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中，单击需要删除组件所在行右侧的**删除**，如下图所示：

5. 在弹出的 “删除资源” 窗口中，单击**确认**即可完成组件卸载。

组件升级

1. 登录 容器服务控制台，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中，单击需要升级组件所在行右侧的**升级**，如下图所示：



5. 在弹出的 “组件升级” 窗口中，单击**确认**即可完成升级。

> **说明：**
> 各组件版本更新详情，请参考 增强组件版本维护说明。

OOMGuard 说明

最近更新时间：2022-06-09 11:38:17

简介

组件介绍

内存溢出（Out of Memory，OOM）是指应用系统中存在无法回收的内存或使用的内存过多，最终使得程序运行要用到的内存大于能提供的最大内存。当 cgroup 内存不足时，Linux 内核会触发 cgroup OOM 来选择一些进程杀掉，以便能回收一些内存从而尽量继续保持系统继续运行。但 Linux 内核（尤其是3.10等低版本内核）对 cgroup OOM 的处理存在很多问题，频繁的 cgroup OOM 经常会带来节点故障（例如卡死、重启或进程异常但无法杀死）的情况。

OOM-Guard 是容器服务 TKE 提供用于在用户态处理容器 cgroup OOM 的组件。当 cgroup OOM 情况出现时，在系统内核杀死相关容器进程之前，OOM-Guard 组件会直接在用户空间杀掉超过内存限制的容器，从而减少了在内核态回收内存失败而触发各种节点故障的概率。

在触发阈值进行 OOM 之前，OOM-Guard 会先通过写入 memory.force_empty 触发相关 cgroup 的内存回收，如果 memory.stat 显示还有较多 cache，则不会触发后续处理策略。在 cgroup OOM 杀掉容器后，会向 Kubernetes 上报 OomGuardKillContainer 事件，可以通过 kubectl get event 命令进行查看。

原理介绍

核心思想是在发生内核 cgroup OOM kill 之前，在用户空间杀掉超限的容器， 减少走到内核 cgroup 内存回收失败后的代码分支从而触发各种内核故障的机会。

oom-guard 会给 memory cgroup 设置 threshold notify，接受内核的通知。详情见 threshold notify。

示例

假如一个 pod 设置的 memory limit 是1000M，oom-guard 会根据配置参数计算出 margin。

```
margin = 1000M * margin_ratio = 20M // 缺省 margin_ratio 是 0.02
```

另外 margin 最小不小于 min_margin（缺省1M），最大不大于 max_margin（缺省为50M）。如果超出范围，则取 min_margin 或 max_margin。

然后计算 threshold：

```
threshold = limit - margin // 即 1000M - 20M = 980M
```

把980M作为阈值设置给内核。当这个 pod 的内存使用量达到980M时，oom-guard 会收到内核的通知。

在触发阈值之前，oom-gurad 会先通过 memory.force_empty 触发相关 cgroup 的内存回收。另外，如果触发阈值时，相关 cgroup 的 memory.stat 显示还有较多 cache，则不会触发后续处理策略，这样当 cgroup 内存达到 limit 时，内核还是会触发 cgroup OOM。

达到阈值后的处理策略

通过 --policy 参数来控制处理策略。目前有以下三个策略，缺省策略是 container。

策略	描述
process	采用跟内核 cgroup OOM killer 相同的策略，在该 cgroup 内部，选择一个 oom_score 得分最高的进程杀掉。通过 oom-guard 发送 SIGKILL 来杀掉进程。
container	在该 cgroup 下选择一个 docker 容器，杀掉整个容器。
noop	只记录日志，并不采取任何措施。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	默认占用资源	所属 Namespaces
oomguard	ServiceAccount	-	kube-system

Kubernetes 对象名称	类型	默认占用资源	所属 Namespaces
system:oomguard	ClusterRoleBinding	-	-
oom-guard	DaemonSet	0.02核 CPU，120MB内存	kube-system

使用场景

应用于节点内存压力比较大，业务容器经常发生 OOM 导致节点故障的 Kubernetes 集群。

限制条件

- 没有修改 containerd 服务 socket 路径，保持 TKE 的默认路径：
  - docker 运行时：/run/docker/containerd/docker-containerd.sock
  - containerd 运行时：/run/containerd/containerd.sock
- 没有修改 cgroup 内存子系统挂载点，保持默认挂载点：/sys/fs/cgroup/memory

使用方法

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 OOM-Guard。
5. 单击**完成**即可安装组件。

NodeProblemDetectorPlus 说明

最近更新时间：2022-04-18 14:15:45

简介

组件介绍

Node-Problem-Detector-Plus 是 Kubernetes 集群节点的健康监测组件。在容器服务 TKE 环境中以 DaemonSet 方式运行，帮助用户实时检测节点上的各种异常情况，并将检测结果报告给上游的 Kube-apiserver。

部署在集群内的 Kubernetes 对象

kubernetes 对象名称	类型	资源量	Namespaces
node-problem-detector	DaemonSet	0.5C80M	kube-system
node-problem-detector	ServiceAccount	-	kube-system
node-problem-detector	ClusterRole	-	-
node-problem-detector	ClusterRoleBinding	-	-

使用场景

使用 Node-Problem-Detector-Plus 组件可以监控节点的工作状态，包括内核死锁、OOM、系统线程数压力、系统文件描述符压力等指标，通过 Node Condition 和 Event 的形式上报给 Apiserver。
您可以通过检测相应的指标，提前预知节点的资源压力，可以在节点开始驱逐 Pod 之前手动释放或扩容节点资源压力，防止 Kubenetes 进行资源回收或节点不可用可能带来的损失。

限制条件

在集群中使用 NPD，需要在集群内安装该扩展组件，NPD 容器将被限制使用0.5核 CPU，80M内存的系统资源。

使用方法

1. 登录 容器服务控制台 ，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 Node-Problem-Detector-Plus。
5. 单击**完成**即可创建组件。安装成功后，您的集群中会有对应的 node-problem-detector 资源，在 Node 的 Condition 中也会增加相应的条目。

附录

Node Conditions

安装 NPD 插件后，会在节点中增加以下特定的 Conditions：

Condition Type	默认值	描述
ReadonlyFilesystem	False	文件系统是否只读
FDPressure	False	查看主机的文件描述符数量是否达到最大值的80%
FrequentKubeletRestart	False	Kubelet 是否在20Min内重启超过5次
CorruptDockerOverlay2	False	DockerImage 是否存在问题
KubeletProblem	False	Kubelet service 是否 Running
KernelDeadlock	False	内核是否存在死锁

Condition Type	默认值	描述
FrequentDockerRestart	False	Docker 是否在20Min内重启超过5次
FrequentContainerdRestart	False	Containerd 是否在20Min内重启超过5次
DockerdProblem	False	Docker service 是否 Running（若节点运行时为 Containerd，则一直为 False）
ContainerdProblem	False	Containerd service 是否 Running（若节点运行时为 Docker，则一直为 False）
ThreadPressure	False	系统目前线程数是否达到最大值的90%
NetworkUnavailable	False	NTP service 是否 Running
SerfFailed	False	分布式检测节点网络健康状态

NodeLocalDNSCache 说明

最近更新时间：2022-01-24 17:17:52

简介

组件介绍

NodeLocal DNSCache 通过在集群节点上作为 DaemonSet 运行 DNS 缓存代理来提高集群 DNS 性能。在当今的体系结构中，处于 ClusterFirst DNS 模式的 Pod 可以连接到 kube-dns serviceIP 进行 DNS 查询。通过 kube-proxy 添加的 iptables 规则将其转换为 kube-dns/CoreDNS 端点。借助此新架构，Pods 将可以访问在同一节点上运行的 DNS 缓存代理，从而避免了 iptables DNAT 规则和连接跟踪。本地缓存代理将查询 kube-dns 服务以获取集群主机名的缓存缺失（默认为 cluster.local 后缀）。

部署在集群内的 Kubernetes 对象

kubernets 对象名称	类型	请求资源	所属 Namespace
node-local-dns	DaemonSet	每节点50mCPU，5Mi内存	kube-system
kube-dns-upstream	Service	-	kube-system
node-local-dns	ServiceAccount	-	kube-system
node-local-dns	Configmap	-	kube-system

限制条件

- 仅支持 1.14 版本以上的 kubernetes 版本。
- VPC-CNI 同时支持 kube-proxy 的 iptables 和 ipvs 模式，GlobalRouter 仅支持 kube-proxy 的 iptables 模式，ipvs 模式下需要更改 kubelet 参数，详情请参见 官方文档。
- 集群创建后没有调整过 dns 服务对应工作负载的相关 name 和 label，检查集群 kube-system 命名空间中存在以下 dns 服务的相关工作负载：
  - service/kube-dns
  - deployment/kube-dns 或者 deployment/coredns，且存在 k8s-app: kube-dns 的 label
- IPVS 的独立集群，需要确保 add-pod-eni-ip-limit-webhook ClusterRole 具备以下权限：

```
- apiGroups:
- ""
resources:
- configmaps
- secrets
- namespaces
- services
verbs:
- list
- watch
- get
- create
- update
- delete
- patch
```

- IPVS 的独立集群和托管集群，都需要确保 tke-eni-ip-webhook Namespace 下的 add-pod-eni-ip-limit-webhook Deployment 镜像版本大于等于 v0.0.6。

推荐配置

当安装 NodeLocal DNSCache 后，推荐为 CoreDNS 增加如下配置：

```
template ANY HINFO . {
rcode NXDOMAIN
}
forward . /etc/resolv.conf {
prefer_udp
}
```

操作步骤

1. 登录 容器服务控制台，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 NodeLocalDNSCache。NodeLocalDNSCache 详细配置可参见 官方文档。
5. 单击**完成**即可创建组件。

DNSAutoscaler 说明

最近更新时间：2022-04-18 14:15:57

简介

组件介绍

DNSAutoscaler 是 DNS 自动水平伸缩组件，可通过一个 deployment 获取集群的节点数和核数，根据预设的伸缩策略，自动水平伸缩 DNS 的副本数。目前的伸缩模式分为两种，分别是 Linear 线性模式 和 Ladder 阶梯模式。

Linear Mode

ConfigMap 配置示例如下：

```
data:
linear: |-
{
"coresPerReplica": 2,
"nodesPerReplica": 1,
"min": 1,
"max": 100,
"preventSinglePointFailure": true
}
```

目标副本计算公式：
replicas = max(ceil(cores _ 1/coresPerReplica) , ceil(nodes _ 1/nodesPerReplica))
replicas = min(replicas, max)
replicas = max(replicas, min)

Ladder Mode

ConfigMap 配置示例如下：

```
data:
ladder: |-
{
"coresToReplicas":
[
[ 1, 1 ],
[ 64, 3 ],
[ 512, 5 ],
[ 1024, 7 ],
[ 2048, 10 ],
[ 4096, 15 ]
],
"nodesToReplicas":
[
[ 1, 1 ],
[ 2, 2 ]
]
}
```

目标副本计算：
假设 100nodes/400cores 的集群中，按上述配置，nodesToReplicas 取2（100>2），coresToReplicas 取3（64<400<512），二者取较大值3，最

终 replica 为3。

部署在集群内的 Kubernetes 对象

kubernets 对象名称	类型	请求资源	所属 Namespace
tke-dns-autoscaler	Deployment	每节点20mCPU，10Mi内存	kube-system
dns-autoscaler	ConfigMap	-	kube-system
tke-dns-autoscale	ServiceAccount	-	kube-system
tke-dns-autoscaler	ClusterRole	-	kube-system
tke-dns-autoscaler	ClusterRoleBinding	-	kube-system

限制条件

- 仅在 1.8 版本以上的 kubernetes 集群支持。
- 集群中的 dns server 的工作负载为 deployment/coredns。

使用方法

1. 登录 容器服务控制台 ，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 DNSAutoscaler。
   该组件默认伸缩配置策略如下：

```
data:
ladder: |-
{
"coresToReplicas":
[
[ 1, 1 ],
[ 128, 3 ],
[ 512,4 ],
],
"nodesToReplicas":
[
[ 1, 1 ],
[ 2, 2 ]
]
}
```

   扩展组件创建成功后，可以通过修改 kube-system 命名空间下的 configmap/tke-dns-autoscaler 来变更配置。详细配置请参见 官方文档。
5. 单击**完成**即可创建组件。

COS-CSI 说明

最近更新时间：2022-06-09 11:34:44

简介

组件介绍

Kubernetes-csi-tencentcloud COS 插件实现 CSI 的接口，可帮助您在容器集群中使用腾讯云对象存储 COS。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	默认占用资源	所属 Namespaces
csi-coslauncher	DaemonSet	-	kube-system
csi-cosplugin	DaemonSet	-	kube-system
csi-cos-tencentcloud-token	Secret	-	kube-system

使用场景

对象存储（Cloud Object Storage，COS）是腾讯云提供的一种存储海量文件的分布式存储服务，用户可通过网络随时存储和查看数据。腾讯云 COS 使所有用户都能使用具备高扩展性、低成本、可靠和安全的数据存储服务。

通过 COS-CSI 扩展组件，您可以快速的在容器集群中通过标准原生 Kubernetes 以 COSFS 的形式使用 COS，详情请参见 COSFS 工具介绍。

限制条件

- 支持 Kubernetes 1.10 以上版本的集群。
- Kubernetes 1.12 版本的集群需要增加 kubelet 配置：--feature-gates=KubeletPluginsWatcher=false。
- COSFS 本身限制，详情请参见 COSFS 局限性。
- 在 TKE 中使用 COS，需要在集群内安装该扩展组件，将占用一定的系统资源。

使用方法

安装 COS 扩展组件

1. 登录 容器服务控制台 ，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 COS。
5. 单击**完成**即可创建组件。

使用对象存储 COS

您可在 TKE 集群中为工作负载挂载对象存储，详情请参见 使用对象存储 COS。

CFS-CSI 说明

最近更新时间：2022-04-18 11:43:33

简介

组件介绍

Kubernetes-csi-tencentloud CFS 插件实现 CSI 的接口，可帮助您在容器集群中使用腾讯云文件存储。

> ⚠ 注意：
> 1.12 集群需要修改 kubelet 配置，增加 \--feature-gates=KubeletPluginsWatcher=false\ 。

部署在集群内的 Kubernetes 对象

kubernetes对象名称	类型	默认占用资源	所属Namespaces
csi-provisioner-cfsplugin	StatefulSet	-	kube-system
csi-nodeplugin-cfsplugin	DaemonSet	-	kube-system
csi-provisioner-cfsplugin	Service	1C2G	kube-system

使用场景

文件存储 CFS 提供了可扩展的共享文件存储服务，可与腾讯云 CVM、容器服务 TKE、批量计算等服务搭配使用。CFS 提供了标准的 NFS 及 CIFS/SMB 文件系统访问协议，为多个 CVM 实例或其他计算服务提供共享的数据源，支持弹性容量和性能的扩展，现有应用无需修改即可挂载使用，是一种高可用、高可靠的分布式文件系统，适合于大数据分析、媒体处理和内容管理等场景。

CFS 接入简单，您无需调节自身业务结构，或者是进行复杂的配置。只需三步即可完成文件系统的接入和使用：创建文件系统，启动服务器上文件系统客户端，挂载创建的文件系统。通过 CFS-CSI 扩展组件，您可以快速在容器集群中通过标准原生 Kubernetes 使用 CFS，详情请参见 CFS 使用场景 。

限制条件

- CFS 自身限制可参见 CFS 系统限制。
- 在 TKE 中使用 CFS，需要在集群内安装该扩展组件，这将占用一定的系统资源。

操作步骤

安装并设置 CFS 扩展组件

1. 登录 容器服务控制台 ，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 CFS。
5. 单击**完成**即可创建组件。

创建 CFS 类型 StroageClass

1. 在“集群管理”页面单击使用 CFS 的集群 ID，进入集群详情页。
2. 在左侧导航栏中选择**存储** > **StorageClass**，单击**新建**进入 “新建StorageClass” 页面。

3. 根据实际需求，创建 CFS 类型的 StorageClass。如下图所示：

4. 单击**创建StorageClass**，完成创建。

创建 PersistentVolumeClaim

1. 在“集群管理”页面单击使用 CFS 的集群 ID，进入集群详情页。
2. 在左侧导航栏中选择**存储** > **PersistentVolumeClaim**，单击**新建**进入 “新建PersistentVolumeClaim” 页面。
3. 根据实际需求，创建 CFS 类型 PersistentVolumeClaim，选择上述步骤创建的 StorageClass。
4. 单击**创建PersistentVolumeClaim**，完成创建。

创建工作负载

1. 在“集群管理”页面单击使用 CFS 的集群 ID，进入集群详情页。
2. 在左侧导航栏中选择**工作负载** > **Deployment**，单击**新建**进入 “新建Workload” 页面。
3. 根据实际需求，数据卷选择**使用已有PVC**，并选择上述已创建的 PVC。
4. 挂载到容器的指定路径后，单击**创建Workload**完成创建。

CBS-CSI 说明
CBS-CSI 简介

最近更新时间：2021-12-31 16:35:36

操作场景

CBS-CSI 组件 支持 TKE 集群通过控制台快捷选择存储类型，并创建对应块存储云硬盘类型的 PV 和 PVC。本文提供 CBS-CSI 组件功能特性等说明并介绍几种常见示例用法。

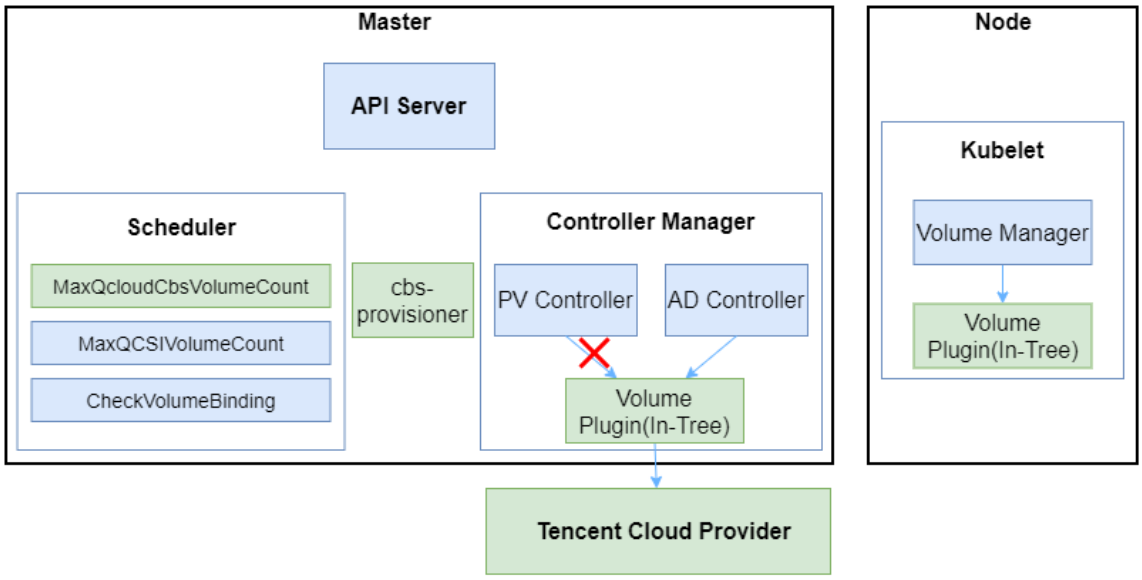
功能特性

功能	说明
静态数据卷	支持手动创建 Volume、PV 对象及 PVC 对象
动态数据卷	支持通过 StorageClass 配置、创建和删除 Volume 及 PV 对象
存储拓扑感知	云硬盘不支持跨可用区挂载，在多可用区集群中，CBS-CSI 组件将先调度 Pod，后调度 Node 的 zone 创建 Volume
调度器感知节点 maxAttachLimit	腾讯云单个云服务器上默认最多挂载20块云硬盘，调度器调度 Pod 时将过滤超过最大可挂载云硬盘数量的节点
卷在线扩容	支持通过修改 PVC 容量字段，实现在线扩容（仅支持云硬盘类型）
卷快照和恢复	支持通过快照创建数据卷

组件说明

CBS-CSI 组件在集群内部署后，包含以下组件：

- DaemonSet：每个 Node 提供一个 DaemonSet，简称为 NodePlugin。由 CBS-CSI Driver 和 node-driver-registrar 两个容器组成，负责向节点注册 Driver，并提供挂载能力。
- StatefulSet 和 Deployment：简称为 Controller。由 Driver 和多个 Sidecar（external-provisioner、external-attacher、external-resizer、external-snapshotter、snapshot-controller）一起构成，提供创删卷、attach、detach、扩容、快照等能力。



限制条件

- TKE 集群版本 ≥ 1.14
- 使用 CBS-CSI 组件后，才可在 TKE 集群中为云硬盘在线扩容和创建快照。
- 已经使用 QcloudCbs（In-Tree 插件）的 TKE 集群，可以继续正常使用。（后续将通过 Volume Migration 统一到 CBS CSI）

使用示例

- 通过 CBS-CSI 避免云硬盘跨可用区挂载
- 在线扩容云硬盘
- 创建快照和使用快照来恢复卷

通过 CBS-CSI 避免云硬盘跨可用区挂载

最近更新时间：2021-12-31 16:35:53

操作场景

云硬盘不支持跨可用区挂载到节点，在跨可用区的集群环境中，推荐通过 CBS-CSI **拓扑感知**特性来避免跨可用区挂载问题。

实现原理

拓扑感知调度需要多个 Kubernetes 组件配合完成，包括 Scheduler、PV controller、external-provisioner。具体流程如下：

1. PV controller 观察 PVC 对象，检查 Storageclass 的 VolumeBindingMode 是否为 **WaitForFirstConsumer**，如是，则不会立即处理该 PVC 的创建事件，等待 Scheduler 处理。
2. Scheduler 调度 Pod 后，会将 nodeName 以 annotation 的方式加入到 PVC 对象上 volume.kubernetes.io/selected-node: 10.0.0.72。
3. PV controller 获取到 PVC 对象的更新事件后，将开始处理 annotation（volume.kubernetes.io/selected-node），根据 nodeName 获取 Node 对象，传入到 external-provisioner 中。
4. external-provisioner 根据传过来的 Node 对象的 label 获取可用区（failure-domain.beta.kubernetes.io/zone）后在对应可用区创建 PV，达到和 Pod 相同可用区的效果，避免云硬盘和 Node 在不同可用区而无法挂载问题。

前提条件

- 已安装1.14或以上版本的 TKE 集群。
- 已将 CBS-CSI 或 In-Tree 组件更新为最新版本。

操作步骤

使用以下 YAML，在 Storageclass 中设置 volumeBindingMode 为 **WaitForFirstConsumer**。示例如下：

```
kind: StorageClass
metadata:
name: cbs-topo
parameters:
type: cbs
provisioner: com.tencent.cloud.csi.cbs
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

> 说明：
> CBS-CSI 和 In-Tree 组件均支持该操作。

在线扩容云硬盘

最近更新时间：2022-01-04 08:56:03

操作场景

TKE 支持在线扩容 PV、对应的云硬盘及文件系统，即不需要重启 Pod 即可完成扩容。为确保文件系统的稳定性，建议在云硬盘文件系统处于未挂载状态时进行操作。

前提条件

- 已创建1.16或以上版本的 TKE 集群。
- 已将 CBS-CSI 更新为最新版本。
- （可选）为避免扩容失败导致数据丢失，可以在扩容前 使用快照备份数据。

操作步骤

创建允许扩容的 StorageClass

使用以下 YAML 创建允许扩容的 StorageClass，在 Storageclass 中设置 allowVolumeExpansion 为 true 。示例如下：

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: cbs-csi-expand
parameters:
diskType: CLOUD_PREMIUM
provisioner: com.tencent.cloud.csi.cbs
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

在线扩容

提供以下两种扩容方式：

扩容方式	说明
重启 Pod 的情况下在线扩容	待扩容的云硬盘文件系统未被挂载，能够避免扩容出错以及方式2存在的问题。**推荐使用该方式进行扩容**。
不重启 Pod 的情况下在线扩容	在节点上挂载着待扩容的云硬盘文件系统，如果存在 I/O 进程，将可能出现文件系统扩容错误。

重启Pod情况下在线扩容

1. 执行以下命令，确认扩容前 PV 和文件系统状态。示例如下，PV 和文件系统大小均为30G：

```
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/vdd 30832548 44992 30771172 1% /usr/share/nginx/html
$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c 30Gi RWO Delete Bound default/www1-ivantestweb-0 cbs-csi 20h
```

2. 执行以下命令，为 PV 对象打上一个非法 zone 标签，旨在下一步重启 Pod 后，使 Pod 无法调度到某个节点上。示例如下：

```
$ kubectl label pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c failure-domain.beta.kubernetes.io/zone=nozone
```

3. 执行以下命令重启 Pod，重启后由于 Pod 对应的 PV 的标签表明的是非法 zone，Pod 将处于 Pending 状态。示例如下：

```
$ kubectl delete pod ivantestweb-0
$ kubectl get pod ivantestweb-0
NAME READY STATUS RESTARTS AGE
ivantestweb-0 0/1 Pending 0 25s
$ kubectl describe pod ivantestweb-0
Events:
Type Reason Age From Message
---- ------ ---- ---- -------
Warning FailedScheduling 40s (x3 over 2m3s) default-scheduler 0/1 nodes are available: 1 node(s) had no available volume zone.
```

4. 执行以下命令，修改 PVC 对象中的容量，将容量扩容至40G。示例如下：

```
kubectl patch pvc www1-ivantestweb-0 -p '{"spec":{"resources":{"requests":{"storage":"40Gi"}}}}'
```

> ⚠ 注意：
> 扩容后的PVC对象容量的大小必须为10的倍数，不同云硬盘类型所支持的存储容量规格可参考说明 创建云硬盘。

5. 执行以下命令，去除 PV 对象之前打上的标签， 标签去除之后 Pod 即可调度成功。示例如下：

```
$ kubectl label pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c failure-domain.beta.kubernetes.io/zone-
persistentvolume/pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c labeled
```

6. 执行以下命令，可以查看到 Pod 状态为 Running、对应的 PV 和文件系统扩容成功，从30G扩容到40G。示例如下：

```
$ kubectl get pod ivantestweb-0
NAME READY STATUS RESTARTS AGE
ivantestweb-0 1/1 Running 0 17m
$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c 40Gi RWO Delete Bound default/www1-ivantestweb-0 cbs-csi 20h
$ kubectl get pvc www1-ivantestweb-0
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
www1-ivantestweb-0 Bound pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c 40Gi RWO cbs-csi 20h
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/vdd 41153760 49032 41088344 1% /usr/share/nginx/html
```

不重启Pod情况下在线扩容

1. 执行以下命令，确认扩容前 PV 和文件系统状态。示例如下，PV 和文件系统大小均为20G：

```
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/vdd 20511312 45036 20449892 1% /usr/share/nginx/html
$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c 20Gi RWO Delete Bound default/www1-ivantestweb-0 cbs-csi 20h
```

2. 执行以下命令，修改 PVC 对象中的容量，将容量扩容至30G。示例如下：

```
$ kubectl patch pvc www1-ivantestweb-0 -p '{"spec":{"resources":{"requests":{"storage":"30Gi"}}}}'
```

> ⚠ 注意：
> 扩容后的PVC对象容量的大小必须为10的倍数，不同硬盘类型所支持的存储容量规格可参考说明 创建云硬盘。

3. 执行以下命令，可以查看到 PV 和文件系统已扩容至30G。示例如下：

```
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/vdd 30832548 44992 30771172 1% /usr/share/nginx/html
$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c 30Gi RWO Delete Bound default/www1-ivantestweb-0 cbs-csi 20h
```

创建快照和使用快照来恢复卷

最近更新时间：2022-03-18 11:32:20

操作场景

如需为 PVC 数据盘创建快照来备份数据，或者将备份的快照数据恢复到新的 PVC 中，可以通过 CBS-CSI 插件来实现，本文将介绍如何利用 CBS-CSI 插件实现 PVC 的数据备份与恢复。

前提条件

- 已创建1.18或以上版本的 TKE 集群。
- 已安装最新版的 CBS-CSI 组件。

操作步骤

备份PVC

创建 VolumeSnapshotClass

1. 使用以下 YAML，创建 VolumeSnapshotClass 对象。示例如下：

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
name: cbs-snapclass
driver: com.tencent.cloud.csi.cbs
deletionPolicy: Delete
```

2. 执行以下命令查看 VolumeSnapshotClass 是否创建成功。示例如下：

```
$ kubectl get volumesnapshotclass
NAME DRIVER DELETIONPOLICY AGE
cbs-snapclass com.tencent.cloud.csi.cbs Delete 17m
```

创建 PVC 快照 VolumeSnapshot

1. 本文以 new-snapshot-demo 快照名为例，使用以下 YAML 创建 VolumeSnapshot 对象。示例如下：

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
name: new-snapshot-demo
spec:
volumeSnapshotClassName: cbs-snapclass
source:
persistentVolumeClaimName: csi-pvc
```

2. 执行以下命令，查看 Volumesnapshot 和 Volumesnapshotcontent 对象是否创建成功，若 READYTOUSE 为 true，则创建成功。示例如下：

```
$ kubectl get volumesnapshot
NAME READYTOUSE SOURCEPVC SOURCESNAPSHOTCONTENT RESTORESIZE SNAPSHOTCLASS SNAPSHOTCONTENT CREATIONTIME AG
E
new-snapshot-demo true www1-ivantestweb-0 10Gi cbs-snapclass snapcontent-ea11a797-d438-4410-ae21-41d9147fe610 22m 22m
```

```
$ kubectl get volumesnapshotcontent
NAME READYTOUSE RESTORESIZE DELETIONPOLICY DRIVER VOLUMESNAPSHOTCLASS VOLUMESNAPSHOT AGE
snapcontent-ea11a797-d438-4410-ae21-41d9147fe610 true 10737418240 Delete com.tencent.cloud.csi.cbs cbs-snapclass new-snaps
hot-demo 22m
```

3. 执行以下命令，可以获取 Volumesnapshotcontent 对象的快照 ID，字段是 status.snapshotHandle（如下为 snap-e406fc9m），可以根据快照 ID 在 容器服务控制台 确认快照是否存在。示例如下：

```
$ kubectl get volumesnapshotcontent snapcontent-ea11a797-d438-4410-ae21-41d9147fe610 -oyaml
```

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotContent
metadata:
creationTimestamp: "2020-11-04T08:58:39Z"
finalizers:
- snapshot.storage.kubernetes.io/volumesnapshotcontent-bound-protection
name: snapcontent-ea11a797-d438-4410-ae21-41d9147fe610
resourceVersion: "471437790"
selfLink: /apis/snapshot.storage.k8s.io/v1beta1/volumesnapshotcontents/snapcontent-ea11a797-d438-4410-ae21-41d9147fe610
uid: 70d0390b-79b8-4276-aa79-a32e3bdef3d6
spec:
deletionPolicy: Delete
driver: com.tencent.cloud.csi.cbs
source:
volumeHandle: disk-7z32tin5
volumeSnapshotClassName: cbs-snapclass
volumeSnapshotRef:
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
name: new-snapshot-demo
namespace: default
resourceVersion: "471418661"
uid: ea11a797-d438-4410-ae21-41d9147fe610
status:
creationTime: 1604480319000000000
readyToUse: true
restoreSize: 10737418240
snapshotHandle: snap-e406fc9m
```

从快照恢复数据到新 pvc

1. 本文以上述 步骤 中创建的 VolumeSnapshot 的对象名为 new-snapshot-demo 为例，使用以下 YAML 从快照恢复卷。示例如下：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: restore-test
spec:
storageClassName: cbs-csi
dataSource:
name: new-snapshot-demo
```

```
kind: VolumeSnapshot
apiGroup: snapshot.storage.k8s.io
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 10Gi
```

2. 执行以下命令，查看恢复的 PVC 已成功创建，从 PV 中可以查看到对应的 diskid（如下为 disk-gahz1kw1）。示例如下：

```
$ kubectl get pvc restore-test
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
restore-test Bound pvc-80b98084-29a3-4a38-a96c-2f284042cf4f 10Gi RWO cbs-csi 97s
```

```
$ kubectl get pv pvc-80b98084-29a3-4a38-a96c-2f284042cf4f -oyaml
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
annotations:
pv.kubernetes.io/provisioned-by: com.tencent.cloud.csi.cbs
creationTimestamp: "2020-11-04T12:08:25Z"
finalizers:
- kubernetes.io/pv-protection
name: pvc-80b98084-29a3-4a38-a96c-2f284042cf4f
resourceVersion: "474676883"
selfLink: /api/v1/persistentvolumes/pvc-80b98084-29a3-4a38-a96c-2f284042cf4f
uid: 5321df93-5f21-4895-bafc-71538d50293a
spec:
accessModes:
- ReadWriteOnce
capacity:
storage: 10Gi
claimRef:
apiVersion: v1
kind: PersistentVolumeClaim
name: restore-test
namespace: default
resourceVersion: "474675088"
uid: 80b98084-29a3-4a38-a96c-2f284042cf4f
csi:
driver: com.tencent.cloud.csi.cbs
fsType: ext4
volumeAttributes:
diskType: CLOUD_PREMIUM
storage.kubernetes.io/csiProvisionerIdentity: 1604478835151-8081-com.tencent.cloud.csi.cbs
volumeHandle: disk-gahz1kw1
nodeAffinity:
required:
nodeSelectorTerms:
- matchExpressions:
```

```
- key: topology.com.tencent.cloud.csi.cbs/zone
operator: In
values:
- ap-beijing-2
persistentVolumeReclaimPolicy: Delete
storageClassName: cbs-csi
volumeMode: Filesystem
status:
phase: Bound
```

说明：

如果 StorageClass 使用了拓扑感知（先调度 Pod 再创建 PV），即指定 volumeBindingMode: WaitForFirstConsumer **，则需要先部署 Pod（需挂载 PVC）才会触发创建 PV（从快照创建新的 CBS 并与 PV 绑定）。**

TCR 说明

最近更新时间：2022-04-24 16:45:29

简介

组件介绍

TCR Addon 是容器镜像服务 TCR 推出的容器镜像内网免密拉取的官方插件。在容器服务 TKE 集群中安装该插件后，集群节点可通过内网拉取企业版实例内容器镜像，且无需在集群资源 YAML 中显式配置 ImagePullSecret。可提高 TKE 集群内镜像拉取速度，简化镜像配置流程。

> 说明：
>
> - TKE 集群需为 v1.10.x 及以上版本。建议在v1.12.x 及以上版本中使用本插件。
> - Kubernetes 的 controller manager 组件的启动参数需要包含 authentication-kubeconfig 和 authorization-kubeconfig（TKE v.12.x 默认启用）。

部署在集群内的 Kubernetes 对象

名称	类型	资源量	Namespace
tcr-assistant-system	Namespace	1	-
tcr-assistant-manager-role	ClusterRole	1	-
tcr-assistant-manager-rolebinding	ClusterRoleBinding	1	-
tcr-assistant-leader-election-role	Role	1	tcr-assistant-system
tcr-assistant-leader-election-rolebinding	RoleBinding	1	tcr-assistant-system
tcr-assistant-webhook-server-cert	Secret	1	tcr-assistant-system
tcr-assistant-webhook-service	Service	1	tcr-assistant-system
tcr-assistant-validating-webhook-configuration	ValidatingWebhookConfiguration	1	tcr-assistant-system
imagepullsecrets.tcr.tencentcloudcr.com	CustomResourceDefinition	1	tcr-assistant-system
tcr.ips*	ImagePullSecret CRD	(2-3)	tcr-assistant-system
tcr.ips*	Secret	(2-3)*{Namespace No.}	tcr-assistant-system
tcr-assistant-controller-manager	Deployment	1	tcr-assistant-system
updater-config	ConfigMap	1	tcr-assistant-system
hosts-updater	DaemonSet	{Node No.}	tcr-assistant-system

组件资源用量

组件	资源用量	实例数量

组件	资源用量	实例数量
tcr-assistant-controller-manager	CPU：100m memory：30Mi	1
hosts-updater	CPU：100m memory：100Mi	工作节点数

使用场景

免密拉取镜像

Kubernetes 集群拉取私有镜像需要创建访问凭证 Secret 资源，并配置资源 YAML 中的 ImagePullSecret 属性，显式指定已创建的 Secret。整体配置流程较为繁琐，且会因未配置 ImagePullSecret 或指定错误 Secret 而造成镜像拉取失败。
为解决以上问题，可集群中安装 TCR 插件，插件将自动获取指定的 TCR 企业版实例的访问凭证，并下发至 TKE 集群指定命名空间内。在使用 YAML 创建或更新资源时，无需配置 ImagePullSecret，集群会将自动使用已下发的访问凭证拉取 TCR 企业版内镜像。

内网拉取镜像

组件将自动创建 DaemonSet 工作负载 host-updater，用于更新集群节点的Host配置，解析当前关联实例域名至已建立的内网访问链路专用内网 IP 上。请注意，本配置仅用于测试场景配置，建议直接使用 TCR 提供的内网链路自动解析功能，或直接使用 PrivateDNS 产品进行私有域解析配置，也可使用自建 DNS 服务自行管理解析。

限制条件

针对免密拉取镜像使用场景：

- 用户需要具有指定的 TCR 企业版实例的获取访问凭证的权限，即 CreateInstanceToken 接口调用权限。建议具有 TCR 管理员权限的用户进行此插件的配置。
- 安装插件并生效后，请避免在资源 YAML 中重复指定 ImagePullSecret，从而造成节点使用错误的镜像拉取访问凭证，引起拉取失败。

使用方法

1. 选择关联实例：选择当前登录账户下已有的 TCR 企业版实例，并确认当前登录用户具有创建实例长期访问凭证的权限。如果需要新建企业版实例，请在当前集群所在地域内新建。
2. 配置免密拉取（默认启用）：可选自动下发当前操作用户的访问凭证，或指定用户名及密码，可选配置免密拉取生效的命名空间及 ServiceAccount。建议均使用默认配置，避免新建命名空间后无法使用该功能。
3. 配置内网解析（高级功能）：确认集群与关联 TCR 实例已建立内网访问链路，并启用内网解析功能。请注意，本配置仅用于测试场景配置，建议直接使用TCR提供的内网链路自动解析功能，或直接使用 PrivateDNS 产品进行私有域解析配置，也可使用自建DNS服务自行管理解析。
4. 创建插件完成后，如需修改插件相关配置，请删除插件并重新配置及安装。

> ⚠ 注意：
> 删除插件将不会同时删除自动创建的专属访问凭证，可前往 TCR 控制台手动禁用或删除。

原理说明

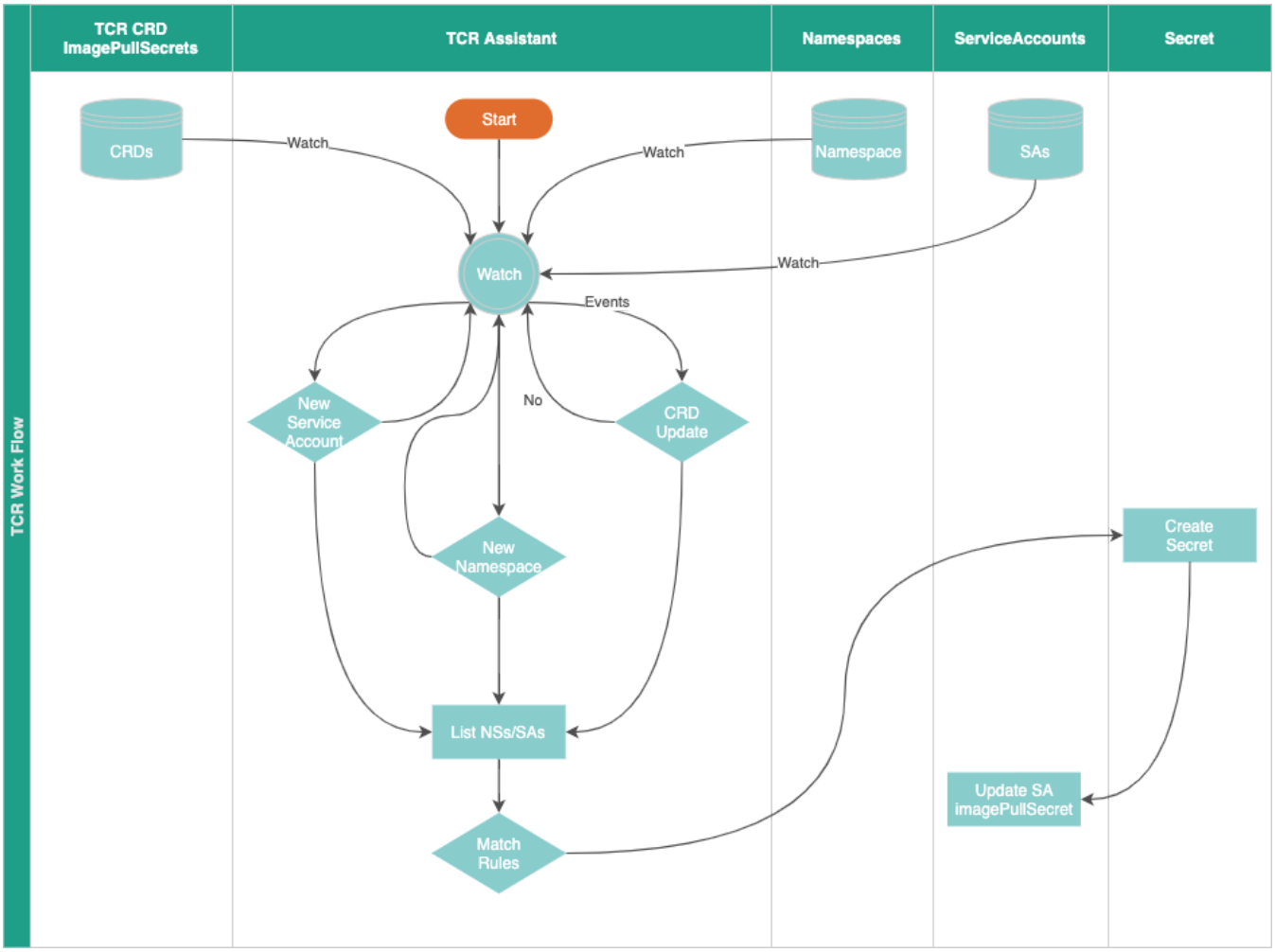
概述

TCR Assistant 用于帮助用户自动部署 k8s imagePullSecret 到任意 Namespace，并关联到该空间下的 ServiceAccount。在用户创建的工作负载当中**没有明确指定** imagePullSecret 和 serviceAccount 的情况下，k8s 会尝试从当前命名空间下名为 default 的 ServiceAccount 资源中查找、匹配合适的 imagePullSecret。

术语表

Name	别名	描述
ImagePullSecret	ips, ipss	TCR Assistant 定义的 CRD。用于存储镜像仓库用户名与密钥，分发目标 Namespace 和目标 ServiceAccount。

实现原理



TCR Assistant 是一个典型的 k8s Operator。在部署 TCR Assistant 时，我们会在目标 k8s 集群当中创建 CRD 对象：imagepullsecrets.tcr.tencentcloudcr.com 。该 CRD 的 kind 为 ImagePullSecret，版本是 tcr.tencentcloudcr.com/v1，缩写为 ips 或者 ipss 。

TCR Assistant 通过持续观察（watch）k8s 集群的 Namespace 和 ServiceAccount 资源，并在这些资源发生变更的时候，检查资源变化是否匹配 ImagePullSecret 中设定的规则来自动的为用户部署拉取私有镜像仓库所需要的 Secret 资源。程序通常部署在 k8s 集群内，使用 in cluster 模式访问 k8s master API。

创建 CRD 资源

程序部署完成后，不会在目标 k8s 集群部署任何的 TCR 镜像拉取密钥。此时，需要我们使用 kubectl 或者通过 Client Go 新建 ImagePullSecret 资源。

```
# 新建 ImagePullSecret 资源
$ kubectl create -f allinone/imagepullsecret-sample.yaml

imagepullsecret.tcr.tencentcloudcr.com/imagepullsecret-sample created
```

ImagePullSecret 资源示例文件（allinone/imagepullsecret-sample.yaml）：

```
apiVersion: tcr.tencentcloudcr.com/v1
kind: ImagePullSecret
metadata:
name: imagepullsecret-sample
spec:
namespaces: "*"
serviceAccounts: "*"
docker:
```

```
username: "100012345678"
password: tcr.jwt.token
server: fanjiankong-bj.tencentcloudcr.com
```

ImagePullSecret spec 字段解释如下表：

字段	作用	注释
namespaces	NameSpace 匹配规则	* 或者空字符表示匹配任意；要匹配任意多个 NameSpace 则使用 , 分隔多个资源名称，注意：不支持任何表达式，需要明确填写资源名称。
serviceAccounts	serviceAccounts 匹配规则	* 或者空字符表示匹配任意；要匹配任意多个 ServiceAccount 则使用 , 分隔多个资源名称，注意：不支持任何表达式，需要明确填写资源名称。
docker.server	镜像仓库域名	仅填写仓库域名
docker.username	镜像仓库用户名	请确保用户在镜像仓库拥有足够的访问权限
docker.password	镜像仓库用户名所对应的密码	

创建完成后，我们可以使用下列命令观察 TCR Assistant 执行结果：

```
# 列出 ImagePullSecret 信息
$ kubectl get ipss
NAME NAMESPACES SERVICE-ACCOUNTS SECRETS-DESIRED SECRETS-SUCCESS
imagepullsecret-sample * * 10 10

# 查看详细信息
$ kubectl describe ipss
Name: imagepullsecret-sample
Namespace:
Labels: <none>
Annotations: <none>
API Version: tcr.tencentcloudcr.com/v1
Kind: ImagePullSecret
Metadata:
Creation Timestamp: 2021-12-01T06:47:34Z
Generation: 1
Manager: kubectl-client-side-apply
Operation: Update
Time: 2021-12-01T06:47:34Z
API Version: tcr.tencentcloudcr.com/v1
Manager: manager
Operation: Update
Time: 2021-12-01T06:47:38Z
Resource Version: 30389349
UID: 2109f384-240b-405c-9ce8-73ce938a7c2f
Spec:
Docker:
Password: tcr.jwt.token
Server: fanjiankong-bj.tencentcloudcr.com
Username: 100012345678
Namespaces: *
Service Accounts: *
Status:
```

```
S As Desired: 47
S As Success: 1
Secret Update Successful:
Namespaced Name: kube-public/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:36Z
Namespaced Name: devtools/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:36Z
Namespaced Name: demo/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:36Z
Namespaced Name: kube-system/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:36Z
Namespaced Name: tcr-assistant-system/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:36Z
Namespaced Name: kube-node-lease/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:36Z
Namespaced Name: cert-manager/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:36Z
Namespaced Name: default/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:36Z
Namespaced Name: afm/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:37Z
Namespaced Name: lens-metrics/tcr.ipsimagepullsecret-sample
Updated At: 2021-12-01T06:47:37Z
Secrets Desired: 10
Secrets Success: 10
Service Accounts Modify Successful:
Namespaced Name: default/default
Updated At: 2021-12-01T06:47:38Z
Events: <none>
```

注意：

如果需要更新 TCR Assistant 部署的 Secret **资源，无需删除重建** ImagePullSecret **资源，只需要编辑其中** docker.username **和** docker.password **字段即可生效。例如：**

```
$ kubectl edit ipss imagepullsecret-sample
```

Namespace 变更

TCR Assistant 在观察到有新的 k8s Namespace **资源创建后，会首先检查名称是否和** ImagePullSecret **资源中的** namespaces **字段匹配。如果资源名称不匹配跳过后续流程；资源名称匹配的情况下，会调用 k8s API 创建** Secret **资源，并添加** Secret **资源名称到** ServiceAccount **资源的** imagePullSecrets **字段当中。示例如下：**

```
# 查看 newns 下自动部署的 Secret
$ kubectl get secrets -n newns
NAME TYPE DATA AGE
tcr.ipsimagepullsecret-sample kubernetes.io/dockerconfigjson 1 7m2s
default-token-nb5vw kubernetes.io/service-account-token 3 7m2s

# 查看 newns 下自动关联到 ServiceAccount 资源 default 中的 Secret
$ kubectl get serviceaccounts default -o yaml -n newns
apiVersion: v1
```

```
imagePullSecrets:
- name: tcr.ipsimagepullsecret-sample
kind: ServiceAccount
metadata:
creationTimestamp: "2021-12-01T07:09:56Z"
name: default
namespace: newns
resourceVersion: "30392461"
uid: 7bc67144-3685-4666-ba41-b1447bbbaa38
secrets:
- name: default-token-nb5vw
```

ServiceAccount 变更

TCR Assistant 在观察到有新的 k8s ServiceAccount **资源创建后，会首先检查名称是否和** ImagePullSecret **资源中的** serviceAccounts **字段匹配。如果资源名称不匹配跳过后续流程；资源名称匹配的情况下，会调用 k8s API 创建或更新** Secret **资源，并添加** Secret **资源名称到** ServiceAccount **资源的** imagePullSecrets **字段当中。示例如下：**

```
# 在 newns 新建 ServiceAccount 资源
$ kubectl create sa kung -n newns
serviceaccount/kung created

# 查看 newns 下自动关联到新建 ServiceAccount 资源 kung 中的 Secret
$ kubectl get serviceaccounts kung -o yaml -n newns
apiVersion: v1
imagePullSecrets:
- name: tcr.ipsimagepullsecret-sample
kind: ServiceAccount
metadata:
creationTimestamp: "2021-12-01T07:19:12Z"
name: kung
namespace: newns
resourceVersion: "30393760"
uid: e236829e-d88e-4feb-9e80-5e4a40f2aea2
secrets:
- name: kung-token-fljt8
```

P2P 说明

最近更新时间：2022-04-18 11:43:39

简介

组件介绍

P2P Addon 是容器镜像服务 TCR 推出的基于 P2P 技术的容器镜像加速分发插件，可应用于大规模容器服务 TKE 集群快速拉取 GB 级容器镜像，支持上千节点的并发拉取。

该组件由 p2p-agent 、p2p-proxy 和 p2p-tracker 组成：

- p2p-agent：部署在集群中每个节点上，代理每个节点的镜像拉取请求，并转发至 P2P 网络的各个 peer（node 节点）间。
- p2p-proxy：部署在集群部分节点上，作为原始种子连接被加速的镜像仓库。proxy 节点既需要做种，也需要从目标镜像仓库中拉取原始数据。
- p2p-tracker：部署在集群部分节点上，开源 bittorrent 协议的 tracker 服务。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	请求资源	所属 Namespace
p2p-agent	DaemonSet	每个节点0.2核 CPU，0.2G内存	kube-system
p2p-proxy	Deployment	每个节点0.5核 CPU，0.5G内存	kube-system
p2p-tracker	Deployment	每个节点0.5核 CPU，0.5G内存	kube-system
p2p-proxy	Service	-	kube-system
p2p-tracker	Service	-	kube-system
agent	Configmap	-	kube-system
proxy	Configmap	-	kube-system
tracker	Configmap	-	kube-system

使用场景

应用于大规模 TKE 集群快速拉取 GB 级容器镜像，支持上千节点的并发拉取，推荐如下使用场景：

- 集群内具备节点500 - 1000台，使用本地盘存储拉取的容器镜像。此场景下，集群内节点最高可支持100MB/s的并发拉取速度。
- 集群内具备节点500 - 1000台，使用 CBS 云盘存储拉取的容器镜像，且集群所在地域为广州、北京、上海等国内主要地域。此场景下，集群内节点最高可支持20MB/s的并发拉取速度。

限制条件

- 在大规模集群内启用 P2P Addon 拉取容器镜像时，将对节点数据盘造成较高读写压力，可能影响集群内已有业务。若集群内节点使用 CBS 云盘存储拉取的容器镜像，请按照集群所在地域选择合适的下载限速或联系您的售后/架构师，避免因镜像拉取时云盘读写负载过高造成集群内现网业务中断现象，甚至影响该地域内其他用户的正常使用。
- 开启 P2P 插件需要预留一定的资源，P2P 组件在镜像加速拉取的过程中会占用节点的 CPU 和内存资源，加速结束后不再占用资源。其中：
  - Proxy 的 limit 限制为：4核 CPU 和4G 内存。
  - Agent 的 limit 限制为：4核 CPU 和2G 内存。
  - Tracker 的 limit 限制为：2核 CPU 和4G 内存。
- 需要根据集群的节点规模，估算启动的 Proxy 个数。Proxy 运行节点的最低配置为4C8G，内网带宽1.5GB/s，单个 Proxy 服务可支撑200个集群节点。
- 需要主动为 Proxy 和 Tracker 组件选择部署节点，使用方式为手动为节点打 K8S 标签，详情请参见 使用方法。Proxy 和 Agent 所在的节点需要能够访问的仓库源站。
- Agent 组件将会占用节点的5004端口，以及 P2P 专用通信端口6881（Agent）和6882（Proxy）。Agent、Proxy 组件会分别创建本地工作目录 /p2p_agent_data 和 /p2p_proxy_data 用于缓存容器镜像，请提前确认节点已预留足够的存储空间。

使用方法

1. 选取合适的节点部署运行 Proxy 组件。
   可通过 kubectl label nodes XXXX proxy=p2p-proxy 命令标记节点，插件安装时将自动在这些节点中部署该组件。安装后如果需要调整 Proxy 组件的个数，可在指定节点上添加或者删除该 label 后，修改集群中 kube-system 命名空间下 p2p-proxy 工作负载的副本个数。
2. 选取合适的节点部署运行 Tracker 组件。
   可通过 kubectl label nodes XXXX tracker=p2p-tracker 命令标记节点，插件安装时将自动在这些节点中部署该组件。安装后如果需要调整 Tracker 的个数，可在指定节点上添加或者删除该 label 后，修改集群中 kube-system 命名空间下 p2p-tracker 工作负载的副本个数。
3. 节点安全组需要添加的配置为：入站规则放通 TCP 和 UDP 的30000 - 32768 端口、以及 VPC 内 IP 全放通。出站规则放通全部（TKE 集群 work 节点默认安全组已满足要求）。
4. 选择指定集群 开启 P2P Addon 插件。填写需要加速的镜像仓库域名，节点拉取限速、Proxy 个数，Tracker 个数。安装后如果需要重新调整下载的最高速度，可修改 p2p-agent configmap 中的 downloadRate 和 uploadRate。
5. 在业务命名空间内创建拉取镜像所需的 dockercfg，其中仓库域名为 localhost:5004，用户名及密码即为目标镜像仓库的原有访问凭证。
6. 修改业务 YAML，将需要加速的镜像仓库域名地址修改为 localhost:5004，如 localhost:5004/p2p-test/test:1.0，并使用新建的 dockercfg 作为 ImagePullSecret。
7. 使用业务 YAML 部署更新工作负载，并实时观察镜像拉取速度及节点磁盘读写负载，及时调整节点的下载限速以达到最好加速效果。

操作步骤

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入“组件列表”页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 P2P。
5. 选择“参数配置”，在弹出的“P2P组件参数设置”窗口中，填写需要加速的镜像仓库域名、节点拉取限速、Proxy 个数及 Tracker 个数。如下图所示：

DynamicScheduler 说明

最近更新时间：2022-04-18 11:44:09

简介

组件介绍

Dynamic Scheduler 是容器服务 TKE 基于 Kubernetes 原生 Kube-scheduler Extender 机制实现的动态调度器插件，可基于 Node 真实负载进行预选和优选。在 TKE 集群中安装该插件后，该插件将与 Kube-scheduler 协同生效，有效避免原生调度器基于 request 和 limit 调度机制带来的节点负载不均问题。
该组件依赖 Prometheus 监控组件以及相关规则配置，可参见本文 依赖部署 进行操作，避免遇到插件无法正常工作的情况。

部署在集群内的 Kubernetes 对象

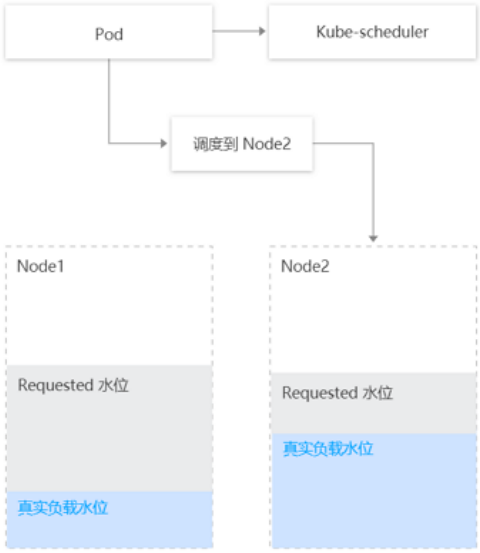
Kubernetes 对象名称	类型	请求资源	所属 Namespace
node-annotator	Deployment	每个实例 CPU:100m，Memory:100Mi ，共1个实例	kube-system
dynamic-scheduler	Deployment	每个实例 CPU:400m，Memory:200Mi，共3个实例	kube-system
dynamic-scheduler	Service	-	kube-system
node-annotator	ClusterRole	-	kube-system
node-annotator	ClusterRoleBinding	-	kube-system
node-annotator	ServiceAccount	-	kube-system
dynamic-scheduler-policy	ConfigMap	-	kube-system
restart-kube-scheduler	ConfigMap	-	kube-system
probe-prometheus	ConfigMap	-	kube-system

应用场景

集群负载不均

Kubernetes 原生调度器大部分基于 Pod Request 资源进行调度，并不具备根据 Node 当前和过去一段时间的真实负载情况进行相关调度的决策，因此可能会导致如下问题：
集群内部分节点的剩余可调度资源较多（根据节点上运行的 Pod 的 request 和 limit 计算出的值）但真实负载却比较高，而另外节点的剩余可调度资源比较少但真实负载却比较低，此时 Kube-scheduler 会优先将 Pod 调度到剩余资源比较多的节点上（根据 LeastRequestedPriority 策略）。

如下图所示，Kube-Scheduler 会将 Pod 调度到 Node2 上，但明显调度到 Node1（真实负载水位更低）是更优的选择。



防止调度热点

为防止低负载的节点被持续调度 Pod，Dynamic Scheduler 支持设置防调度热点策略（统计节点过去几分钟调度 Pod 的数量，并相应减小节点在优选阶段的评分）。
当前采取策略如下：

- 如果节点在过去1分钟调度了超过2个 Pod，则优选评分减去1分。
- 如果节点在过去5分钟调度了超过5个 Pod，则优选评分减去1分。

风险控制

- 该组件已对接 TKE 的监控告警体系。
- 推荐您为集群开启事件持久化，以便更好的监控组件异常以及故障定位。
- 该组件卸载后，只会删除动态调度器有关调度逻辑，不会对原生 Kube-Scheduler 的调度功能有任何影响。

限制条件

- TKE 版本建议 ⩾ v1.10.x
- 如果需要升级 Kubernetes master 版本：
  - 对于托管集群无需再次设置本插件。
  - 对于独立集群，master 版本升级会重置 master 上所有组件的配置，从而影响到 Dynamic Scheduler 插件作为 Scheduler Extender 的配置，因此 Dynamic Scheduler 插件需要卸载后再重新安装。

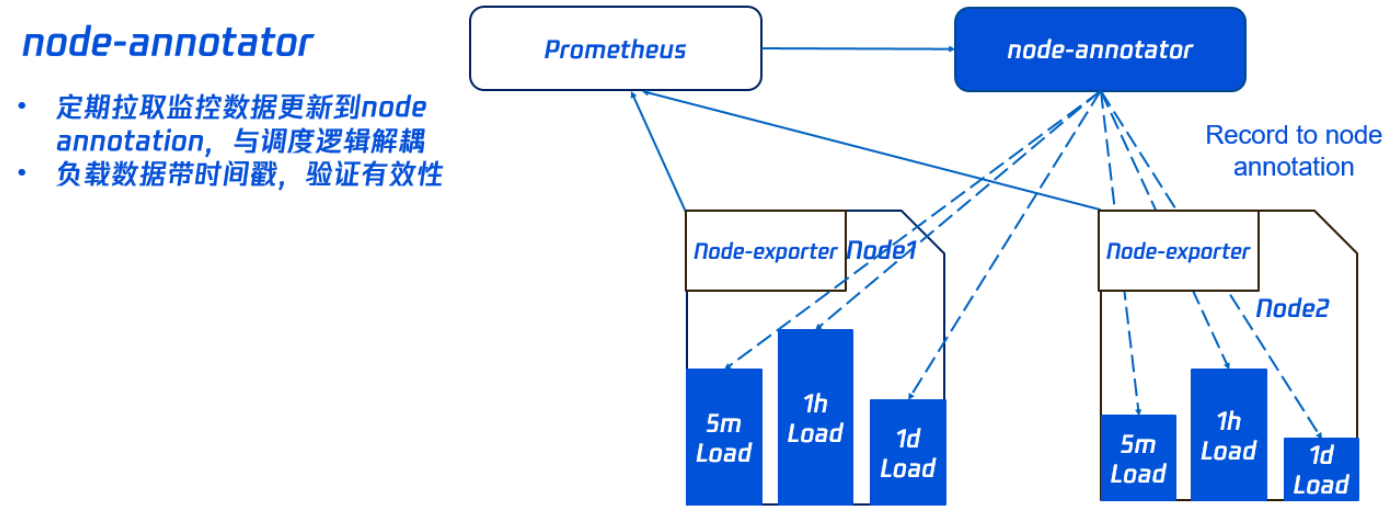
组件原理

动态调度器基于 scheduler extender 扩展机制，从 Prometheus 监控数据中获取节点负载数据，开发基于节点实际负载的调度策略，在调度预选和优选阶段进行干预，优先将 Pod 调度到低负载节点上。该组件由 node-annotator 和 Dynamic-scheduler 构成。

node-annotator

node-annotator 组件负责定期从监控中拉取节点负载 metric，同步到节点的 annotation。如下图所示：

> ⚠ 注意：
> 组件删除后，node-annotator 生成的 annotation 并不会被自动清除。您可根据需要手动清除。

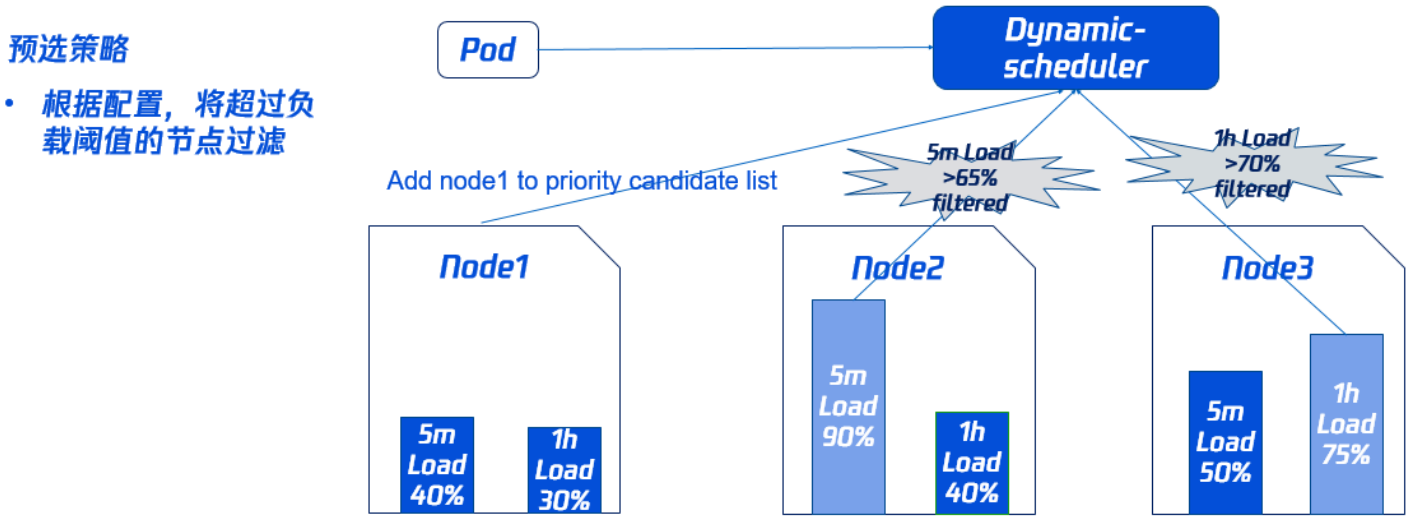


Dynamic-scheduler

Dynamic-scheduler 是一个 scheduler-extender，根据 node annotation 负载数据，在节点预选和优选中进行过滤和评分计算。

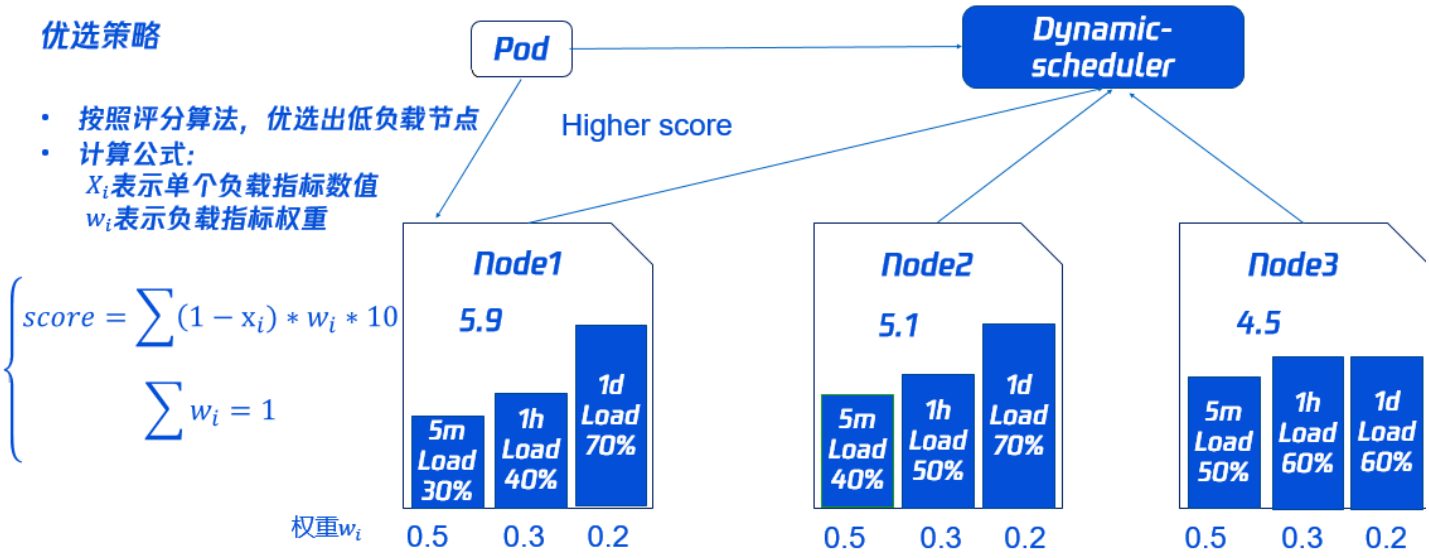
预选策略

为了避免 Pod 调度到高负载的 Node 上，需要先通过预选过滤部分高负载的 Node（其中过滤策略和比例可以动态配置，具体请参见本文 组件参数说明）。
如下图所示，Node2 过去5分钟的负载，Node3 过去1小时的负载均超过对应的域值，因此不会参与接下来的优选阶段。



优选策略

同时为了使集群各节点的负载尽量均衡，Dynamic-scheduler 会根据 Node 负载数据进行打分，负载越低打分越高。
如下图所示，Node1 的打分最高将会被优先调度（其中打分策略和权重可以动态配置，具体请参见本文 组件参数说明）。



组件参数说明

Prometheus 数据查询地址

> 注意：
>
> - 为确保组件可以拉取到所需的监控数据、调度策略生效，请按照 依赖部署> Prometheus 规则配置步骤配置监控数据采集规则。
> - 预选和优选参数已设置默认值，如您无额外需求，可直接采用。

- 如果使用自建 Prometheus，直接填入数据查询 URL（HTTP/HTTPS）即可。
- 如果使用托管 Prometheus，选择托管实例 ID 即可，系统会自动解析实例对应的数据查询 URL。

预选参数

预选参数默认值	说明
5分钟平均 CPU 利用率阈值	节点过去5分钟**平均** CPU 利用率超过设定阈值，不会调度 Pod 到该节点上。
1小时最大 CPU 利用率阈值	节点过去1小时**最大** CPU 利用率超过设定阈值，不会调度 Pod 到该节点上。
5分钟平均**内存**利用率阈值	节点过去5分钟**平均**内存利用率超过设定阈值，不会调度 Pod 到该节点上。
1小时最大**内存**利用率阈值	节点过去1小时**最大**内存利用率超过设定阈值，不会调度 Pod 到该节点上。

优选参数

优选参数默认值	说明
5分钟平均 CPU 利用率权重	该权重越大，过去5分钟节点**平均** CPU 利用率对节点的评分影响越大。
1小时最大 CPU 利用率权重	该权重越大，过去1小时节点**最大** CPU 利用率对节点的评分影响越大。
1天最大 CPU 利用率权重	该权重越大，过去1天内节点**最大** CPU 利用率对节点的评分影响越大。
5分钟平均**内存**利用率权重	该权重越大，过去5分钟节点**平均**内存利用率对节点的评分影响越大。
1小时最大**内存**利用率权重	该权重越大，过去1小时节点**最大**内存利用率对节点的评分影响越大。
1天最大**内存**利用率权重	该权重越大，过去1天内节点**最大**内存利用率对节点的评分影响越大。

操作步骤

依赖部署

Dynamic Scheduler 动态调度器依赖于 Node 当前和过去一段时间的真实负载情况来进行调度决策，需通过 Prometheus 等监控组件获取系统 Node 真实负载信息。在使用动态调度器之前，需要部署 Prometheus 等监控组件。在容器服务 TKE 中，您可按需选择采用自建的 Prometheus 监控服务或采用 TKE 推出的云原生监控。

自建Prometheus监控服务

部署 node-exporter 和 prometheus

通过 node-exporter 实现对 Node 指标的监控，用户可以根据业务需求部署 node-exporter 和 prometheus。

聚合规则配置

在 node-exporter 获取节点监控数据后，需要通过 Prometheus 对原始的 node-exporter 采集数据进行聚合计算。为了获取动态调度器中需要的 cpu_usage_avg_5m 、cpu_usage_max_avg_1h 、cpu_usage_max_avg_1d 、mem_usage_avg_5m 、mem_usage_max _avg_1h 、mem_usage_max_avg_1d 等指标，需要在 Prometheus 的 rules 规则进行如下配置：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
name: example-record
spec:
groups:
- name: cpu_mem_usage_active
interval: 30s
rules:
- record: cpu_usage_active
expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[30s])) * 100)
- record: mem_usage_active
expr: 100*(1-node_memory_MemAvailable_bytes/node_memory_MemTotal_bytes)
- name: cpu-usage-5m
interval: 5m
rules:
- record: cpu_usage_max_avg_1h
expr: max_over_time(cpu_usage_avg_5m[1h])
- record: cpu_usage_max_avg_1d
expr: max_over_time(cpu_usage_avg_5m[1d])
- name: cpu-usage-1m
interval: 1m
rules:
- record: cpu_usage_avg_5m
expr: avg_over_time(cpu_usage_active[5m])
- name: mem-usage-5m
interval: 5m
rules:
- record: mem_usage_max_avg_1h
expr: max_over_time(mem_usage_avg_5m[1h])
- record: mem_usage_max_avg_1d
expr: max_over_time(mem_usage_avg_5m[1d])
- name: mem-usage-1m
interval: 1m
rules:
- record: mem_usage_avg_5m
expr: avg_over_time(mem_usage_active[5m])
```

Prometheus 文件配置

1. 上述定义了动态调度器所需要的指标计算的 rules，需要将 rules 配置到 Prometheus 中，参考一般的 Prometheus 配置文件。示例如下：

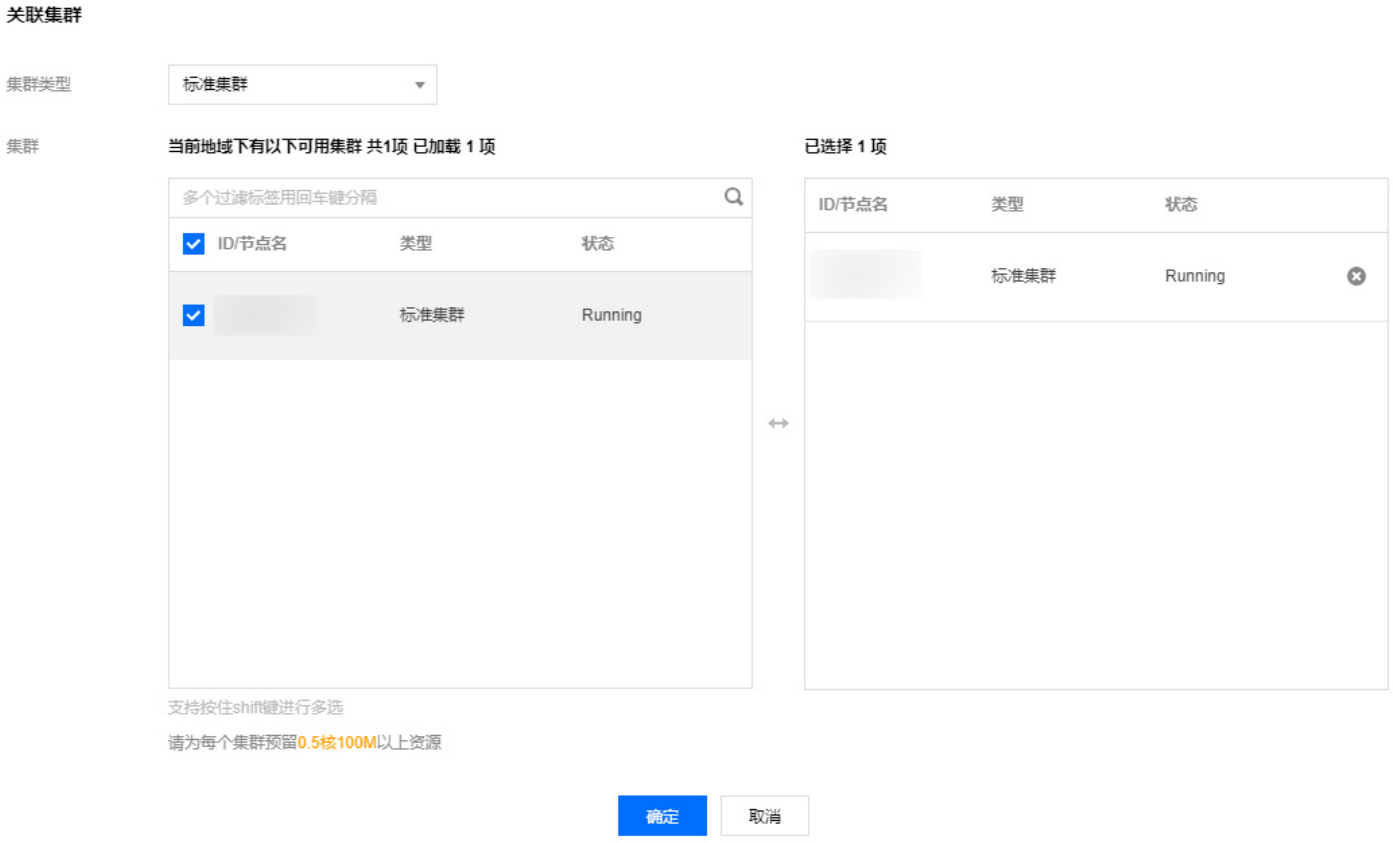
```
global:
evaluation_interval: 30s
scrape_interval: 30s
external_labels:
rule_files:
- /etc/prometheus/rules/*.yml # /etc/prometheus/rules/*.yml 是定义的rules文件
```

2. 将 rules 配置复制到一个文件（例如 dynamic-scheduler.yaml），文件放到上述 prometheus 容器的 /etc/prometheus/rules/ 目录下。
3. 加载 Prometheus server，即可从 Prometheus 获取到动态调度器需要的指标。

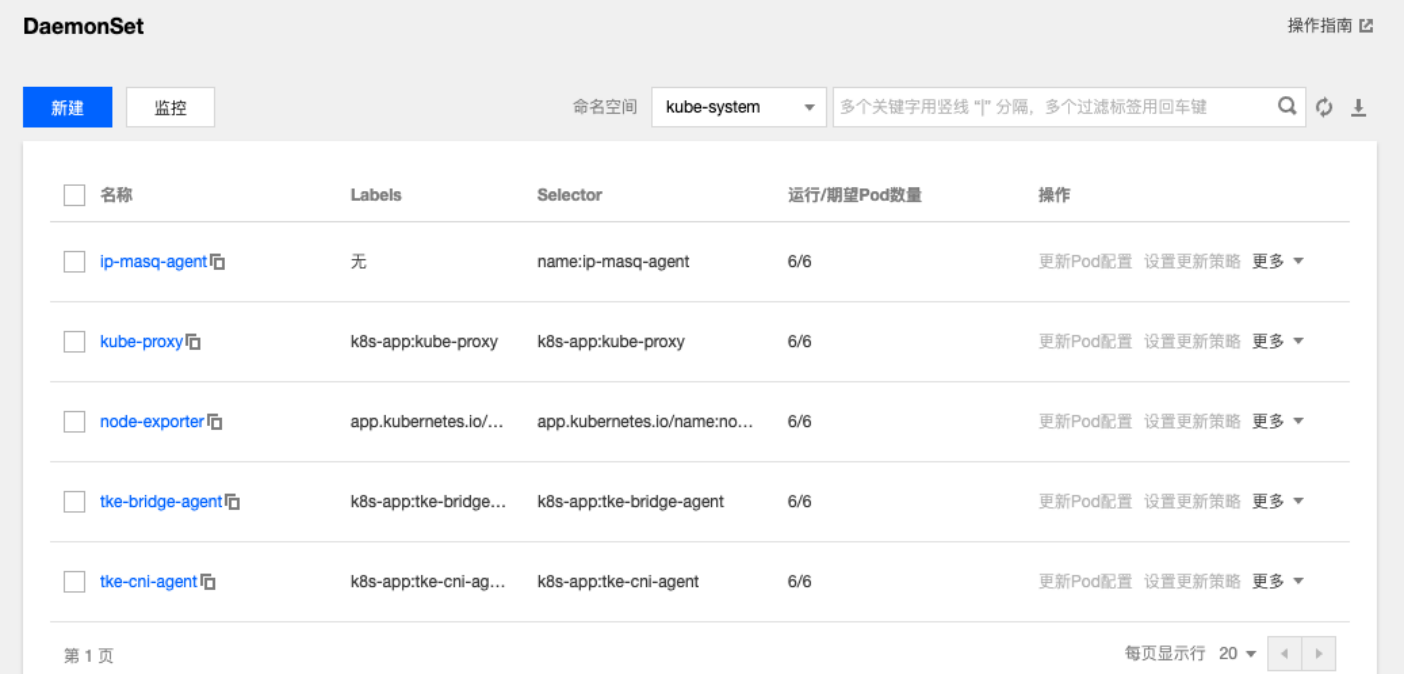
> 说明：
> 通常情况下，上述 Prometheus 配置文件和 rules 配置文件都是通过 configmap 存储，再挂载到 Prometheus server 容器，因此修改相应的 configmap 即可。

云原生监控 Prometheus

1. 登录容器服务控制台，在左侧菜单栏中选择 **云原生监控**，进入“云原生监控”页面。
2. 创建与 Cluster 处于同一 VPC 下的 云原生监控 Prometheus 实例，并 关联用户集群。如下图所示：



3. 与原生托管集群关联后，可以在用户集群查看到每个节点都已安装 node-exporter。如下图所示：



4. 设置 Prometheus 聚合规则，具体规则内容与上述 自建Prometheus监控服务 中的“聚合规则配置”相同。如下所示：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
name: example-record
spec:
groups:
- name: cpu_mem_usage_active
interval: 30s
rules:
- record: cpu_usage_active
expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[30s])) * 100)
- record: mem_usage_active
expr: 100*(1-node_memory_MemAvailable_bytes/node_memory_MemTotal_bytes)
- name: cpu-usage-5m
interval: 5m
rules:
- record: cpu_usage_max_avg_1h
expr: max_over_time(cpu_usage_avg_5m[1h])
- record: cpu_usage_max_avg_1d
expr: max_over_time(cpu_usage_avg_5m[1d])
- name: cpu-usage-1m
interval: 1m
rules:
- record: cpu_usage_avg_5m
expr: avg_over_time(cpu_usage_active[5m])
- name: mem-usage-5m
interval: 5m
rules:
- record: mem_usage_max_avg_1h
expr: max_over_time(mem_usage_avg_5m[1h])
- record: mem_usage_max_avg_1d
expr: max_over_time(mem_usage_avg_5m[1d])
```

```
- name: mem-usage-1m
interval: 1m
rules:
- record: mem_usage_avg_5m
expr: avg_over_time(mem_usage_active[5m])
```

安装组件

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 DynamicScheduler（动态调度器插件）。
5. 单击**参数配置**，按照 参数说明 填写组件所需参数。
6. 单击**完成**即可创建组件。安装成功后，Dynamic Scheduler 即可正常运行，无需进行额外配置。

DeScheduler 说明

最近更新时间：2022-04-18 11:44:01

简介

组件介绍

DeScheduler 是容器服务 TKE 基于 Kubernetes 原生社区 DeScheduler 实现的一个基于 Node 真实负载进行重调度的插件。在 TKE 集群中安装该插件后，该插件会和 Kube-scheduler 协同生效，实时监控集群中高负载节点并驱逐低优先级 Pod。建议您搭配 TKE Dynamic Scheduler（动态调度器扩展组件）一起使用，多维度保障集群负载均衡。
该插件依赖 Prometheus 监控组件以及相关规则配置，建议您安装插件之前仔细阅读 依赖部署，以免插件无法正常工作。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	请求资源	所属 Namespace
descheduler	Deployment	每个实例 CPU:200m，Memory:200Mi，共1个实例	kube-system
descheduler	ClusterRole	-	kube-system
descheduler	ClusterRoleBinding	-	kube-system
descheduler	ServiceAccount	-	kube-system
descheduler-policy	ConfigMap	-	kube-system
probe-prometheus	ConfigMap	-	kube-system

使用场景

DeScheduler 通过重调度来解决集群现有节点上不合理的运行方式。社区版本 DeScheduler 中提出的策略基于 APIServer 中的数据实现，并未基于节点真实负载。因此可以增加对于节点的监控，基于真实负载进行重调度调整。

容器服务 TKE 自研的 ReduceHighLoadNode 策略依赖 Prometheus 和 node_exporter 监控数据，根据节点 CPU 利用率、内存利用率、网络 IO、system loadavg 等指标进行 Pod 驱逐重调度，防止出现节点极端负载的情况。DeScheduler 的 ReduceHighLoadNode 与 TKE 自研的 Dynamic Scheduler 基于节点真实负载进行调度的策略需配合使用。

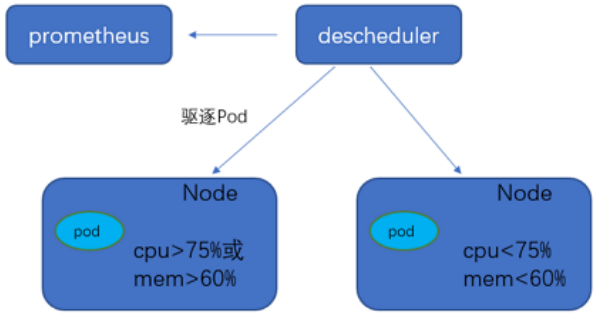
限制条件

- Kubernetes 版本 ≥ v1.10.x
- 在特定场景下，某些 Pod 会被重复调度到需要重调度的节点上，从而引发 Pod 被重复驱逐。此时可以根据实际场景改变 Pod 可调度的节点，或者将 Pod 标记为不可驱逐。
- 该组件已对接容器服务 TKE 的监控告警体系。
- 建议您为集群开启事件持久化，以便更好的监控组件异常以及故障定位。Descheduler 驱逐 Pod 时会产生对应事件，可根据 reason 为“Descheduled” 类型的事件观察 Pod 是否被重复驱逐。
- 为避免 DeScheduler 驱逐关键的 Pod，设计的算法默认不驱逐 Pod。对于可以驱逐的 Pod，用户需要显示给判断 Pod 所属 workload。例如，statefulset、deployment 等对象设置可驱逐 annotation。
- 驱逐大量 Pod，导致服务不可用。
  Kubernetes 原生提供 PDB 对象用于防止驱逐接口造成的 workload 不可用 Pod 过多，但需要用户创建该 PDB 配置。容器服务 TKE 自研的 DeScheduler 组件加入了兜底措施，即调用驱逐接口前，判断 workload 准备的 Pod 数是否大于副本数一半，否则不调用驱逐接口。

组件原理

DeScheduler 基于 社区版本 Descheduler 的重调度思想，定期扫描各个节点上的运行 Pod，发现不符合策略条件的进行驱逐以进行重调度。社区版本 DeScheduler 已提供部分策略，策略基于 APIServer 中的数据，例如 LowNodeUtilization 策略依赖的是 Pod 的 request 和 limit 数据，这类数据能够有效均衡集群资源分配、防止出现资源碎片。但社区策略缺少节点真实资源占用的支持，例如节点 A 和 B 分配出去的资源一致，由于 Pod 实际运行情况，CPU 消耗型和内存消耗型不同，峰谷期不同造成两个节点的负载差别巨大。

因此，腾讯云 TKE 推出 DeScheduler，底层依赖对节点真实负载的监控进行重调度。通过 Prometheus 拿到集群 Node 的负载统计信息，根据用户设置的负载阈值，定期执行策略里面的检查规则，驱逐高负载节点上的 Pod。



组件参数说明

Prometheus 数据查询地址

> 注意：
> 为确保组件可以拉取到所需的监控数据、调度策略生效，请按照 依赖部署>Prometheus 文件配置步骤配置监控数据采集规则。

- 如果使用自建 Prometheus，直接填入数据查询 URL（HTTPS/HTTPS）即可。
- 如果使用托管 Prometheus，选择托管实例 ID 即可，系统会自动解析实例对应的数据查询 URL。

利用率阈值和目标利用率

> 注意：
> 负载阈值参数已设置默认值，如您无额外需求，可直接采用。

过去5分钟内，节点的 CPU 平均利用率或者内存平均使用率超过设定阈值，Descheduler 会判断节点为高负载节点，执行 Pod 驱逐逻辑，并尽量通过 Pod 重调度使节点负载降到目标利用率以下。

操作步骤

依赖部署

DeScheduler 组件依赖于 Node 当前和过去一段时间的真实负载情况来进行调度决策，需要通过 Prometheus 等监控组件获取系统 Node 真实负载信息。在使用 DeScheduler 组件之前，您可以采用自建 Prometheus 监控或采用 TKE 云原生监控。

自建 Prometheus 监控服务

部署 node-exporter 和 Prometheus

通过 node-exporter 实现对于 Node 指标的监控，您可按需部署 node-exporter 和 Prometheus。

聚合规则配置

在 node-exporter 获取节点监控数据后，需要通过 Prometheus 对原始的 node-exporter 中采集数据进行聚合计算。为获取 DeScheduler 所需要的 cpu_usage_avg_5m 、mem_usage_avg_5m 等指标，需要在 Prometheus 的 rules 规则中进行配置。示例如下：

```
groups:
- name: cpu_mem_usage_active
interval: 30s
rules:
- record: mem_usage_active
expr: 100*(1-node_memory_MemAvailable_bytes/node_memory_MemTotal_bytes)
- name: cpu-usage-1m
interval: 1m
```

```
rules:
- record: cpu_usage_avg_5m
expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
- name: mem-usage-1m
interval: 1m
rules:
- record: mem_usage_avg_5m
expr: avg_over_time(mem_usage_active[5m])
```

注意：

当您使用 TKE 提供的 DynamicScheduler 时，需在 Prometheus 配置获取 Node 监控数据的聚合规则。DynamicScheduler 聚合规则与 DeScheduler 聚合规则有部分重合，但并不完全一样，请您在配置规则时不要互相覆盖。同时使用 DynamicScheduler 和 DeScheduler 时应该配置如下规则：

```
groups:
- name: cpu_mem_usage_active
interval: 30s
rules:
- record: mem_usage_active
expr: 100*(1-node_memory_MemAvailable_bytes/node_memory_MemTotal_bytes)
- name: mem-usage-1m
interval: 1m
rules:
- record: mem_usage_avg_5m
expr: avg_over_time(mem_usage_active[5m])
- name: mem-usage-5m
interval: 5m
rules:
- record: mem_usage_max_avg_1h
expr: max_over_time(mem_usage_avg_5m[1h])
- record: mem_usage_max_avg_1d
expr: max_over_time(mem_usage_avg_5m[1d])
- name: cpu-usage-1m
interval: 1m
rules:
- record: cpu_usage_avg_5m
expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
- name: cpu-usage-5m
interval: 5m
rules:
- record: cpu_usage_max_avg_1h
expr: max_over_time(cpu_usage_avg_5m[1h])
- record: cpu_usage_max_avg_1d
expr: max_over_time(cpu_usage_avg_5m[1d])
```

Prometheus 文件配置

1. 上述定义了 DeScheduler 所需要的指标计算的 rules，需要将 rules 配置到 Prometheus 中，参考一般的 Prometheus 配置文件。示例如下：

```
global:
evaluation_interval: 30s
scrape_interval: 30s
external_labels:
```

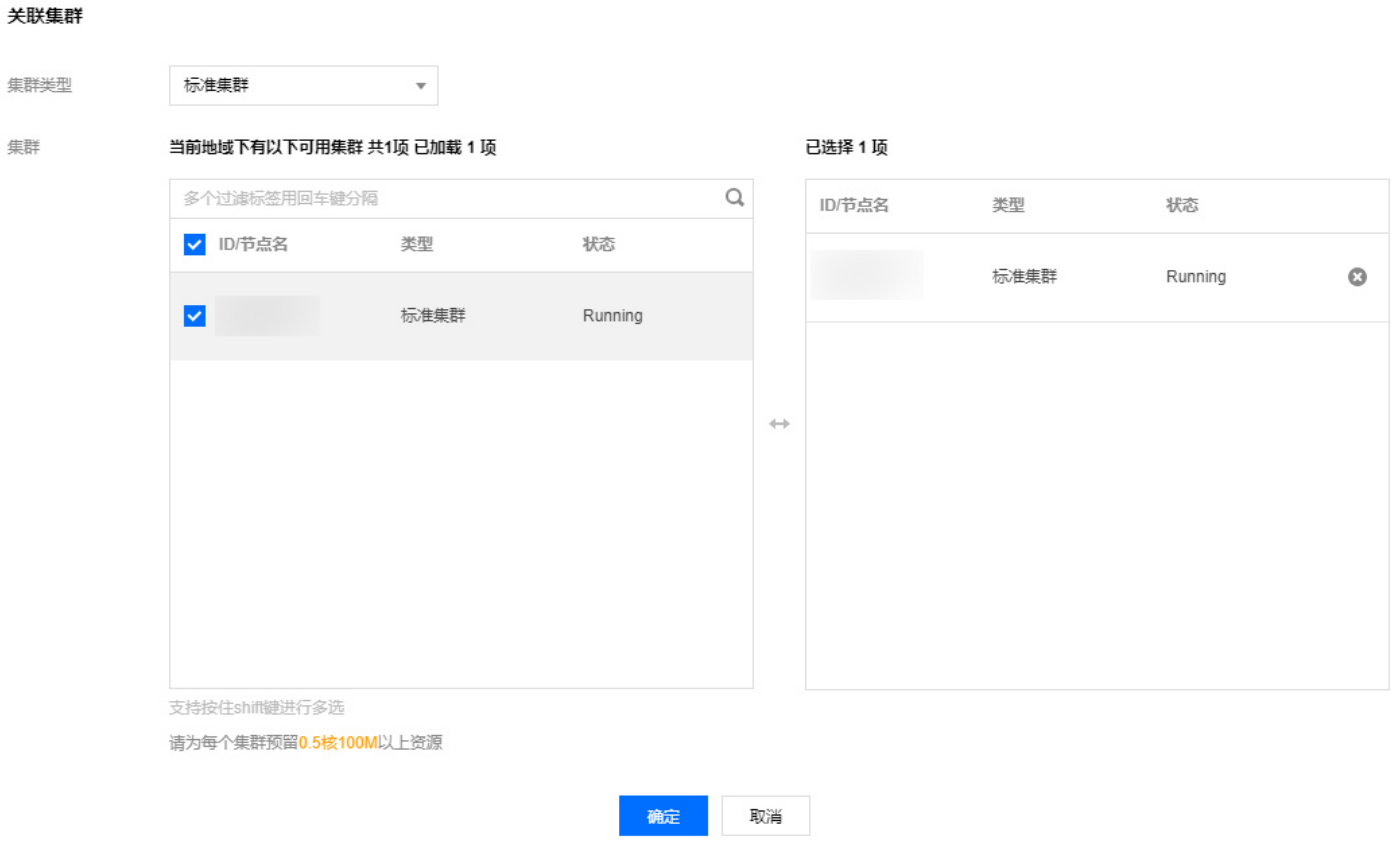
```
rule_files:
- /etc/prometheus/rules/*.yml # /etc/prometheus/rules/*.yml 是定义的 rules 文件
```

2. 将 rules 配置复制到一个文件（例如 de-scheduler.yaml），文件放到上述 Prometheus 容器的 /etc/prometheus/rules/ 下。
3. 重新加载 Prometheus server，即可从 Prometheus 中获取到动态调度器需要的指标。

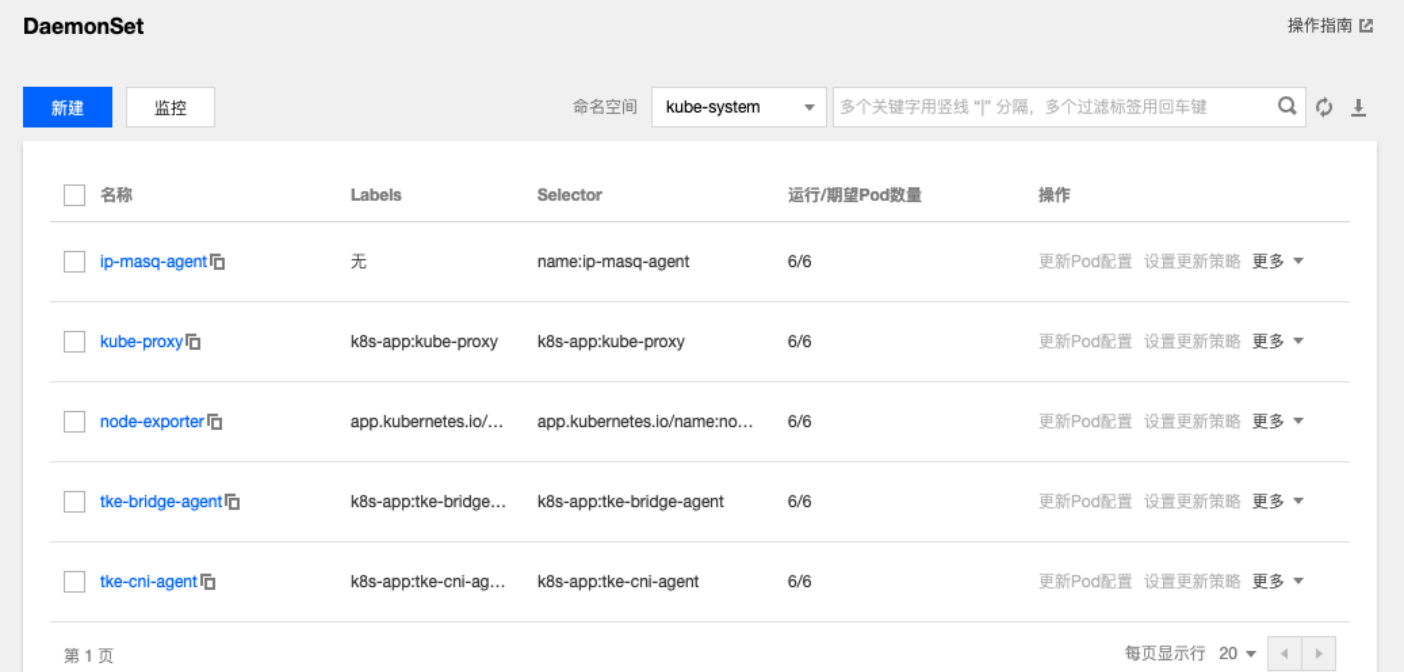
> 说明：
> 通常情况下，上述 Prometheus 配置文件和 rules 配置文件都是通过 configmap 存储，再挂载到 Prometheus server 容器，因此修改相应的 configmap 即可。

云原生监控 Prometheus

1. 登录容器服务控制台，在左侧菜单栏中选择 **云原生监控**，进入“云原生监控”页面。
2. 创建与 Cluster 处于同一 VPC 下的 云原生监控 Prometheus 实例，并 关联用户集群。如下图所示：



3. 与原生托管集群关联后，可以在用户集群查看到每个节点都已安装 node-exporter。如下图所示：



4. 设置 Prometheus 聚合规则，具体规则内容与上述 自建Prometheus监控服务 中的“聚合规则配置”相同。规则保存后立即生效，无需重新加载 server。

安装组件

1. 登录 容器服务控制台 ，选择左侧导航栏中的**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 Decheduler（重调度器）。
5. 单击**参数配置**，按照 参数说明 填写组件所需参数。
6. 单击**完成**即可创建组件。安装成功后，DeScheduler 即可正常运行，无需进行额外配置。
7. 若您需要驱逐 workload（例如 statefulset、deployment 等对象），可以设置 Annotation 如下：

```
descheduler.alpha.kubernetes.io/evictable: 'true'
```

Network Policy 说明

最近更新时间：2022-04-18 11:44:16

简介

组件介绍

Network Policy 是 Kubernetes 提供的一种资源，用于定义基于 Pod 的网络隔离策略。它描述了一组 Pod 是否可以与其他组 Pod，以及其他 Network Entities 进行通信。本组件提供了针对该资源的 Controller 实现。如果您希望在 IP 地址或端口层面（OSI 第3层或第4层）控制特定应用的网络流量，则可考虑使用本组件。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	请求资源	所属 Namespace
networkpolicy	DaemonSet	每个实例CPU:250m，Memory:250Mi	kube-system
networkpolicy	ClusterRole	-	kube-system
networkpolicy	ClusterRoleBinding	-	kube-system
networkpolicy	ServiceAccount	-	kube-system

操作步骤

1. 登录 容器服务控制台 ，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 NetworkPolicy。NetworkPolicy 详细配置可参见 Network Policy 最佳实践。
5. 单击**完成**即可创建组件。

Nginx-ingress 说明

最近更新时间：2022-01-17 14:32:36

简介

组件介绍

Nginx 可以用作反向代理、负载平衡器和 HTTP 缓存。Nginx-ingress 组件是使用 Nginx 作为反向代理和负载平衡器的 Kubernetes 的 Ingress 控制器。您可以部署 Nginx-ingress 组件，在集群中使用 Nginx-ingress。

部署在集群内的 Kubernetes 对象

在集群内部署 Nginx-ingress Add-on，将在集群内部署以下 Kubernetes 对象：

Kubernetes 对象名称	类型	默认占用资源	所属 Namespaces
nginx-ingress	Service	-	自定义设置
nginx-ingress	Configmap	-	自定义设置
tke-ingress-nginx-controller-operator	Deployment	0.13核 CPU，128MB内存	kube-system
ingress-nginx-controller	Deployment/DaementSet	0.1核 CPU	kube-system
ingress-nginx-controller-hpa	HPA	-	kube-system

前提条件

- Kubernetes 版本建议在1.12版本及以上。
- 建议您使用 TKE 节点池功能。
- 建议您使用 TKE 云原生监控功能。
- 建议您使用 腾讯云日志服务 CLS。

使用方法

- Nginx-ingress 概述
- Nginx-ingress 安装
- 使用 Nginx-ingress 对象接入集群外部流量
- Nginx-ingress 监控配置
- Nginx-ingress 日志配置

OLM 说明

最近更新时间：2022-04-18 14:17:01

简介

组件介绍

OLM（Operator Lifecycle Manager）作为 Operator Framework 的一部分，可以帮助用户进行 Operator 的自动安装，升级及生命周期的管理。同时 OLM 自身以 Operator 的形式进行安装部署，其工作方式是以 Operators 来管理 Operators，且面向 Operator 提供的声明式（declarative）自动化管理能力完全符合 Kubernetes 交互的设计理念。

组件原理

OLM 由两个 Operator 构成：OLM Operator 和 Catalog Operator，其分别管理以下几个基础 CRD 模型：

资名称	简称	所属 Operator	描述
ClusterServiceVersion	csv	OLM	业务应用元数据，包括应用名称、版本、图标、依赖资源、安装方式等。
InstallPlan	ip	Catalog	计算自动安装或升级 CSV 过程中需要创建的资源集。
CatalogSource	catsrc	Catalog	用于定义应用的 CSVs、CRDs、安装包的仓库。
Subscription	sub	Catalog	通过跟踪安装包中的 channel 保证 CSVs 的版本更新。
OperatorGroup	og	OLM	用于 Operators 安装过程中的多租户配置，可以定义一组目标 namespaces 指定创建 Operators 所需的 RBAC 等资源配置。

- 在 Operator 安装管理生命周期中的 Deployment、Serviceaccount、RBAC 相关的角色和角色绑定通过 OLM operator 创建。Catalog Operator 负责 CRDs 和 CSVs 等资源的创建。
- OLM Operator 的工作基于 ClusterServiceVersion，一旦 CSV 中声明的依赖资源在目标集群中注册成功，OLM Operator 将负责安装这些资源对应的应用实例。
- Catalog Operator 主要负责解析 CSV 中声明的依赖资源定义，同时通过监听 catalog 中安装包对应 channels 的版本定义完成 CSV 对应的版本更新。

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	请求资源	所属 Namespace
catalogsources.operators.coreos.com	CustomResourceDefinition	-	-
clusterserviceversions.operators.coreos.com	CustomResourceDefinition	-	-
installplans.operators.coreos.com	CustomResourceDefinition	-	-
operatorgroups.operators.coreos.com	CustomResourceDefinition	-	-
operators.operators.coreos.com	CustomResourceDefinition	-	-
subscriptions.operators.coreos.com	CustomResourceDefinition	-	-
olm-operator	Deployment	cpu request: 10m memory request: 160Mi	operator-lifecycle-manager
catalog-operator	Deployment	cpu request: 10m memory request: 80Mi	operator-lifecycle-manager
system:controller:operator-lifecycle-manager	ClusterRole	-	-
aggregate-olm-view	ClusterRole	-	-

Kubernetes 对象名称	类型	请求资源	所属 Namespace
aggregate-olm-edit	ClusterRole	-	-
olm-operator-binding-operator-lifecycle-manager	ClusterRoleBinding	-	-
olm-operator	ServiceAccount	-	operator-lifecycle-manager
operators	Namespace	-	-
operator-lifecycle-manager	Namespace	-	-
packageserver	ClusterServiceVersion	-	operator-lifecycle-manager
olm-operators	OperatorGroup	-	operator-lifecycle-manager
global-operators	OperatorGroup	-	operators

使用场景

OLM 可以帮助用户安装、更新和管理所有 Operator 的生命周期。

风险控制

OLM 组件卸载后，为了保证用户的业务不会被影响，通过 OLM 部署的 Operator 不会被清理，并且该组件相关的 CRD 资源也不会被清理，此类 CRD 资源可以通过手动方式进行删除。

限制条件

> 您在创建集群时选择1.12.4以上版本集群，无需修改任何参数，开箱可用。

- 仅支持1.12版本以上的 kubernetes。
- 需设置 kube-apiserver 的启动参数：--feature-gates=CustomResourceSubresources=true

操作步骤

1. 登录 容器服务控制台 ，在左侧导航栏中选择**集群**。
2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 OLM。
5. 单击**完成**即可创建组件。

HPC 说明

最近更新时间：2022-06-09 14:47:31

简介

组件介绍

HPC（HorizontalPodCronscaler）是一种可以对 K8S workload 副本数进行定时修改的自研组件，配合 HPC CRD 使用，最小支持秒级的定时任务。

组件功能

- 支持设置“实例范围”（关联对象为 HPA）或“目标实例数量”（关联对象为 deployment 和 statefulset）。
- 支持开关“例外时间”。例外时间的最小配置粒度是日期，支持设置多条。
- 支持设置定时任务是否只执行一次。

部署在集群内的 Kubernetes 对象

在集群内部署 HPC，将在集群内部署以下 Kubernetes 对象：

Kubernetes 对象名称	类型	默认占用资源	所属Namespaces
horizontalpodcronscalers.autoscaling.cloud.tencent.com	CustomResourceDefinition	-	-
hpc-leader-election-role	Role	-	kube-system
hpc-leader-election-rolebinding	RoleBinding	-	kube-system
hpc-manager-role	ClusterRole	-	-
hpc-manager-rolebinding	ClusterRoleBinding	-	-
cronhpa-controller-manager-metrics-service	Service	-	kube-system
hpc-manager	ServiceAccount	-	kube-system
tke-hpc-controller	Deployment	100mCPU/pod、100Mi/pod	kube-system

限制条件

环境要求

> 您在创建集群时选择1.12.4以上版本集群，无需修改任何参数，开箱可用。

- 仅支持1.12版本以上的 kubernetes。
- 需设置 kube-apiserver 的启动参数：--feature-gates=CustomResourceSubresources=true

节点要求

- HPC 组件默认挂载主机的时区将作为定时任务的参考时间，因此要求节点存在 /etc/localtime 文件。
- HPC 默认安装2个 HPC Pod 在不同节点，因此节点数推荐为2个及以上。

被控资源要求

在创建 HPC 资源时，被控制的 workload（K8S 资源）需要存在于集群中。

操作步骤

安装 HPC

1. 登录 容器服务控制台 ，在左侧导航栏中选择集群。

2. 在“集群管理”页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入 “组件列表” 页面。
4. 在“组件列表”页面中选择**新建**，并在“新建组件”页面中勾选 HPC。
5. 单击**完成**即可创建组件。

创建并使用 HPC 工作负载示例

创建关联 Deployment 的定时任务资源

示例如下：

```
apiVersion: autoscaling.cloud.tencent.com/v1
kind: HorizontalPodCronscaler
metadata:
name: hpc-deployment
namespace: default
spec:
scaleTarget:
apiVersion: apps/v1
kind: Deployment
name: nginx-deployment
namespace: default
crons:
- name: "scale-down"
excludeDates:
- "* * * 15 11 *"
- "* * * * * 5"
schedule: "30 */1 * * * *"
targetSize: 1
- name: "scale-up"
excludeDates:
- "* * * 15 11 *"
- "* * * * * 5"
schedule: "0 */1 * * * *"
targetSize: 3
```

创建关联 StatefulSet 的定时任务资源

示例如下：

```
apiVersion: autoscaling.cloud.tencent.com/v1
kind: HorizontalPodCronscaler
metadata:
name: hpc-statefulset
namespace: default
spec:
scaleTarget:
apiVersion: apps/v1
kind: Statefulset
name: nginx-statefulset
namespace: default
crons:
- name: "scale-down"
excludeDates:
- "* * * 15 11 *"
schedule: "0 */2 * * * *"
targetSize: 1
```

```
- name: "scale-up"
excludeDates:
- "* * * 15 11 *"
schedule: "30 */2 * * * *"
targetSize: 4
```

创建关联 HPA 的定时任务资源

示例如下：

```
apiVersion: autoscaling.cloud.tencent.com/v1
kind: HorizontalPodCronscaler
metadata:
labels:
controller-tools.k8s.io: "1.0"
name: hpc-hpa
spec:
scaleTarget:
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
name: nginx-hpa
namespace: default
crons:
- name: "scale-up"
schedule: "30 */1 * * * *"
minSize: 2
maxSize: 6
- name: "scale-down"
schedule: "0 */1 * * * *"
minSize: 1
maxSize: 5
```

定时时间设置参考

字段名称	是否必选	允许值范围	允许的特殊字符
Seconds	是	0 – 59	* / , –
Minutes	是	0 – 59	* / , –
Hours	是	0 – 23	* / , –
Day of month	是	1 – 31	* / , – ?
Month	是	1 – 12 或 JAN – DEC	* / , –
Day of week	是	0 – 6 或 SUN – SAT	* / , – ?

应用管理

概述

最近更新时间：2022-01-17 15:08:05

应用功能是指腾讯云容器服务（Tencent Kubernetes Engine，TKE）集成的 Helm 3.0 相关功能，为您提供创建 helm chart、容器镜像、软件服务等各种产品和服务的能力。已创建的应用将在您指定的集群中运行，为您带来相应的能力。

应用相关操作

- 应用管理
- 本地 Helm 客户端连接集群

应用管理

最近更新时间：2022-04-26 16:48:22

操作场景

本文介绍如何通过容器服务控制台 对应用进行创建、更新、回滚、删除操作。

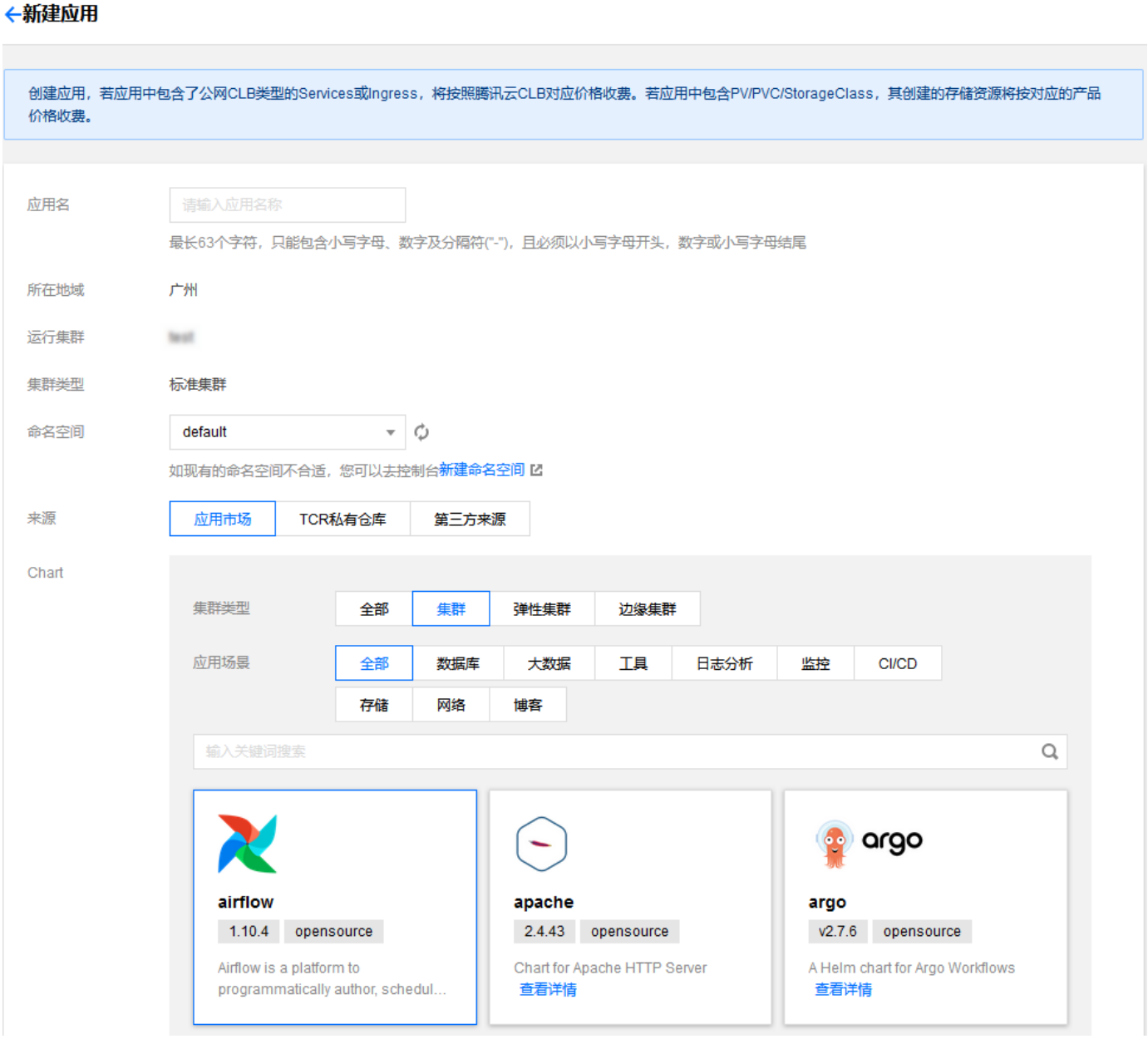
说明事项

应用管理仅支持 Kubernetes 1.8 版本以上集群。

操作步骤

创建应用

1. 登录容器服务控制台 ，选择左侧导航栏中的 **应用**。
2. 在“应用”列表页面上方，选择需创建应用的集群及地域，并单击**新建**。
3. 在“新建应用”页面中，参考以下信息设置应用的基本信息。如下图所示：



主要参数信息如下：

- **应用名**：自定义应用名称。
- **来源**：可选择**应用市场**、**TCR私有仓库**及**第三方来源**。详细配置见下表：

来源	配置项
应用市场	根据集群类型、应用场景进行 Chart 筛选。选择适用的应用包及 Chart 版本，并可编辑参数。
TCR私有仓库	▪ TCR实例名称：按需选择 腾讯云容器镜像服务 TCR 企业版实例。 ▪ 命名空间：按需选择指定 TCR 实例下命名空间。指定命名空间后，该命名空间下的 Chart 将会展示在应用列表页。 ▪ Chart 版本及参数：按需选择适用版本，并可编辑参数。
第三方来源	▪ Chart地址：支持 Helm 官方或自建 Helm Repo 仓库。注意必须设置为以 http 开头 .tgz 结尾的参数值。本文示例为 http://139.199.162.50/test/nginx-0.1.0.tgz。 ▪ 类型：提供**公有**及**私有**两种类型，请按需选择。 ▪ 参数：按需进行参数编辑。

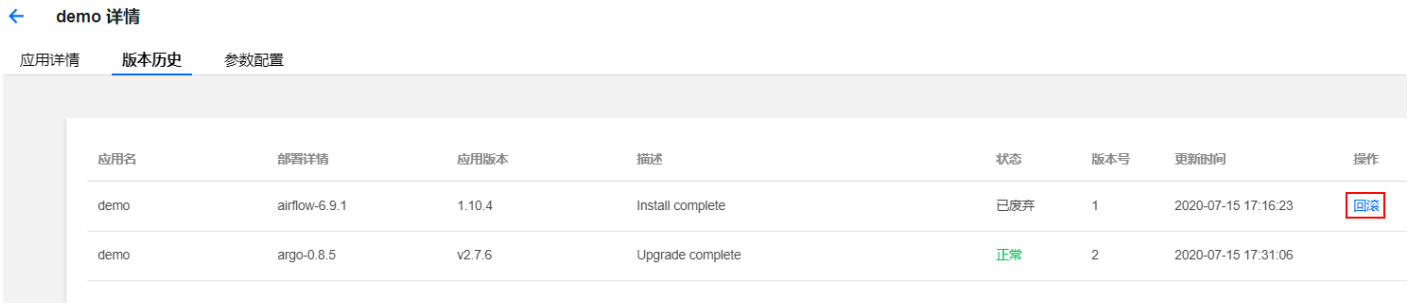
4. 单击**完成**即可创建应用。

更新应用

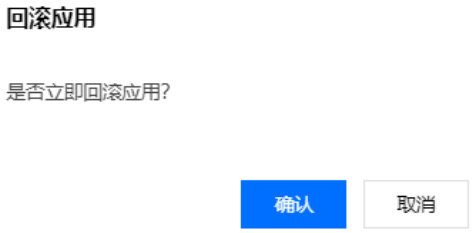
1. 前往 应用控制台，选择左侧导航栏中的**应用**，进入“应用”列表页面。
2. 在“应用”列表中，选择需更新的应用所在行右侧的**更新应用**。
3. 在弹出的 “更新应用” 窗口中，按需进行关键信息配置，并单击**完成**。

回滚应用

1. 前往 应用控制台，选择左侧导航栏中的**应用**，进入应用列表页面。
2. 在“应用”列表中，选择需要更新的应用名，进入该应用详情页面。
3. 在应用详情页中，选择**版本历史**页签，单击需回滚版本所在行右侧的**回滚**。如下图所示：



4. 在弹出的“回滚应用”窗口中，单击**确认**即可。如下图所示：



删除应用

1. 前往 应用控制台，选择左侧导航栏中的**应用**，进入应用列表页面。
2. 在“应用”列表中，选择需删除应用所在行右侧的**删除**。
3. 在弹出的 “删除应用” 窗口中，单击**确认**即可。

本地 Helm 客户端连接集群

最近更新时间：2022-04-12 16:49:37

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

依次执行以下命令，下载 Helm 客户端。关于安装 Helm 的更多信息，请参见 Installing Helm。

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
```

```
chmod 700 get_helm.sh
```

```
./get_helm.sh
```

配置 Helm Chart 仓库（可选）

1. 执行以下命令，配置 kubernetes 官方仓库。

```
helm repo add stable https://kubernetes-charts.storage.googleapis.com/
```

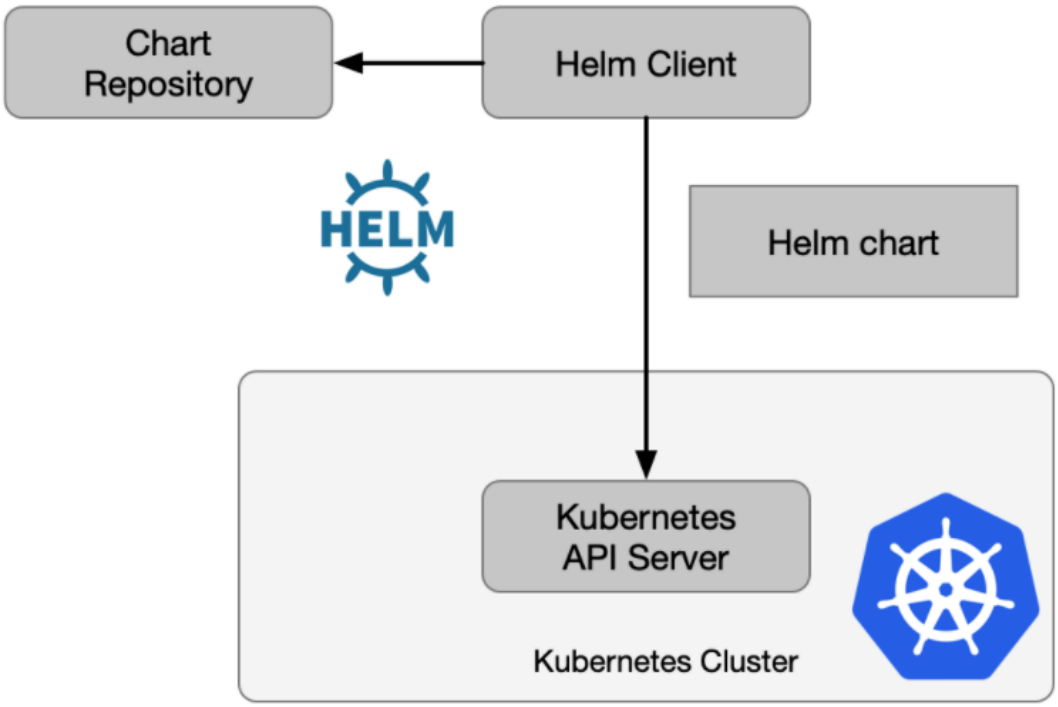
2. 执行以下命令，配置腾讯云应用市场。

```
helm repo add tkemarket https://market-tke.tencentcloudcr.com/chartrepo/opensource-stable
```

3. 配置 TCR 私有 Helm 仓库。

连接集群

Helm v3对比 Helm v2已移除 Tiller 组件，Helm 客户端可直接连接集群的 ApiServer，应用相关的版本数据直接存储在 Kubernetes 中。如下图所示：



Helm Client 使用 TKE 生成的客户端证书访问集群，具体操作步骤如下：

1. 通过 TKE 控制台或 API 获取可用公网或内网访问的 Kubeconfig。
2. 连接目标集群可参考以下两种方式：
   - 使用上述获取的 kubeconfig，对 Helm Client 所在机器的 kubectl config use-context 进行配置。
   - 执行以下命令，通过指定参数的形式访问目标集群。

```
helm install .... --kubeconfig [kubeconfig所在路径]
```

网络管理
容器网络概述

最近更新时间：2021-12-30 10:24:43

容器网络与集群网络说明

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。

容器网络与集群网络的关系

- **集群网络**：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 VPC 中的子网用于集群的节点网络。更多 VPC 的介绍可参见 VPC 概述。
- **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，包含 GlobalRouter 模式和 VPC-CNI 模式。
  - GlobalRouter 模式：您可以自定义三大私有网段作为容器网络，根据您选择的集群内服务数量的上限，自动分配适当大小的 CIDR 段用于 Kubernetes service。也可以根据您选择的每个节点的 Pod 数量上限，自动为集群内每台云服务器分配一个适当大小的网段用于该主机分配 Pod 的 IP 地址。
  - VPC-CNI 模式：选择与集群同 VPC 的子网用于容器分配 IP。

容器网络与集群网络的限制

- 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。

集群网络与腾讯云其他资源通信

- 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 TencentDB、云数据库 Redis、云数据库 Memcached 等资源在同一 VPC 下内网互通。

> ⚠ 注意
> - 集群内容器与同一 VPC 下其他资源连接时，请注意排查安全组是否已放通容器网段。
> - 容器服务 TKE 集群中的 ip-masq 组件使容器不能通过 SNAT 访问集群网络和 VPC 网络，而其他网段不受影响，因此容器访问同一 VPC 下其他资源（例如 Redis）时需要放通容器网段。

- 可 设置同地域集群间互通。
- 可 设置跨地域集群间互通。
- 可 设置容器集群与 IDC 互通。
- 可 设置 CVM 容器集群与黑石容器集群互通。

容器网络说明

- **容器 CIDR**：集群内 Service、Pod 等资源所在网段。
- **单节点 Pod 数量上限**：决定分配给每个 Node 的 CIDR 的大小。

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

- **集群内 Service 数量上限**：决定分配给 Service 的 CIDR 大小。

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

- **节点**：集群中 Worker 节点。

> 说明：
> 节点数计算公式为（CIDR IP 数量 - 集群内 Service 数量上限）/ 单节点 Pod 数量上限。

如何选择容器网络模式

容器服务 TKE 针对不同应用场景提供不同的网络模式。本文详细介绍了 TKE 提供的两种网络模式 GlobalRouter 和 VPC-CNI，以及从两者的使用场景、优势、使用限制等多个角度进行对比展示，您可以根据业务需要自行选择。

GlobalRouter 模式

GlobalRouter 网络模式是 TKE 基于底层私有网络（VPC）的全局路由能力，实现了容器网络和 VPC 互访的路由策略。详情可参见 GlobalRouter 模式介绍。

VPC-CNI 模式

VPC-CNI 模式是 TKE 基于 CNI 和 VPC 弹性网卡实现的容器网络能力，适用于对时延有较高要求的场景。该网络模式下，容器与节点分布在同一网络平面，容器 IP 为 IPAMD 组件所分配的弹性网卡 IP。详情可参见 VPC-CNI 模式介绍。

选择网络模式

本节从使用场景、优势、使用限制等多个角度出发，进行容器服务 TKE 所提供的 GlobalRouter、VPC-CNI 两种网络模式对比，请参考以下内容选择合适的网络模式：

角度	GlobalRouter	VPC-CNI
使用场景	• 普通容器业务场景。 • 离线计算相关业务。	• 对网络时延有较高要求的场景。 • 传统架构迁移到容器平台，依赖容器有固定 IP 的场景。
优势	• 容器路由直接经过 VPC，容器与节点分布在同一网络平面。 • 容器网段分配灵活，容器 IP 段不占用 VPC 的其他网段，可用 IP 资源丰富。	• ENI 的容器网络属于一个 VPC 子网，可纳入 VPC 产品的管理范围。 • 支持固定 IP、负载均衡（LB）直通 Pod 等用户场景。 • 网络性能优于 GlobalRouter 模式。
使用限制	• 专线、对等连接及云联网等互通场景需要额外配置。 • 不支持固定 Pod IP。	• 容器网络与节点网络属于同一个 VPC，IP 地址资源有限。 • 节点内容器数量受 弹性网卡和弹性网卡可分配 IP 数量的限制。 • 固定 IP 模式不支持容器与其他业务混用子网。 • 固定 IP 模式不支持 Pod 跨可用区调度。 • 网络规划需提前合理规划，后期调整困难。
具备额外的能力	标准 Kubernetes 功能。	• 容器服务支持固定 Pod IP。 • 容器网络在 VPC 控制台管控。 • LB 直接转发到 Pod，Pod 可以获取来源 IP。

GlobalRouter 模式
GlobalRouter 模式介绍

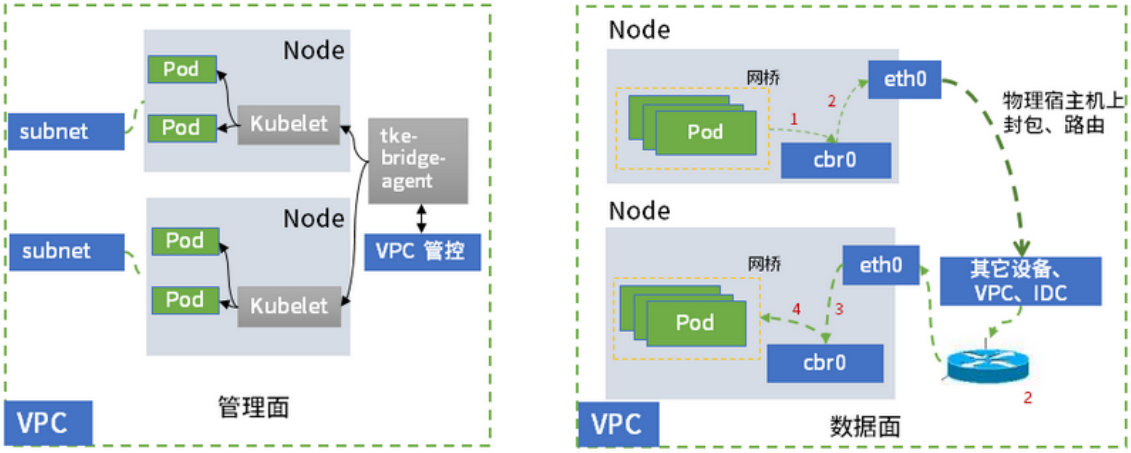
最近更新时间：2022-01-19 14:35:59

使用原理

GlobalRouter 网络模式是容器服务 TKE 基于底层私有网络 VPC 的全局路由能力，实现了容器网络和 VPC 互访的路由策略。该网络模式特征包含以下几点：

- 容器路由直接通过 VPC。
- 容器与节点分布在同一网络平面。
- 容器网段分配灵活，容器 IP 段不占用 VPC 的其他网段。

GlobalRouter 网络模式适用于常规场景，可与标准 Kuberentes 功能无缝使用。使用原理图如下所示：



使用限制

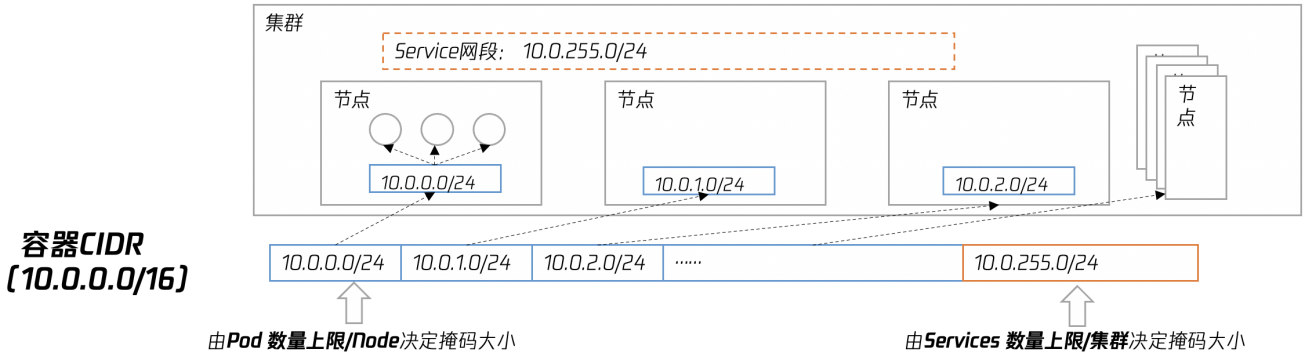
- 集群网络和容器网络网段不能重叠。
- 同一 VPC 内，不同集群的容器网络网段不能重叠。
- 容器网络和 VPC 路由重叠时，优先在容器网络内转发。
- 不支持固定 Pod IP。

容器 IP 分配机制

容器网络名词介绍和数量计算可参见 容器网络说明。

Pod IP 分配

工作原理如下图所示：



- 集群的每一个节点会使用容器 CIDR 中的指定大小的网段用于该节点下 Pod 的 IP 地址分配。
- 集群的 Service 网段会选用容器 CIDR 中最后一段指定大小的网段用于 Service 的 IP 地址分配。
- 节点释放后，使用的容器网段也会释放回 IP 段池。
- 扩容节点自动按顺序循环选择容器 CIDR 大段中可用的 IP 段。

同地域及跨地域 GlobalRouter 模式集群间互通

最近更新时间：2022-05-05 11:11:39

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，可以打通腾讯云上的资源通信链路。请参考 创建对等连接 建立对等连接，您可以通过对等连接实现**同地域和跨地域**的不同集群互通。

前提条件

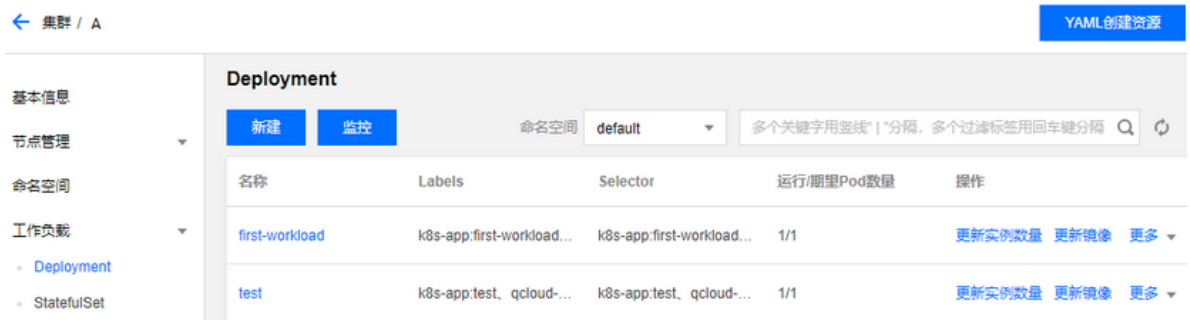
- 本文档以已创建集群并已添加节点为例。若未创建，请参考 创建集群 进行创建。
- 请先确认对等连接已成功建立，且子机间能互通。若对等连接建立有问题，请排查**控制台路由表项、CVM 安全组、子网 ACL** 的设置是否有问题。

操作步骤

> 说明：
> 如需实现跨地域集群间互通，请在执行完以下操作步骤后 在线咨询 申请打通容器路由，实现容器间互通。

获取容器的基本信息

1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 单击需要设置集群间互通的集群 ID/名称，进入该集群的管理页面。如下图所示：
   例如，进入 A 集群的管理页面。



3. 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：

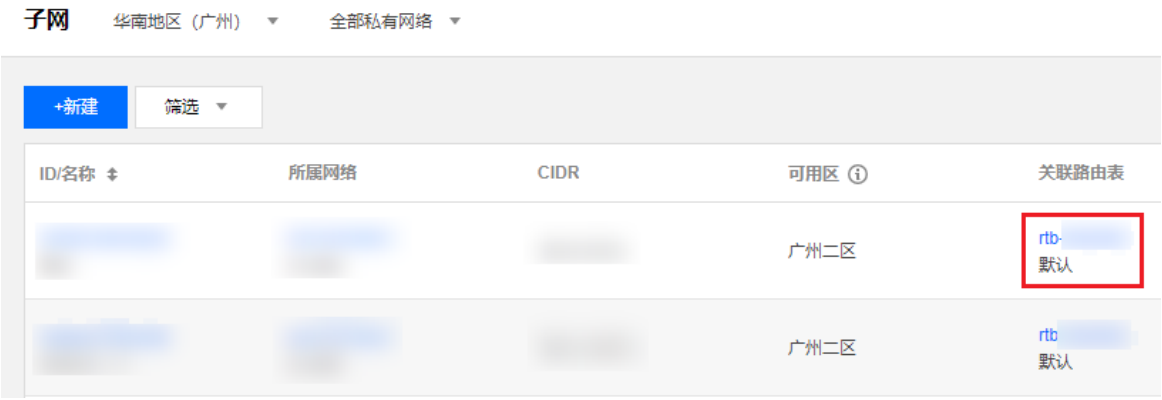
4. 记录 “基础信息” 中 “所在地域”、“节点网络” 和 “容器网络” 的信息。
5. 重复执行 步骤3 – 步骤5，记录另一个集群容器 “所在地域”、“节点网络” 和 “容器网络” 的信息。
   例如，记录 B 集群容器 “所在地域”、“节点网络” 和 “容器网络” 的信息。

配置路由表

1. 登录私有网络控制台，选择左侧导航栏中的 **对等连接**。
2. 在对等连接管理页面，记录对等连接的 **ID/名称**。如下图所示：

3. 选择左侧导航栏中的 **子网**，进入子网管理页面。

4. 单击对等连接本端指定子网的关联路由表。如下图所示：



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

预期结果

- **同地域集群**：通过上述操作可直接实现容器之间的互通。
- **跨地域集群**：对等连接建立成功后，请 在线咨询 打通容器路由，实现容器之间的互通。

请参考 远程终端基本操作 登录容器，并按照以下步骤进行容器间的访问，验证容器间是否互通：

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

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

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

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

GlobalRouter 模式集群与 IDC 互通

最近更新时间：2022-06-09 11:18:01

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**。

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

操作步骤

通过专线方式互通

1. 参考 申请物理专线，申请物理专线。
2. 参考 申请通道，申请通道。
3. 参考 创建专线网关，创建专线网关。
4. 验证容器节点与 IDC 互通。

> 注意：
>
> 执行此步骤时，请保证容器节点与 IDC 互通，验证通过。

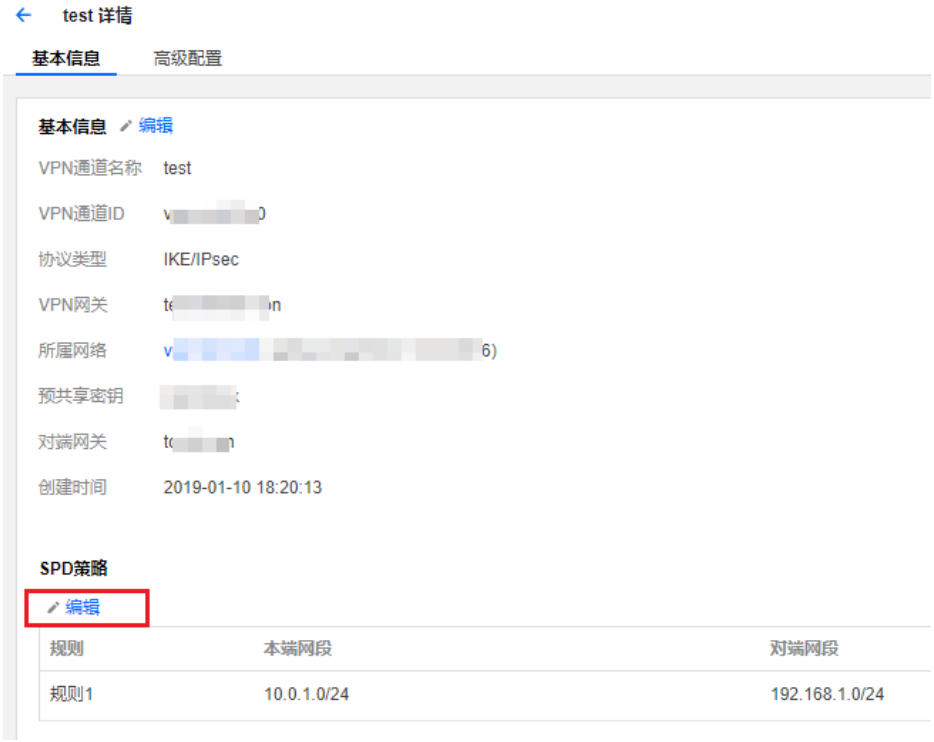
5. 准备地域，appID，集群 ID，vpcID，专线网关 ID 信息，提交工单 打通容器网络。
6. 根据 IDC 使用的协议类型，选择操作方式。
   - 若 IDC 使用的是 BGP 协议，容器网段路由将自动同步。
   - 若是其他协议，需在 IDC 内配置访问容器网段下一跳路由到专线网关。
7. 验证容器与 IDC 互通。

通过 VPN 方式互通

配置 SPD 策略

1. 登录 私有网络控制台。
2. 在左侧导航栏中，单击**VPN链接** > **VPN通道**，进入VPN 通道管理页面。
3. 单击需要配置 SPD 策略的本端 VPN 通道的 ID/名称。如下图所示：

4. 在 VPN 通道的详情页面，单击 “SPD策略” 栏下的**编辑**，添加容器网段。如下图所示：



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

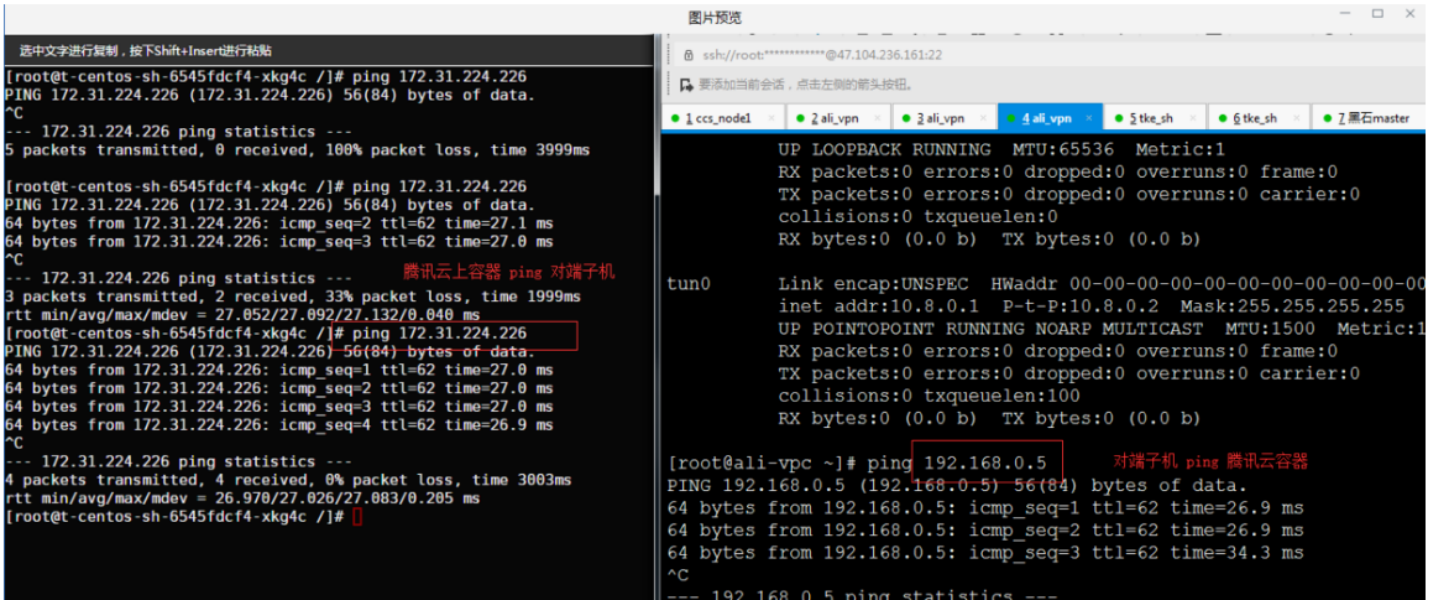
添加容器网段

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

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

预期结果

容器和对端子机可以互通。如下图所示：



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**。

注册 GlobalRouter 模式集群到云联网

最近更新时间：2022-06-09 11:26:47

云联网说明

云联网（Cloud Connect Network，CCN）为您提供云上私有网络间（Virtual Private Cloud，VPC）、VPC 与本地数据中心间互联的服务。您可以将 VPC 和专线网关实例加入云联网，以实现单点接入、全网资源互通，轻松构建简单、智能、安全、灵活的混合云及全球互联网络。

操作场景

您可以将已有容器集群注册到云联网，云联网可将容器网络归纳至管理范围中。当容器网络完成注册后，您可以在云联网侧启用或关闭容器网络的网段路由，实现容器集群与云联网内的资源互通。

> 注意：
> 当容器集群注册到云联网后，该网段与云联网实例中已有路由不冲突时开启，冲突时默认关闭。

前提条件

- 集群所在 VPC 已在云联网中，云联网相关操作请参见 云联网操作总览。
- 评估集群容器网络的网段是否与云联网网内其他资源网段冲突。

操作步骤

注册容器网络至对应云联网

1. 登录容器服务控制台，单击左侧导航栏中的 集群进入集群管理页面。
2. 选择需要进行云联网注册的集群 ID，单击左侧的基本信息进入集群基础信息页面。
3. 单击云联网的注册开关，将容器网络注册到云联网。如下图所示：

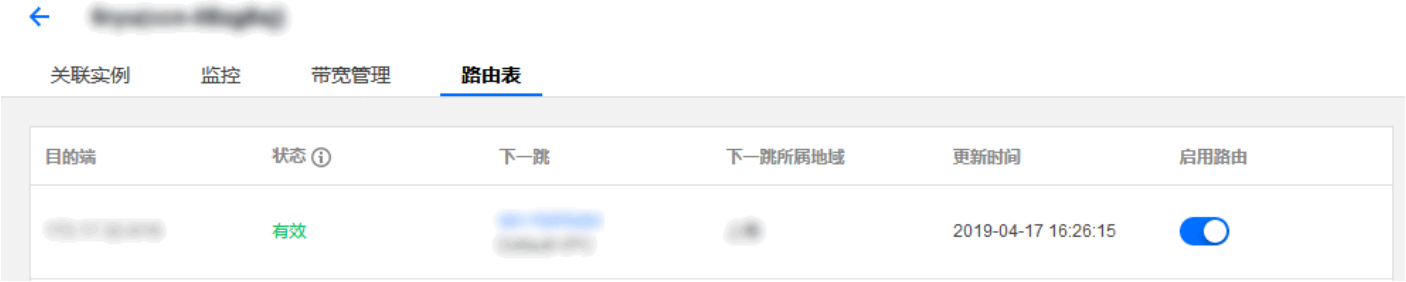
> 注意：
> 此步骤仅是将容器网段注册到云联网，路由是否生效，需要在云联网侧控制。

查看容器网段路由

1. 登录私有网络控制台，单击左侧导航栏中的 **云联网**进入云联网管理页面。
2. 单击集群 VPC 关联的云联网所在行右侧的**管理实例**，进入云联网实例管理页面。如下图所示：

3. 在云联网实例管理页面中，单击**路由表**页签，查看容器网段路由启动情况。如下图所示：

> 说明：
>
> - 若网段不冲突，则路由默认启动。网段冲突，则路由默认关闭。
> - 路由冲突原则请参见 路由限制，如需启动冲突路由，请参见 启用路由。



4. 可开始测试容器集群与云联网其他资源的互通性。

VPC-CNI 模式
VPC-CNI 模式介绍

最近更新时间：2022-06-09 11:38:08

使用原理

VPC-CNI 模式是容器服务 TKE 基于 CNI 和 VPC 弹性网卡实现的容器网络能力，适用于对时延有较高要求的场景。该网络模式下，容器与节点分布在同一网络平面，容器 IP 为 IPAMD 组件所分配的弹性网卡 IP。

其中 VPC-CNI 模式分为共享网卡模式和独占网卡模式，两种网络模式适用于不同的场景。您可以根据业务需要选择不同的网络模式。

- 共享网卡模式：Pod 共享一张弹性网卡，IPAMD 组件为弹性网卡申请多个 IP 给到不同的 Pod。可固定 Pod IP，详情请参见 固定 IP 模式使用说明。
- 独占网卡模式：每个 Pod 有独立的弹性网卡，性能更高。受机型影响，不同节点可使用的弹性网卡数量有限，单节点 Pod 密度更低。

使用限制

- 当前 VPC-CNI 模式的子网不能与其他云上资源共用（如云服务器、负载均衡等）。
- 集群内的节点需要和子网处于相同可用区，如果节点可用区与容器子网不在相同可用区，Pod 将无法调度。
- 节点上可调度的 VPC-CNI 模式的 Pod 数量受限于节点所支持插入弹性网卡能绑定 IP 的最大数量。配置越高的机器可插入的弹性网卡数量越多，可以通过查看节点的 Allocatable 来确认。

应用场景

相比 Global Router，VPC-CNI 具有以下优势及适用场景：

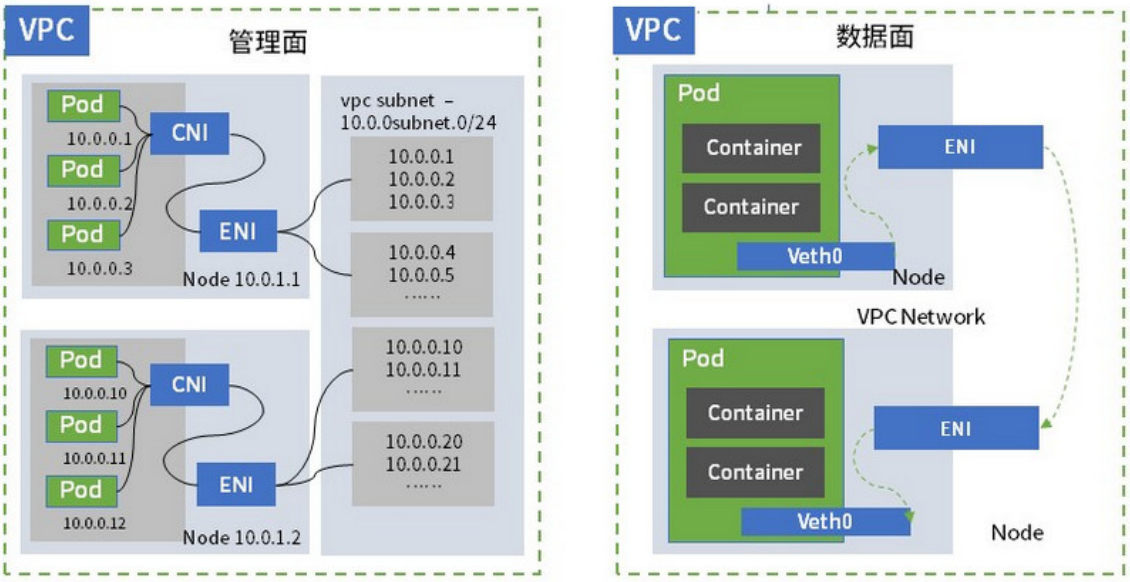
- 少了一层网桥，网络转发性能更高，大约提升10%，适用于对网络时延要求较高的场景。
- 支持 Pod 固定 IP，适用于依赖容器固定 IP 的场景。例如，传统架构迁移到容器平台及针对 IP 做安全策略限制。
- 支持 LB 直通 Pod。

多 Pod 共享网卡模式

最近更新时间：2022-06-09 11:37:43

使用原理

VPC-CNI 多 Pod 共享网卡模式使用原理图如下所示：



- 集群网络是用户的 VPC，节点和容器子网属于该 VPC。
- 容器子网可以选择多个 VPC 内的子网。
- 可设置是否开启固定 IP。您可参考 固定 IP 模式使用说明。

IP 地址管理原理

非固定 IP 模式

- TKE 组件在每个节点维护一个可弹性伸缩的 IP 池。已绑定的 IP 数量将被维持在 **Pod 数量 + 最小预绑定数量**及 **Pod 数量 + 最大预绑定数量**之间：
  - 当**已绑定数量 < Pod 数量 + 最小预绑定数量**时，会绑定 IP 使得**已绑定数量 = Pod 数量 + 最小预绑定数量**。
  - 当**已绑定数量 > Pod 数量 + 最大预绑定数量**时，会定时释放IP（约2分钟一次），直到**已绑定数量 = Pod 数量 + 最大预绑定数量**。
  - 当**最大可绑定数量 < 当前已绑定数量**时，会直接释放多余的空闲 IP，使得**已绑定数量 = 最大可绑定数量**。
- 共享网卡的 Pod 创建时，从节点可用 IP 池中随机分配一个可用 IP。
- 共享网卡的 Pod 销毁时，IP 释放回节点的 IP 池，留给下一个 Pod 使用，不会在 VPC 侧释放（删除）。
- IP 和网卡的分配和释放目前基于**最少网卡原则**，即保证使用的弹性网卡尽量的少：
  - **IP 分配给 Pod**：优先分配已分配 IP 数量最多的网卡上的 IP
  - **IP 释放**：优先释放已分配 IP 数量最少的网卡上的 IP
  - **新网卡绑定**：若当前已绑定网卡 IP 配额用尽或网卡所在的子网 IP 用完，则申请新网卡绑定 IP
  - **网卡释放**：若已绑定网卡的辅助 IP 都已解绑，且不再需要新增 IP，则解绑并删除网卡
- 节点会注册扩展资源 tke.cloud.tencent.com/eni-ip，资源的可分配数(Allocatable) 为实际的已绑定 IP 资源数，总量(Capacity) 为节点可绑定的 IP 资源上限。因此，当 Pod 调度到某节点失败时，说明节点的 IP 已用尽。
- 新网卡的子网选择：新网卡优先选择可用 ip 最多的子网。
- 各节点最大可绑定 IP = 最大绑定网卡数 * 单网卡可绑定 IP 数
- 当前**最小预绑定数量**和**最大预绑定数量**的默认值为5

固定 IP 模式

- TKE 网络组件维护一个集群维度的可用 IP 池。
- 集群每新增一个节点，将申请一张弹性网卡。不会提前绑定任何辅助 IP，但会在网络组件内给此节点预留网卡 IP 配额数量的 IP。
- 新建一个使用 VPC-CNI 模式的 Pod 时，IPAMD 组件会依据节点绑定网卡所在的子网分配1个 IP，然后才会即时申请绑定该辅助 IP 到相应节点的网卡上。

- Pod 销毁时，IP 地址回归集群的可用 IP 池，并触发网卡解绑 IP，IP 地址将释放回 VPC 子网内。
- 固定 IP 的 Pod 的 IP 仅在 TKE 集群内部保留，保证下一次创建同名 Pod 的时候仍使用这个 IP。
- 节点删除时，将释放网卡占用的 IP 资源。
- 多容器子网的情况下，网卡优先分配到可用 IP 数量最多且能完全满足网卡 IP 配额需求的子网内，若没有完全满足需求的子网，则节点绑定网卡失败。

多网卡数据面原理

> **注意：**
> 当前仅非固定 IP 模式支持多网卡。

当节点绑定了多张网卡时，Pod 发出的网络包遵循策略路由转发到对应的网卡上：

- 在节点上执行 ip link 可看到节点所有的网络设备信息，通过弹性网卡的 mac 地址比对，可知道其中弹性网卡对应的网络设备。一般情况下，eth0 为主网卡，eth1 、eth2 等为辅助弹性网卡：

```
[[root@VM-4-196-tlinux ~]# ip l
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 52:54:00:23:98:4a brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 20:90:6f:cf:6a:b9 brd ff:ff:ff:ff:ff:ff
4: eni4b1d9e7dd9b@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether da:53:24:6c:af:e4 brd ff:ff:ff:ff:ff:ff link-netnsid 0
5: eni43625ff3fd0@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 46:17:d2:59:ff:d9 brd ff:ff:ff:ff:ff:ff link-netnsid 1
6: eni206b44a5853@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 0e:0d:c6:3b:70:99 brd ff:ff:ff:ff:ff:ff link-netnsid 2
7: enia8afb31cbdb@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 12:1d:32:9d:8f:9f brd ff:ff:ff:ff:ff:ff link-netnsid 3
8: eni2435af570f1@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 42:a2:64:9a:3f:01 brd ff:ff:ff:ff:ff:ff link-netnsid 4
9: eni4ab50a1bd0b@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 5a:3e:c6:3e:5c:51 brd ff:ff:ff:ff:ff:ff link-netnsid 5
10: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 20:90:6f:97:cf:45 brd ff:ff:ff:ff:ff:ff
11: eni0a1cc6fca61@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 6a:4d:57:25:45:3c brd ff:ff:ff:ff:ff:ff link-netnsid 6
```

- 在节点上执行 ip rule 可看到策略路由表的信息，TKE 网络组件通过弹性网卡的 <link index>+2000 得到路由表号，绑定了对应网卡 IP 的 Pod 网络包都将转发到该路由表，如此例中，eth1 对应的路由表即为 2003，eth2 对应的路由表即为 2010：

```
[[root@VM-4-196-tlinux ~]# ip rule
0:      from all lookup local
512:    from all to 172.16.34.28 lookup main
512:    from all to 172.16.34.102 lookup main
512:    from all to 172.16.33.104 lookup main
512:    from all to 172.16.34.107 lookup main
512:    from all to 172.16.33.72 lookup main
512:    from all to 172.16.33.110 lookup main
512:    from all to 172.16.33.36 lookup main
512:    from all to 172.16.33.108 lookup main
512:    from all to 172.16.33.14 lookup main
512:    from all to 172.16.34.74 lookup main
512:    from all to 172.16.34.30 lookup main
512:    from all to 172.16.33.60 lookup main
1536:   from 172.16.34.28 lookup 2003
1536:   from 172.16.34.102 lookup 2003
1536:   from 172.16.33.104 lookup 2003
1536:   from 172.16.34.107 lookup 2003
1536:   from 172.16.33.72 lookup 2003
1536:   from 172.16.33.110 lookup 2003
1536:   from 172.16.33.36 lookup 2003
1536:   from 172.16.33.108 lookup 2003
1536:   from 172.16.33.14 lookup 2003
1536:   from 172.16.34.74 lookup 2010
1536:   from 172.16.34.30 lookup 2010
1536:   from 172.16.33.60 lookup 2010
32766:  from all lookup main
32767:  from all lookup default
```

- 对应的路由表则设置了到对应网卡的默认路由，节点上执行 ip route show table <id> 可查看：

```
[[root@VM-4-196-tlinux ~]# ip route show table 2003
default via 172.16.32.1 dev eth1 onlink
[[root@VM-4-196-tlinux ~]# ip route show table 2010
default via 172.16.33.1 dev eth2 onlink
```

而欲发送给 Pod 的网络包到达节点时，同样遵循策略路由，直接通过主路由表发送给 Pod 的 Veth 网卡。

使用方法

使用 VPC-CNI 需要确保 rp_filter 处于关闭状态。可参考以下代码示例：

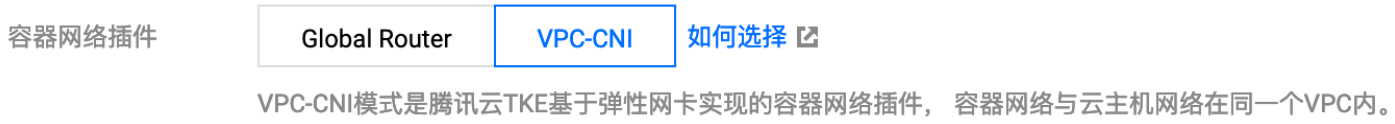
```
sysctl -w net.ipv4.conf.all.rp_filter=0
# 假设 eth0 为主网卡
sysctl -w net.ipv4.conf.eth0.rp_filter=0
```

> **注意：**
> tke-eni-agent 组件自动设置节点的内核参数。若您自己有维护内核参数且打开 rpfilter，则会导致网络不通。

开启 VPC-CNI

创建集群时开启 VPC-CNI

1. 登录 容器服务控制台，单击左侧导航栏中**集群**。
2. 在“集群管理”页面，单击集群列表上方的**新建**。
3. 在“创建集群”页面，在容器网络插件中选择 “VPC-CNI”。如下图所示：



> **说明：**
> 默认情况下，VPC-CNI 模式**不支持固定 Pod IP 能力**，且该能力仅支持在 创建集群 时设置。如需为集群开启支持固定 Pod IP，请参见 固定 IP 模式使用说明。

为已有集群开启 VPC-CNI

创建集群时选择 Global Router 网络插件，后续在集群基本信息页面开启 VPC-CNI 模式（两种默认混用）。

1. 登录 容器服务控制台，单击左侧导航栏中**集群**。
2. 在“集群管理”页面，选择需开启 VPC-CNI 的集群 ID，进入集群详情页。
3. 在集群详情页面，选择左侧**基本信息**。
4. 在集群“基本信息”页面的集群信息模块，在 VPC-CNI 字段中单击开启。

5. 在弹出窗口中选择子网，并确认 IP 回收策略。如下图所示：

> 注意：
>
> - 针对固定 IP 场景，启用 VPC-CNI 后需要设置 IP 回收策略，即设置 Pod 销毁后需要退还 IP 的时长。
> - 非固定 IP 的 Pod 销毁后可立即释放 IP（非释放回 VPC，释放回容器管理的 IP 池），不受此设置的影响。

6. 单击**提交**，即可完成为已有集群开启 VPC-CNI。

关闭 VPC-CNI

1. 登录 容器服务控制台 ，单击左侧导航栏中**集群**。
2. 在“集群管理”页面，选择需开启 VPC-CNI 的集群 ID，进入集群详情页。
3. 在集群详情页面，选择左侧**基本信息**。
4. 在集群“基本信息”页面的集群信息模块，在 VPC-CNI 字段中单击关闭。
5. 在弹出窗口中选择**提交**，即可关闭 VPC-CNI。

Pod 间独占网卡模式

最近更新时间：2022-06-09 14:32:40

Pod 间独占网卡模式在原有 VPC-CNI 模式单网卡多 IP 模式的基础上，进阶为容器直接独享使用弹性网卡。无缝对接腾讯云私有网络产品的全部功能，同时在性能做了极大的提升。

> ⚠ 注意：
> 目前该功能正处于内测阶段，您可通过 内测申请 开通使用。

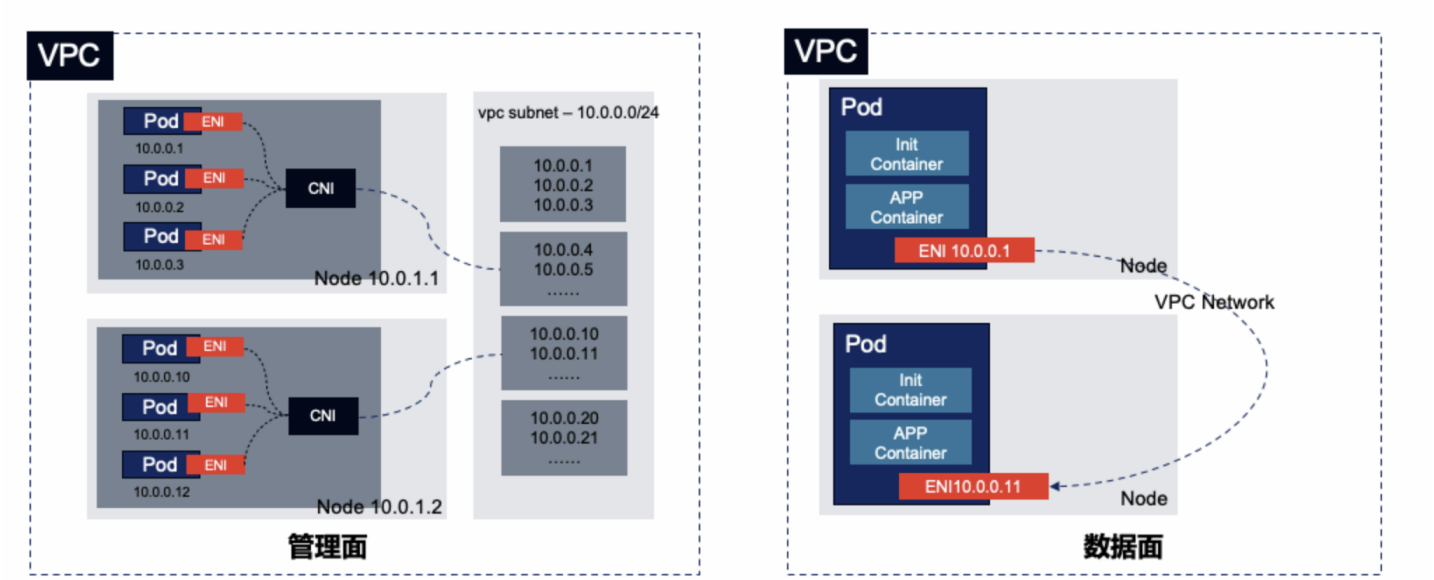
功能简介

新一代 VPC-CNI 模式的网络方案中，能够在原有的网络能力中额外增加以下能力：

- 支持 Pod 绑定 EIP/NAT，不再依赖节点的外网访问能力，无须做 SNAT，可以满足爬虫，视频会议等高并发，高带宽外网访问场景。
- 支持基于 Pod 名称的固定 IP，Pod 调度重启后仍能保证 IP 不变。
- 支持 CLB 直通 Pod，不再经过 NodePort 转发，提升转发性能并拥有统一的负载均衡视图。

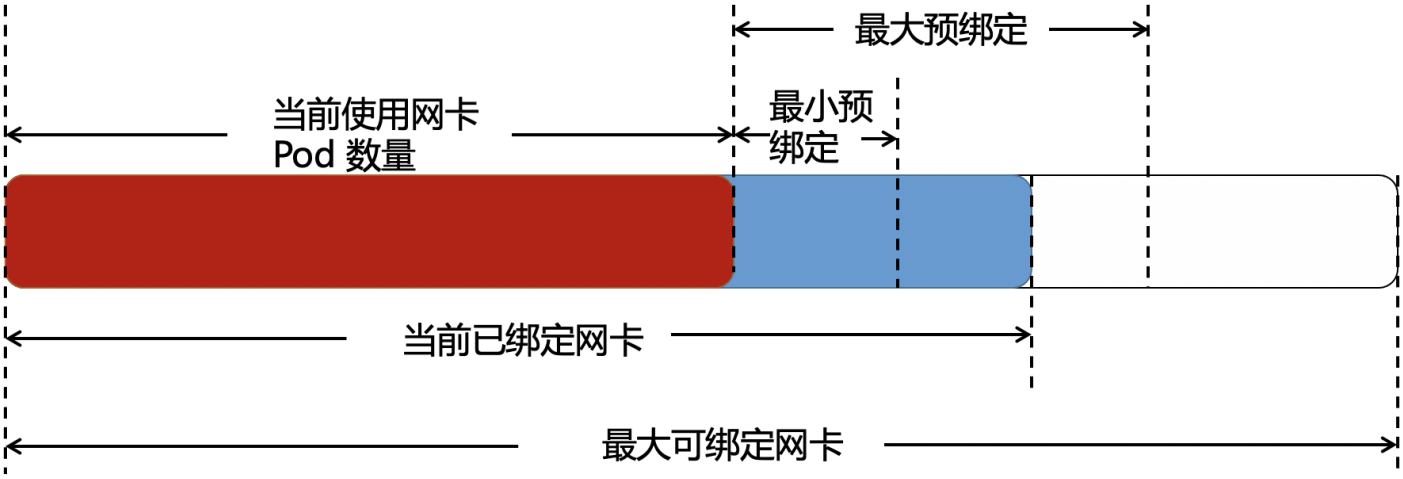
实现方式

新一代方案在原有 VPC-CNI 模式的基础上扩展，依托于弹性网卡，将绑定到节点的弹性网卡通过 CNI 配置到容器网络命名空间，实现容器直接独享使用弹性网卡。实现原理如下图所示：



IP 地址管理原理

非固定 IP 模式



- TKE 组件在每个节点维护一个可弹性伸缩的网卡池。已绑定的网卡数量将被维持在 **Pod 数量 + 最小预绑定数量**及 **Pod 数量 + 最大预绑定数量**之间：
  - 当已绑定数量 < Pod 数量 + 最小预绑定数量时，会绑定网卡使得已绑定数量 = Pod 数量 + 最小预绑定数量。
  - 当已绑定数量 > Pod 数量 + 最大预绑定数量时，会定时释放1个网卡（约2分钟一次），直到已绑定数量 = Pod 数量 + 最大预绑定数量。
  - 当最大可绑定网卡数量 < 当前已绑定数量时，会直接释放多余的空闲网卡，使得已绑定数量 = 最大可绑定数量。
- 独占网卡的 Pod 创建时，从节点可用网卡池中随机分配一个可用网卡。
- 独占网卡的 Pod 销毁时，网卡释放回节点的网卡池，留给下一个 Pod 使用，不会在 VPC 侧释放（删除）。
- 节点删除时，将释放（删除）所有已绑定的网卡。
- 多容器子网的情况下，网卡优先分配到可用 IP 数量最多的子网内。

固定 IP 模式

- TKE 不会为每个节点维护网卡池，网卡不会预绑定到节点上。
- 独占网卡的 Pod 创建时，直接绑定一张网卡到节点上，给这个 Pod 使用。
- 非固定 IP 的独占网卡 Pod 销毁时，直接在 VPC 侧删除释放该 Pod 使用的网卡，固定 IP 的 Pod 销毁时，网卡仅做解绑，不会删除释放。
- 节点删除时，将释放（删除）所有已绑定的网卡。
- 多容器子网的情况下，网卡优先分配到可用 IP 数量最多的子网内。

功能限制

- 仅支持 S5、SA2、IT5、SA3 等部分机型使用该网络模式。
- 节点上运行的独立网卡方案的 Pod 数量限制受到机型可绑定弹性网卡数量的影响。其最大数量为最大可绑定弹性网卡数量 - 1，详见 VPC-CNI 模式 Pod 数量限制。
- 仅支持新集群，存量容器服务 TKE 集群暂不支持变更网络方案。
- 有 VPC-CNI 模式的统一限制：
  - 需要为容器专门规划子网，子网不建议其他云上资源共用（如云服务器、负载均衡等）。
  - 集群内的节点需要和子网处于相同可用区，如果节点可用区与容器子网不在相同可用区，Pod 将无法调度。

固定 IP 模式使用说明
固定 IP 使用方法

最近更新时间：2022-06-09 11:27:01

使用场景

适用于依赖容器固定 IP 的场景。例如，传统架构迁移到容器平台及针对 IP 做安全策略限制。 对 IP 无限制的业务不推荐您使用固定 IP 模式。

能力和限制

- 支持 Pod 销毁 IP 保留，Pod 迁移 IP 不变，从而实现固定 IP。
- 支持多子网，但不支持跨子网调度固定 IP 的 Pod， 因此固定 IP 模式的 Pod 不支持跨可用区调度。
- 支持 Pod IP 自动关联弹性公网 IP，从而可支持 Pod 外访。
- 共享网卡的固定 IP 模式，固定 IP 的 Pod 销毁后，其 IP 只在集群范围内保留。若有其他集群或者业务（如 CVM、CDB、CLB 等）使用了同一子网，可能会导致保留的固定 IP 被占用，Pod 再启动时将无法获取 IP。**因此请保证该模式的容器子网是独占使用。**

使用方法

您可以通过以下两种方式启用固定 IP：

- 创建集群选择固定 IP 模式的 VPC-CNI。
- 为 GlobalRouter 模式附加固定 IP VPC-CNI 模式。

创建集群选择固定 IP 模式的 VPC-CNI

> 说明：
> 使用此方式启用 VPC-CNI，通过控制台或通过 yaml 创建工作负载，Pod 均默认使用弹性网卡。

1. 登录 容器服务控制台 ，单击左侧导航栏中**集群**。
2. 在“集群管理”页面，单击集群列表上方的**新建**。
3. 在“创建集群”页面，在容器网络插件中选择 “VPC-CNI” 。
4. 选择“容器网络插件”为**VPC-CNI**，并勾选“开启支持”固定 Pod IP 即可。如下图所示：

为 GlobalRouter 模式附加固定 IP VPC-CNI 模式

为已有集群开启 VPC-CNI

> 说明：
> - 为 GlobalRouter 模式附加固定 IP VPC-CNI 模式即创建集群时选择 Global Router 网络插件，后续在集群基本信息页面开启 VPC-CNI 模式（两种模式默认混用）。
> - 使用此方式启用 VPC-CNI，Pod 默认不使用弹性网卡。

1. 登录 容器服务控制台 ，单击左侧导航栏中**集群**。
2. 在“集群管理”页面，选择需开启 VPC-CNI 的集群 ID，进入集群详情页。
3. 在集群详情页面，选择左侧**基本信息**。
4. 在集群“基本信息”页面的集群信息模块，在 VPC-CNI 字段中单击开启。

5. 在弹出窗口中选择子网，并确认 IP 回收策略。如下图所示：

> ⚠ 注意：
>
> - 针对固定 IP 场景，启用 VPC-CNI 后需要设置 IP 回收策略，即设置 Pod 销毁后需要退还 IP 的时长。
> - 非固定 IP 的 Pod 销毁后可立即释放 IP（非释放回 VPC，释放回容器管理的 IP 池），不受此设置的影响。

6. 单击**提交**，即可完成为已有集群开启 VPC-CNI。

创建固定 Pod IP 类型 StatefulSet

在 GlobalRouter 模式附加 VPC-CNI 模式下，如果您存在业务需要在容器服务 TKE 中部署，并存在固定 Pod IP 的需求，您可以使用固定 IP 类型的 StatefulSet。TKE 提供扩展 StatefulSet 固定 IP 的能力，该类型的 StatefulSet 创建的 Pod 将通过弹性网卡分配真实的 VPC 内的 IP 地址。容器服务 TKE VPC-CNI 的插件负责 IP 分配，当 Pod 重启或迁移，可实现 IP 地址不变。

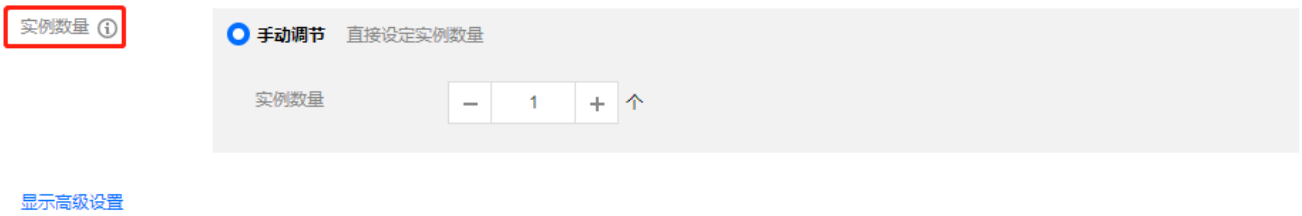
您可以通过创建固定 IP 类型 StatefulSet 来满足以下场景：

- 通过来源 IP 授权。
- 基于 IP 做流程审核。
- 基于 Pod IP 做日志查询等。

> ⚠ 注意：
>
> 固定 IP 类型 StatefulSet 存在使用限制，仅支持 StatefulSet 生命周期内固定 IP。

您可通过以下两种方法创建固定 IP：

- 通过控制台创建固定 IP 类型 StatefulSet
  i. 登录 容器服务控制台 ，单击左侧导航栏中**集群**。
  ii. 选择需要使用固定 IP 模式的集群 ID 名称，进入该集群的管理页面。
  iii. 选择**工作负载** > **StatefulSet**，进入**StatefulSet**的集群管理页面。
  iv. 单击**新建**，查看**实例数量**。如下图所示：



v. **单击显示高级设置，根据您实际需求，设置StatefulSet参数。关键参数信息如下：**

- **网络模式：勾选使用 VPC-CNI 模式。**
  - **IP 地址范围：目前仅支持随机。**
  - **固定 Pod IP：选择开启。**

- **通过 Yaml 创建**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
labels:
k8s-app: busybox
name: busybox
namespace: default
spec:
replicas: 3
selector:
matchLabels:
k8s-app: busybox
qcloud-app: busybox
serviceName: ""
template:
metadata:
annotations:
tke.cloud.tencent.com/networks: "tke-route-eni"
tke.cloud.tencent.com/vpc-ip-claim-delete-policy: Never
creationTimestamp: null
labels:
k8s-app: busybox
qcloud-app: busybox
spec:
containers:
```

```
- args:
- "10000000000"
command:
- sleep
image: busybox
imagePullPolicy: Always
name: busybox
resources:
limits:
tke.cloud.tencent.com/eni-ip: "1"
requests:
tke.cloud.tencent.com/eni-ip: "1"
```

- **spec.template.annotations：**tke.cloud.tencent.com/networks: "tke-route-eni" **表明 Pod 使用共享网卡的 VPC-CNI 模式，如果使用的是独立网卡的 VPC-CNI 模式，请将值修改成** "tke-direct-eni"。
- **spec.template.annotations：创建 VPC-CNI 模式的 Pod，您需要设置 annotations，即** tke.cloud.tencent.com/vpc-ip-claim-delete-policy，**默认是 “Immediate”，Pod 销毁后，关联的 IP 同时被销毁。如需固定 IP，则需设置成 “Never”，Pod 销毁后 IP 也将会保留，那么下一次同名的 Pod 拉起后，会使用之前的 IP。**
- **spec.template.spec.containers.0.resources：创建共享网卡的 VPC-CNI 模式的 Pod，您需要添加 requests 和 limits 限制，即** tke.cloud.tencent.com/eni-ip。**如果是独立网卡的 VPC-CNI 模式，则添加** tke.cloud.tencent.com/direct-eni。

固定 IP 相关特性

最近更新时间：2022-06-09 14:33:11

固定 IP 的保留和回收

固定 IP 模式下，创建使用 VPC-CNI 模式的 Pod 以后，网络组件会为该 Pod 在同 namespace 下创建同名的 CRD 对象 VpcIPClaim 。该对象描述 Pod 对 IP 的需求。网络组件随后会根据这个对象创建 CRD 对象 VpcIP ，并关联对应的 VpcIPClaim 。VpcIP 以实际的 IP 地址为名，表示实际的 IP 地址占用。

您可以通过以下命令查看集群使用的容器子网内 IP 的使用情况：

```
kubectl get vip
```

对于非固定 IP 的 Pod，其 Pod 销毁后 VpcIPClaim 也会被销毁，VpcIP 随之销毁回收。而对于固定 IP 的 Pod，其 Pod 销毁后 VpcIPClaim 仍然保留，VpcIP 也因此保留。同名的 Pod 启动后会使用同名的 VpcIPClaim 关联的 VpcIP ，从而实现 IP 地址保留。

由于网络组件在集群范围内分配 IP 时会依据 VpcIP 信息找寻可用 IP，因此固定 IP 的地址若不使用需要及时回收（目前默认策略是永不回收），否则会导致 IP 浪费而无 IP 可用。本文介绍过期回收、手动回收及级联回收的 IP 回收方法。

过期回收（默认支持）

在 创建集群 页面，容器网络插件选择**VPC-CNI模式并且勾选开启支持**固定Pod IP 支持，如下图所示：

在高级设置中设置 IP 回收策略，可以设置 Pod 销毁后多少秒回收保留的固定 IP。如下图所示：

手动回收

对于急需回收的 IP 地址，需要先确定需回收的 IP 被哪个 Pod 占用，找到对应的 Pod 的名称空间和名称，执行以下命令通过手动回收：

> ⚠ 注意：
> 需保证回收的 IP 对应的 Pod 已经销毁，否则会导致该 Pod 网络不可用。

```
kubectl delete vipc <podname> -n <namespace>
```

级联回收

目前的固定 IP 与 Pod 强绑定，而与具体的 Workload 无关（例如 deployment、statefulset 等）。Pod 销毁后，固定 IP 不确定何时回收。TKE 现已实现删除 Pod 所属的 Workload 后即刻删除固定 IP。

以下步骤介绍如何开启级联回收：

1. 修改现存的 **tke-eni-ipamd deployment**：**kubectl edit deploy tke-eni-ipamd -n kube-system**。
2. 执行以下命令，在 spec.template.spec.containers[0].args 中加入启动参数：

```
- --enable-ownerref
```

修改后，ipamd 会自动重启并生效。生效后，增量 Workload 可实现级联删除固定 IP，存量 Workload 暂不能支持。

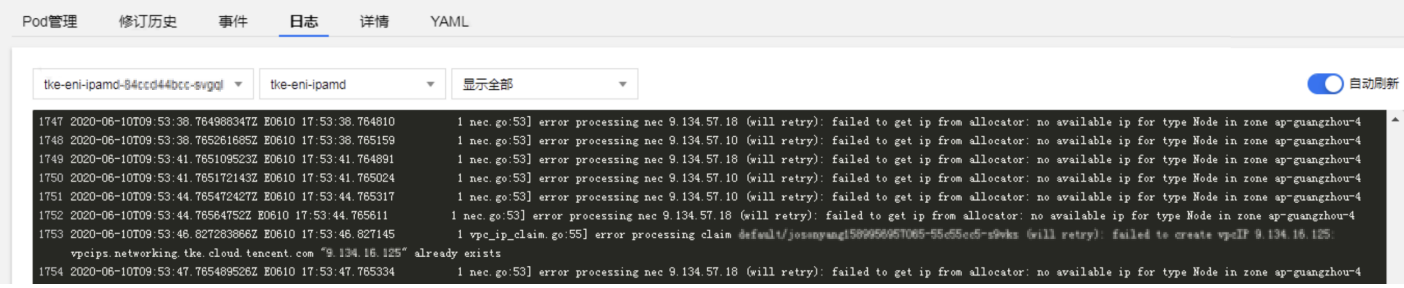
相关问题

节点不能分配到弹性网卡，无法正常调度 Pod（共享网卡模式）

当节点加入到集群后，ipamd 会尝试从和节点相同可用区的子网（配置给 ipamd 的子网）中为节点绑定一个弹性网卡，如果 ipamd 异常或者没有给 ipamd 配置和节点相同可用区的子网，ipamd 将无法给节点分配辅助网卡。此外，如果当前 VPC 使用的辅助网卡数目超过上限，则无法给节点分配辅助网卡。
执行以下命令，确认问题原因：

```
kubectl get event
```

- event 中显示 ENILimit，则是配额问题，可以通过为 VPC 调大弹性网卡数目配额来解决问题。
- event 中显示下图信息则说明子网中的 IP 不足。



可以通过执行以下命令获取当前子网 IP 使用数目。

```
kubectl get vip | wc -l
```

若已确认子网 IP 充足，但仍存在问题，则可能与底层的软限制有关。分析如下：
以高配机型（每个节点关联的弹性网卡可以额外分配29个 IP）和配置的子网是 /23 为例，当集群有17个节点时，这些节点上理论能使用的 IP 资源为 17 *（29 + 1），即已经超过500了，可把 /23 的子网填满，此时 ipamd 会限制新的节点不再分配弹性网卡。为解决这个问题，可再添加一些子网，弹性网卡可以从新子网中创建并绑定到新添加的节点上，新的节点虽然能够加入集群，但是 Pod 不会调度到没有绑定弹性网卡的节点。
如果不加限制，节点越加越多会导致能分给 Pod 的 IP 越来越少，因为弹性网卡本身会占用一个主 IP，这个主 IP 不能用于 Pod，所以添加一个节点，实际上子网可以分给 Pod 的 IP 会少一个。在极端情况下，存在分配给节点的辅助网卡都集中在一个子网中的情况，会限制整个集群中 Pod 的规模，并且只能通过驱逐旧的节点，添加子网后，再将节点加入集群才能恢复。

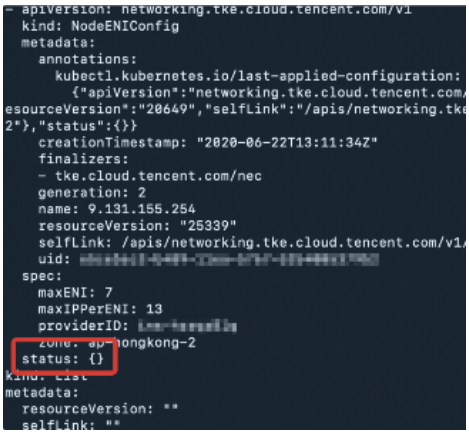
节点不能分配到弹性网卡，提示弹性网卡数量超出限制

现象

节点配置的弹性网卡无法绑定，nec 关联的 vip attach 失败。查看 nec 则看到节点关联的 nec status 为空。
执行以下代码可查看 nec：

```
kubectl get nec -o yaml
```

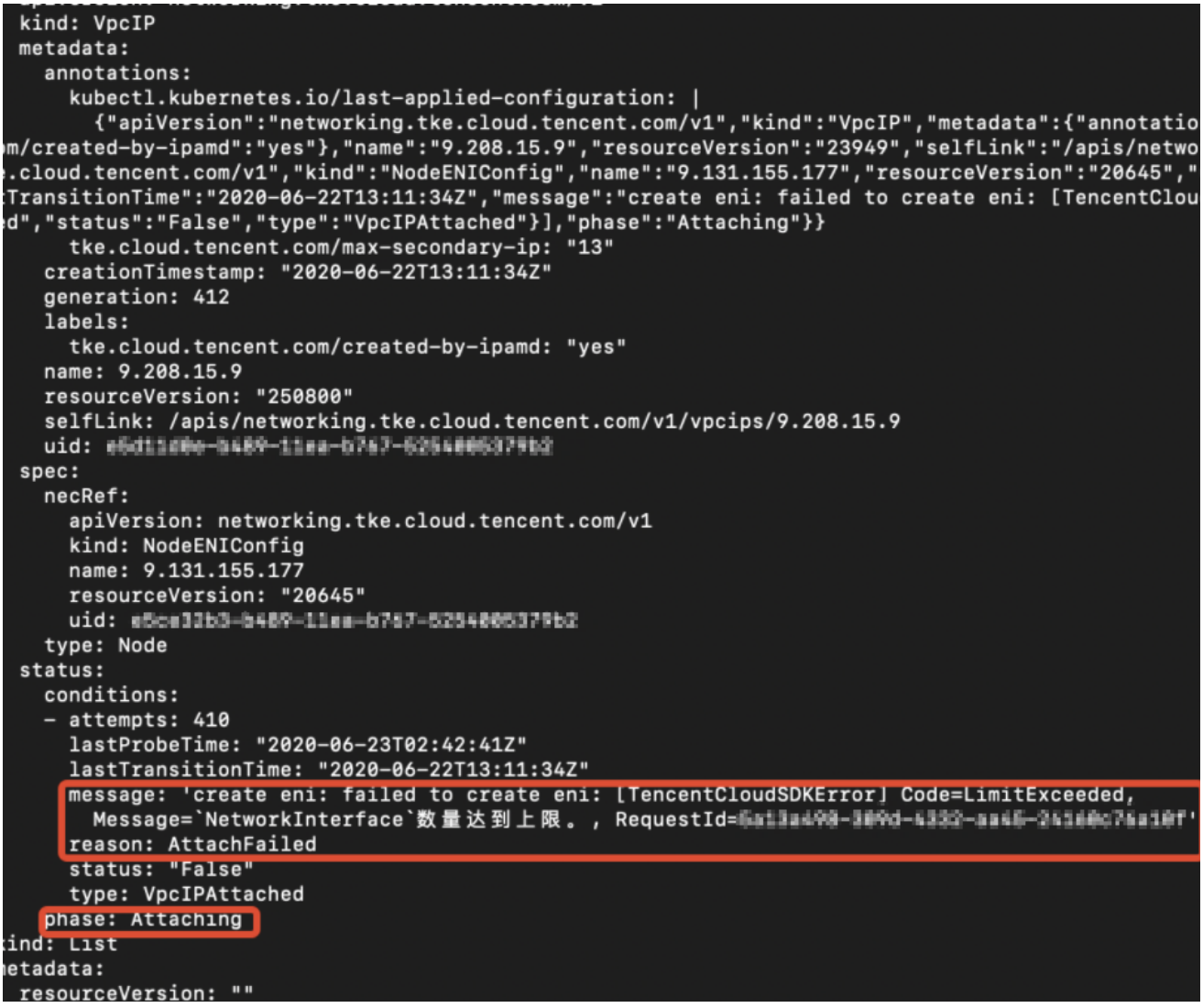
当节点关联的 nec status 为空时返回结果如下图所示：



执行以下代码查看 nec 关联的 VIP：

```
kubectl get vip -oyaml
```

若命令返回成功则报错 VIP 状态为 Attaching，报错信息如下图所示：



解决方案

目前腾讯云弹性网卡限制一个 VPC 下面最多绑定50个弹性网卡。您可 在线咨询 申请提高配额，配额按地域生效。

非固定 IP 模式使用说明

最近更新时间：2021-12-17 14:47:46

使用场景

适用于不依赖容器固定 IP 的场景。例如，可部署多副本的无状态服务，无状态离线业务等。

能力和限制

- 支持节点维护可用的网卡/ IP 池，从而支持 Pod 大规模快速重建。
- 支持预绑定策略，从而一定范围内支持 Pod 快速扩容。
- 支持弹性伸缩网卡/ IP，从而可避免 IP 浪费，提高 IP 利用率。
- 预绑定值不可为0，即暂不能支持完全按需分配，节点数过多可能会造成 IP 浪费。

IP 地址管理原理

TKE 组件在每个节点维护一个可弹性伸缩的独占网卡/IP 池。已绑定的独占网卡/IP 数量将被维持在 **Pod 数量 + 最小预绑定数量**及 **Pod 数量 + 最大预绑定数量**之间。

- 当**已绑定数量 < Pod 数量 + 最小预绑定数量**时，会绑定独占网卡/IP 使得**已绑定数量 = Pod 数量 + 最小预绑定数量**。
- 当**已绑定数量 > Pod 数量 + 最大预绑定数量**时，会定时释放独占网卡/IP（约2分钟一次），直到**已绑定数量 = Pod 数量 + 最大预绑定数量**。
- 当**最大可绑定数量 < 当前已绑定数量**时，会直接释放多余的空闲独占网卡/IP，使得**已绑定数量 = 最大可绑定数量**。

使用方法

您可以通过以下方式启用非固定 IP：

- 创建集群选择非固定 IP 模式的 VPC-CNI：集群创建时不勾选**固定Pod IP** 选项。

支持快释放

默认情况，非固定 IP 模式管理的网卡/IP 池采用慢释放策略，默认是2分钟只释放1个多余的网卡/IP，若用户需要更高效的利用 IP，则需要开启快释放，快释放模式下，每2分钟会检查一次网卡/IP 池，释放多余的网卡/IP，直到空闲网卡/IP 数等于最大预绑定值。

开启方法

- 修改现存的 tke-eni-agent daemonset：kubectl edit ds tke-eni-agent -n kube-system。
- 在 spec.template.spec.containers[0].args 中加入以下启动参数开启快释放。修改后，agent 会滚动更新生效特性。

```
- --enable-quick-release
```

指定某节点预绑定数量

可通过修改节点对应的 CRD NEC 的注解来指定该节点 eni-ip 预绑定的数量，相关的注解为：

```
# 共享网卡模式指定最小预绑定值
"tke.cloud.tencent.com/route-eni-ip-min-warm-target"
# 共享网卡模式指定最大预绑定值
"tke.cloud.tencent.com/route-eni-ip-max-warm-target"
# 独占网卡模式指定最小预绑定值
"tke.cloud.tencent.com/direct-eni-min-warm-target"
# 独占网卡模式指定最大预绑定值
"tke.cloud.tencent.com/direct-eni-max-warm-target"
```

修改方法如下：

```
# 示例，修改节点 <nodeName> 的最小预绑定 ip 值为1
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/route-eni-ip-min-warm-target"="1" --overwrite
# 示例，修改节点 <nodeName> 的最大预绑定 ip 值为3
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/route-eni-ip-max-warm-target"="3" --overwrite
```

- 修改后即触发动态预绑定的检查，如果预绑定数量不满足期望，会绑定足够网卡/IP。反之则会解绑网卡/IP。
- 修改时这两个注解必须同时存在，且满足：0 <= 最小预绑定 <= 最大预绑定，否则修改失败。

指定某节点最大绑定数量

可通过修改节点对应的 CRD nec 的注解来指定该节点网卡/IP 最大绑定的数量，可指定最大的网卡数和单网卡绑定的 IP 数，相关的注解为：

```
# 共享网卡模式指定最大网卡数
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/route-eni-max-attach"="1" --overwrite
# 共享网卡模式指定单网卡绑定的 IP 数
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/max-ip-per-route-eni"="9" --overwrite
# 独占网卡模式指定最大独占网卡数
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/direct-eni-max-attach"="5" --overwrite
```

修改时需保证修改值大于等于节点当前正在使用的网卡/IP 数量，否则修改失败。
修改后即触发动态预绑定的检查，如果已绑定数量 > 最大可绑定值，则会解绑网卡/IP，使已绑定数量 = 最大可绑定值。

指定默认预绑定数量

- 修改现存的 tke-eni-ipamd deployment：kubectl edit deploy tke-eni-ipamd -n kube-system。
- 在 spec.template.spec.containers[0].args 中加入以下启动参数修改默认预绑定值。修改后，ipamd 会自动重启并生效。默认值只影响新增的节点：

```
# 共享网卡模式最小预绑定默认值，默认值为 5
- --ip-min-warm-target=3
# 共享网卡模式最大预绑定默认值，默认值为 5
- --ip-max-warm-target=3
# 独占网卡模式最小预绑定默认值，默认值为 1
- --eni-min-warm-target=3
# 独占网卡模式最大预绑定默认值，默认值为 1
- --eni-max-warm-target=3
```

VPC-CNI 模式与其他云资源、IDC 互通

最近更新时间：2022-06-09 14:33:24

VPC-CNI 模式和容器网络属于 VPC 可管理的网段，因此可以直接通过 VPC 的产品功能配置实现与其他云资源、IDC 资源的互通。

腾讯云为您提供丰富的解决方案，可以满足 VPC 内的云服务器、数据库等实例连接公网（Internet）、连接其他 VPC 内实例、或与本地数据中心（IDC）互联的需求。私有网络 VPC 及其多种连接方式详情可参见 VPC 连接方案概述 。

VPC-CNI 模式安全组使用说明

最近更新时间：2022-06-09 14:42:12

您可以通过下述方式为 VPC-CNI 模式创建的弹性网卡绑定指定的安全组。

前提条件

- IPAMD 组件版本在 v3.2.0+（可通过镜像 tag 查看）。
- IPAMD 组件启动了安全组能力，启动参数：--enable-security-groups（默认未启用）。
- 目前仅支持多 Pod 共享网卡模式。

IPAMD 组件角色添加安全组接口访问权限

1. 登录 访问管理控制台，选择左侧的**策略**。
2. 在“策略”详情页中，单击**新建自定义策略**。
3. 在弹出的选择创建方式窗口中，单击**按策语法创建**，进入选择策略模板页面。
4. 选择“空白模板”，并单击**下一步**，进入编辑策略页面。
5. 在编辑策略页面，确认策略名称、输入以下策略语法后，单击**完成**即可创建自定义策略。

> 说明：
> 策略可命名为 SecurityGroupsAccessForIPAMD 。

```
{
"version": "2.0",
"statement": [
{
"action": [
"cvm:AssociateNetworkInterfaceSecurityGroups",
"cvm:DisassociateNetworkInterfaceSecurityGroups"
],
"resource": "*",
"effect": "allow"
}
]
}
```

6. 在**访问管理控制台** > **角色**中搜索 IPAMD 组件的相关角色 IPAMDofTKE_QCSRole，单击角色名称进入角色详情页面。
7. 在权限设置中，单击**关联策略**。
8. 在弹出的关联策略窗口中，勾选已创建的自定义策略 SecurityGroupsAccessForIPAMD。单击**确定**，完成为 IPAMD 组件角色添加安全组接口访问权限操作。

IPAMD 组件开启安全组特性

- 修改现存的 tke-eni-ipamd deployment：kubectl edit deploy tke-eni-ipamd -n kube-system 。
- 执行以下命令，在 spec.template.spec.containers[0].args 中加入启动参数。
  修改后，ipamd 会自动重启并生效。
  生效后，存量节点上的辅助弹性网卡没有关联安全组的会按以下策略绑定安全组，如果绑定了也会与设置的安全组强同步，除非之前已开启特性，节点安全组已设置。增量节点的弹性网卡则都会绑定以下安全组。

```
- --enable-security-groups
# 如果希望默认继承自主网卡/实例的安全组，则不添加 security-groups 参数
- --security-groups=sg-xxxxxxxx,sg-xxxxxxxx
```

如果想让已设置安全组的存量节点也生效，需要手动禁用安全组，再开启来达到同步。以下为存量节点的同步方法：

i. 给节点加上注解清空并禁用节点的弹性网卡绑定安全组,添加后，节点的存量弹性网卡会解绑所有安全组：

```
kubectl annotate node <nodeName> --overwrite tke.cloud.tencent.com/disable-node-eni-security-groups="yes"
```

ii. 重新设置为 no 后，则可以重新绑定以上策略配置的安全组：

```
kubectl annotate node <nodeName> --overwrite tke.cloud.tencent.com/disable-node-eni-security-groups="no"
```

功能逻辑

- 若未设置启动参数 --security-groups，或者其值为空，则各节点安全组继承自节点实例绑定的安全组。
- 特性开启以后，如果设置了 --security-groups，则各节点安全组设置为该安全组集合。
- 特性开启以后，如果变更 --security-groups 参数，增量节点安全组设置会与全局参数同步，存量节点安全组设置不会改变，若需同步存量节点安全组设置，则需禁用节点安全组再开启，来达到同步。操作方法见 IPAMD 组件开启安全组特性。
- 安全组设置的优先级与节点安全组设置的顺序一致，若继承自主网卡，则与主网卡保持一致。
- 执行以下命令可查看节点安全组。其中 spec.securityGroups 域包含了节点安全组信息。

```
kubectl get nec <nodeName> -oyaml
```

执行以下命令可修改节点安全组，修改后即刻生效。

```
kubectl edit nec <nodeName>
```

- 特性开启以后，节点同步时，存量网卡如果没绑定安全组，则会绑定节点安全组。存量网卡的安全组会与节点安全组强同步，保证与设置的节点安全组保持一致。增量网卡都会绑定节点安全组。

Pod 直接绑定弹性公网 IP 使用说明

最近更新时间：2022-06-09 14:50:02

您可以通过下述方式为 VPC-CNI 模式的 Pod 直接绑定弹性公网 IP（EIP）。

前提条件和限制

- IPAMD 使用的角色策略被授权了 EIP 相关的接口权限。
- 目前仅支持自动新建 EIP，不支持指定使用已有 EIP。
- 目前 VPC-CNI 独占网卡非固定 IP 模式暂不支持 EIP 功能（v3.3.9及之后版本可支持）。
- 当前集群删除时暂不支持回收该集群自动创建的 EIP。

IPAMD 组件角色添加 EIP 接口访问权限

1. 登录 访问管理控制台，选择左侧的**角色**。
2. 在**访问管理控制台** > **角色**中搜索 IPAMD 组件的相关角色 IPAMDofTKE_QCSRole，单击角色名称进入角色详情页面。
3. 在权限设置中，单击**关联策略**。
4. 在弹出的关联策略窗口中，在搜索框中搜索 QcloudAccessForIPAMDRoleInQcloudAllocateEIP，然后勾选已创建的预设策略 QcloudAccessForIPAMDRoleInQcloudAllocateEIP。**单击确定**，完成为 IPAMD 组件角色添加 EIP 接口访问权限操作。该策略包含了 IPAMD 组件操作弹性公网 IP 所需的所有权限。

自动新建 EIP

如需自动关联 EIP，可参考以下 Yaml 示例：

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
labels:
k8s-app: busybox
name: busybox
namespace: default
spec:
replicas: 1
selector:
matchLabels:
k8s-app: busybox
qcloud-app: busybox
serviceName: ""
template:
metadata:
annotations:
tke.cloud.tencent.com/networks: "tke-route-eni"
tke.cloud.tencent.com/vpc-ip-claim-delete-policy: Never
tke.cloud.tencent.com/eip-attributes: '{"Bandwidth":"100","ISP":"BGP"}'
tke.cloud.tencent.com/eip-claim-delete-policy: "Never"
creationTimestamp: null
labels:
k8s-app: busybox
qcloud-app: busybox
spec:
containers:
- args:
```

```
  - "10000000000"
  command:
  - sleep
  image: busybox
  imagePullPolicy: Always
  name: busybox
  resources:
  limits:
  tke.cloud.tencent.com/eni-ip: "1"
  tke.cloud.tencent.com/eip: "1"
  requests:
  tke.cloud.tencent.com/eni-ip: "1"
  tke.cloud.tencent.com/eip: "1"
```

- **spec.template.annotations：tke.cloud.tencent.com/eip-attributes: '{"Bandwidth":"100","ISP":"BGP"}'** 表明该 Workload 的 Pod 需要自动关联 EIP，且 EIP 的带宽是 100 Mbps，线路类型是 BGP。
- **spec.template.annotations：tke.cloud.tencent.com/eip-claim-delete-policy: "Never"** 表明 Workload 的 Pod 的 EIP 也需要固定，Pod 销毁后不能变更。若不需要固定，则不添加该注解。
- **spec.template.spec.containers.0.resources**：关联 EIP 的 Pod，您需要添加 requests 和 limits 限制，即 tke.cloud.tencent.com/eip，从而让调度器保证 Pod 调度到的节点仍有 EIP 资源可使用。

关键配置说明

- 各节点可绑定的 EIP 资源受到相关配额限制和云服务器的绑定数量限制，详情可参考 EIP使用限制。
  各节点可绑定的最大 EIP 数量为**云服务器绑定数量 - 1**。
- **tke.cloud.tencent.com/eip-attributes: '{"Bandwidth":"100","ISP":"BGP"}'**：当前只支持配置带宽和线路类型两个参数。ISP参数可配置为 BGP、CMCC、CTCC、CUCC，分别对应普通线路 BGP IP、静态单线 IP（网络运营商中国移动、中国电信、中国联通）。若不填写，则默认值为 100 Mbps 和 BGP。
- 当前自动申请的 EIP 绑定后不收取 IP 资源费用，访问公网网络默认计费方式为流量按小时后付费，详情见 EIP 计费概述 。

EIP 的保留和回收

Pod 启用自动关联 EIP 特性后，网络组件会为该 Pod 在同 namespace 下创建同名的 CRD 对象 EIPClaim 。该对象描述 Pod 对 EIP 的需求。

对于非固定 EIP 的 Pod，其 Pod 销毁后 EIPClaim 也会被销毁，Pod 关联的 EIP 随之销毁回收。而对于固定 EIP 的 Pod，其 Pod 销毁后 EIPClaim 仍然保留，EIP 也因此保留。**同名**的 Pod 启动后会使用同名的 EIPClaim 关联的 EIP，从而实现 EIP 保留。

下面介绍三种回收 EIP 的方法：过期回收、手动回收及级联回收。

过期回收（默认支持）

在 创建集群 页面，容器网络插件选择 **VPC-CNI** 模式并且勾选**开启支持**固定 Pod IP 支持，如下图所示：

在高级设置中设置 IP 回收策略，可以设置 Pod 销毁后多少秒回收保留的固定 IP。如下图所示：

对于存量集群，也可支持变更：

- **修改现存的 tke-eni-ipamd deployment：**kubectl edit deploy tke-eni-ipamd -n kube-system。
- **执行以下命令，在** spec.template.spec.containers[0].args **中加入/修改启动参数。**

```
- --claim-expired-duration=1h # 可填写不小于 5m 的任意值
```

手动回收

对于急需回收的 EIP，找到对应的 Pod 的名称空间和名称，执行以下命令通过手动回收：

> ⚠ **注意：**
> **需保证回收的 EIP 对应的 Pod 已经销毁，否则会再次触发关联绑定 EIP。**

```
kubectl delete eipc <podname> -n <namespace>
```

级联回收

目前的固定 EIP 与 Pod 强绑定，而与具体的 Workload 无关（例如 deployment、statefulset 等）。Pod 销毁后，固定 EIP 不确定何时回收。TKE 现已实现删除 Pod 所属的 Workload 后即刻删除固定 EIP。要求 IPAMD 组件版本在 v3.3.9+（可通过镜像 tag 查看）。

以下步骤介绍如何开启级联回收：

1. **修改现存的 tke-eni-ipamd deployment：kubectl edit deploy tke-eni-ipamd -n kube-system。**
2. **执行以下命令，在** spec.template.spec.containers[0].args **中加入启动参数：**

```
- --enable-ownerref
```

修改后，ipamd 会自动重启并生效。生效后，增量 Workload 可实现级联删除固定 EIP，存量 Workload 暂不能支持。

VPC-CNI 组件
VPC-CNI 组件介绍

最近更新时间：2022-06-09 14:51:16

VPC-CNI 组件包含3个 kubernetes 集群组件，分别是 tke-eni-agent 、tke-eni-ipamd 和 tke-eni-ip-scheduler 。

tke-eni-agent

以 daemonset 形式部署在集群中的每个节点上，职责：

- 拷贝 tke-route-eni 和 tke-eni-ipamc 等 CNI 插件到节点 CNI 执行文件目录（默认为 /opt/cni/bin ）。
- 在 CNI 配置目录（默认为/etc/cni/net.d/ ）生成 CNI 配置文件。
- 设置节点策略路由和弹性网卡。
- Pod IP 分配/释放的 GRPC Server。
- 定期进行 IP 垃圾回收，回收 Pod 已不在节点上的 IP。
- 通过 kubernetes 的 device-plugin 机制 设置网卡和 IP 的扩展资源。

tke-eni-ipamd

以 deployment 形式部署在集群中的特定节点或 master 上，职责：

- 创建管理 CRD 资源（nec, vipc, vip, veni ）。
- 非固定 IP 模式下，依据节点需求和状态创建/绑定/解绑/删除弹性网卡，分配/释放弹性网卡 IP。
- 固定 IP 模式下，依据 Pod 需求和状态创建/绑定/解绑/删除弹性网卡，分配/释放弹性网卡 IP。
- 节点弹性网卡安全组管理。
- 依据 Pod 需求创建/绑定/解绑/删除弹性公网 IP。

tke-eni-ip-scheduler

以 deployment 形式部署在集群中的特定节点或 master 上，仅固定 IP 模式会部署，为调度扩展插件，职责：

- 多子网情况下，需要让已固定 IP 的 Pod 调度到指定子网的节点。
- 固定 IP 模式下，判断 Pod 调度的节点对应子网 IP 是否充足。

VPC-CNI 组件变更记录

最近更新时间：2022-06-09 14:52:07

VPC-CNI 组件包含3个 kubernetes 集群组件，分别是 tke-eni-agent 、tke-eni-ipamd 和 tke-eni-ip-scheduler 。一般情况下，三个组件版本相同，但 tke-eni-ip-scheduler 组件变更较少，版本可能会稍微落后。

查看当前组件的版本信息

组件的版本即为镜像的 Tag，通过 kubernetes API 可查看：

```
# 查看 tke-eni-agent 的版本
kubectl -nkube-system get ds tke-eni-agent -o jsonpath={.spec.template.spec.containers[0].image}
# 查看 tke-eni-ipamd 的版本
kubectl -nkube-system get deploy tke-eni-ipamd -o jsonpath={.spec.template.spec.containers[0].image}
# 查看 tke-eni-ip-scheduler 的版本
kubectl -nkube-system get deploy tke-eni-ip-scheduler -o jsonpath={.spec.template.spec.containers[0].image}
```

变更记录

版本号	发布时间	变更内容	变更影响
v3.3.9	2021-11-09	• 修复网络原因导致的 EIP 重复创建问题。 • 支持独立网卡非固定 IP 模式的 Pod 绑定 EIP。 • 优化 eni-agent 的扩展资源机制，使扩展资源的管理更加稳定健壮。 • 修复节点设置配额和实际配额不一致导致的问题。 • 优化 eni-agent IP 垃圾回收机制，针对正在创建的 Pod，如果有脏容器，则将回收 IP 分给该 Pod 的新容器。 • 优化非固定 IP 模式下已使用 IP 和网卡的资源计数算法，修复 Error、Evicted、Completed 等状态的 Pod 导致的资源计数不准的问题	对业务无影响
v3.3.8	2021-08-17	• 支持 --master 参数直接配置后端 kube-apiserver 地址，解除 kube-proxy 依赖。 • eni-agent 支持参数 --kube-client-qps 和 --kube-client-burst 配置 kube client 的 QPS 和 Burst，默认值提升至 10 和 20。 • eni-agent 若发现更新后的扩展资源比原来更少，提前将最新的扩展资源信息更新到节点状态中，避免因为 kubelet 异步更新带来的问题。	对业务无影响
v3.3.7	2021-08-13	• eni-ipamd 支持 --enable-node-condition 和 --enable-node-taint 参数，打开后，若节点缺少 eni-ip 或 direct-eni 等本该需要的扩展资源，节点的 condition 或 taints 将被设置。 • EIP 支持 json 格式解析新的 API 参数。 • 修复 containerd 运行时下，eni-agent 的垃圾回收小概率会把刚分配好的 IP 错误回收的问题。 • 修复 EIP 接口可能导致的 ipamd panic 问题。 • 修复非固定 IP 模式升级时，可能误设置了 disable-node-eni annotation 导致网卡被解绑的问题	对业务无影响
v3.3.6	2021-07-26	• 修复 eni-agent 垃圾回收机制可能导致刚分配好的 IP 和路由被错误回收的问题。 • 修复 eni-ipamd 在打开级联回收 --enable-ownerref 之后，在删除deployment等上层资源时，IP 可能先于 Pod 释放的问题。	对业务无影响
v3.3.5	2021-07-20	• 修复非固定 IP 模式下，共享网卡/独占网卡的 Pod 由于 IP 或 ENI 资源被误删除导致本地存储数据不能删除的问题。 • 修复非固定 IP 模式下，共享网卡/独占网卡的 CNI 信息没有存储校验 Pod 网卡信息的问题。	对业务无影响
v3.3.4	2021-07-07	• 修复 CVM 已关机下不断重试解绑网卡的问题。 • 修复异步 志同步写导致的 panic 问题。 • 优化非固定 IP 模式的网卡同步逻辑，保证内部数据一致性，避免解绑正在使用的网卡。 • 修复从 v3.2 升级的非固定 IP 集群由于子网 IP 不足导致存量节点不能分配 IP 的问题。 • 修复存量网卡主 IP 被 Pod 使用的网卡可能会被错误释放的问题。	对业务无影响
v3.3.3	2021-06-07	• 支持混合云 ipam，与 cilium overlay/underlay 模式协同工作。	对业务无影响

v3.3.2	2021-06-01	• ip-scheduler 支持抢占，但只支持默认资源不足导致的抢占，暂不支持 ip 资源不足导致的抢占。 • 重构共享网卡的安全组功能逻辑，支持与节点设置安全组强同步，保证安全组绑定顺序与优先级与用户设置一致。 • 支持 cilium cni-chain 模式。 • eni-agent 支持--port-mapping 参数实现 Pod hostPort 字段支持。 • 支持 Pod 打上注解 tke.cloud.tencent.com/claim-expired-duration 实现特定的固定 IP 回收时间，Pod 注解只影响增量。	对业务无影响
v3.3.1	2021-05-11	• 支持共享网卡非固定 IP 模式使用多网卡。 • 支持腾讯云 API 调用接口 QPS 限制，默认单集群限制为 50 QPS（按 CVM、VPC、TKE 类型限制）。 • 支持非固定 IP 模式升配后的 IP 配额变化感知。 • 支持 node 注解 tke.cloud.tencent.com/desired-route-eni-pod-num，写入需要的 route-eni ip 数量，写入后组件自动调整节点配额。 • 修复由于 VPC 任务不存在导致的 VPC 任务轮询超时问题。 • 修复由于网卡创建任务失败导致的 eni-ipamd panic 问题。 • 优化路由对账逻辑，只清除属于 eni-agent 管理的 IP 路由。 • 修复独立网卡非固定 IP 模式在释放网卡的时候可能由于网卡已经释放导致的异常 panic 问题。	对业务无影响
v3.3.0	2021-04-13	• 支持自定义 GR 模式，该模式支持节点集群多 CIDR。	对业务无影响
v3.2.6	2021-03-31	• 减少独占网卡模式下绑定网卡的重试时间，提高绑定效率。 • 通过并发控制，减少并发绑定和解绑网卡的失败，提高绑定和解绑的效率。 • 非固定 IP 模式优化网卡子网分配逻辑，修复并发加节点时，部分节点在 IP 充足的情况下拿不到 IP 的问题。 • eni-agent 垃圾回收机制支持自感知底层运行时，并支持 containerd。	对业务无影响
v3.2.5	2021-02-22	• eni-ipamd 和 ip-scheduler 部署时增加 dnsConfig，避免用户自建 DNS 带来的问题。 • 共享网卡固定 IP 模式下，每个节点绑定的网卡的 subnetID 信息会同步到节点的 label 上，key 为 tke.cloud.tencent.com/route-eni-subnet-ids。 • eni-agent 会尝试获取 IP 申请分配失败的原因，并返回给 CNI 插件，最终体现在 Pod event 中。 • 支持裸 Pod 指定 IP，通过注解 tke.cloud.tencent.com/nominated-vpc-ip 可指定。 • eni-agent 支持定时测试和 APIServer 的连接情况，若超时则自动重启。 • 修复由于内部数据不一致导致的 ip 浪费的问题。	对业务无影响

VPC-CNI 模式 Pod 数量限制

最近更新时间：2022-06-09 14:50:47

本文说明 VPC-CNI 各网络模式 Pod 数量默认限制，如不满足需求，可以 提交工单 调整限制。

共享网卡 Pod 数量限制

共享网卡的 Pod 数量受限于节点可绑定的网卡数量和单网卡可绑定的 IP 数量，默认情况下，多网卡的**单节点 PodIP 数量上限 = 最大可绑定辅助网卡数 * 单网卡可绑定辅助 IP 数**，而单网卡的**单节点 PodIP 数量上限 = 单网卡可绑定辅助 IP 数**。默认情况详见下表：

CPU 核数	1	2-6	8-10	>=12
最大可绑定辅助弹性网卡	1	3	5	7
单网卡可绑定辅助 IP 数	5	9	19	29
非固定 IP 模式（多网卡）单节点 Pod IP 上限	5	27	95	203
固定 IP 模式（单网卡）单节点 Pod IP 上限	5	9	19	29

各机型可绑定的网卡数量和单网卡可绑定的 IP 数量略有差异， 详情见 弹性网卡使用限制。

独占网卡模式 Pod 数量限制

独占网卡的 Pod 数量只受限于节点可绑定的网卡数量，同时只支持 S5、SA2、IT5、SA3 等部分机型，默认情况详见下表：

CPU核数 / 机型	1	2	4	>=8	>=128
S5	4	9	19	39	23
SA2	4	9	19	39	23
IT5	4	9	19	39	23
SA3	4	9	15	15	15

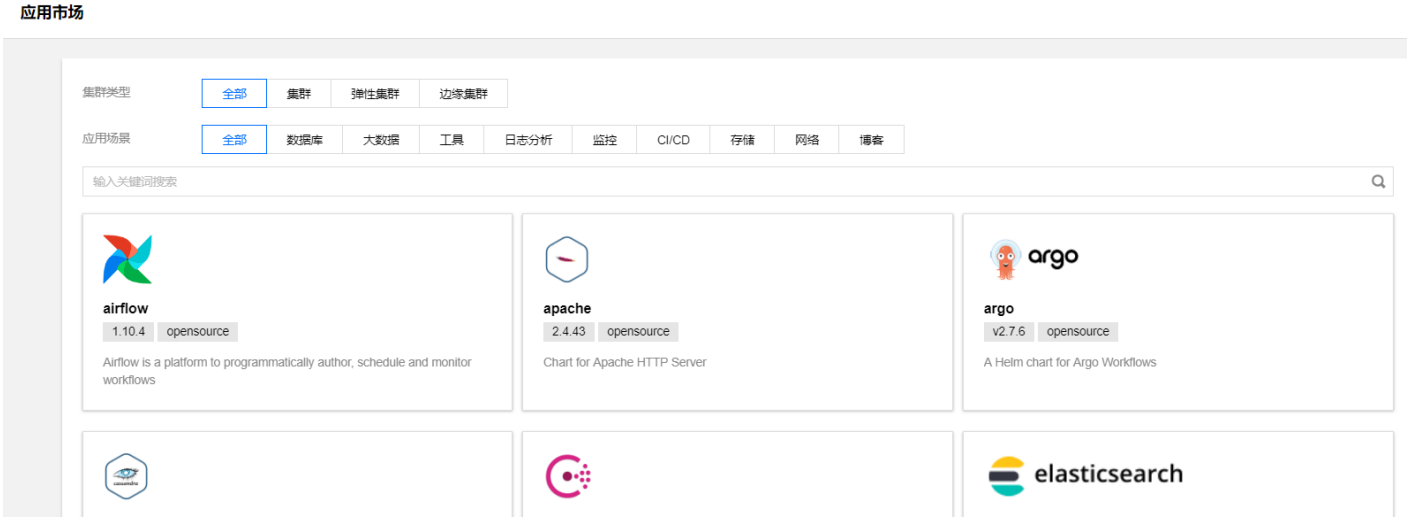
应用市场

最近更新时间：2022-01-19 14:25:01

腾讯云容器服务（Tencent Kubernetes Engine，TKE）应用市场按照集群类型、应用场景等分类方式，为您提供多种产品和服务。例如 helm chart、容器镜像、软件服务等。本文介绍如何通过容器服务控制台中的应用市场，快速完成应用创建。

查看应用

1. 登录 容器服务控制台。
2. 在左侧导航栏中，单击**应用市场**进入“应用市场”管理页面。如下图所示：



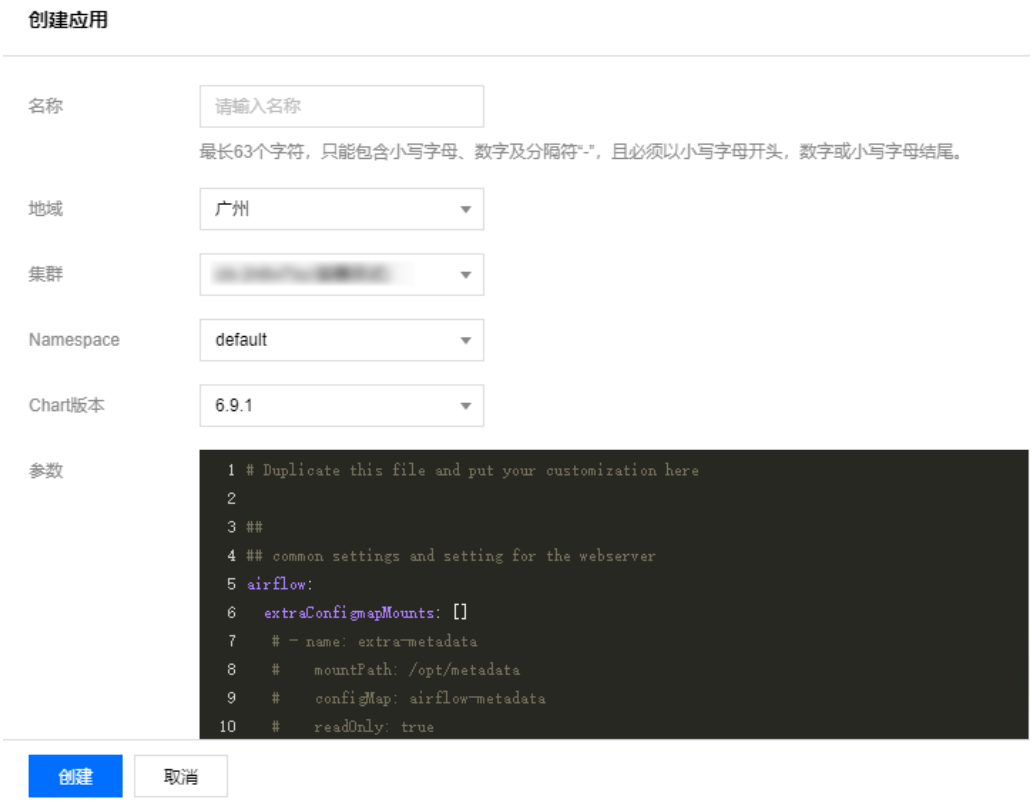
在“应用市场”管理页面中，可进行如下操作：

- 筛选应用：可按照集群类型、应用场景或输入关键词进行应用筛选。
  - 集群类型：包含集群、弹性集群、边缘集群。
  - 应用场景：包含数据库、大数据、工具、日志分析、监控、CI/CD、存储、网络、博客。
- 查看应用：单击需要查看的应用包，即可前往该应用详情页。

创建应用

1. 在“应用市场”管理页面中按需选择应用包，并进入该应用详情页。
2. 在“应用详情页”中，单击“基本信息”模块中的**创建应用**。

3. 在弹出的“创建应用”窗口中，按需配置并创建应用。如下图所示：



4.单击**创建**即可创建应用。

集群运维
日志管理
通过控制台使用日志采集

最近更新时间：2022-04-08 09:58:41

操作场景

日志采集功能是容器服务 TKE 为用户提供的集群内日志采集工具，可以将集群内服务或集群节点特定路径文件的日志发送至 腾讯云日志服务 CLS、消息队列 CKafka。日志采集功能适用于需要对 Kubernetes 集群内服务日志进行存储和分析的用户。

日志采集功能需要为每个集群手动开启并配置采集规则。日志采集功能开启后，日志采集 Agent 会在集群内以 DaemonSet 的形式运行，并根据用户通过日志采集规则配置的采集源、CLS 日志主题和日志解析方式，从采集源进行日志采集，将日志内容发送到日志消费端。您可根据以下操作开启日志采集功能：

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

前提条件

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

概念

- **日志采集 Agent**：TKE 用于采集日志信息的 Agent，采用 Loglistener，在集群内以 DaemonSet 的方式运行。
- **日志规则**：用户可以使用日志规则指定日志的采集源、日志主题、日志解析方式和配置过滤器。
  - 日志采集 Agent 会监测日志采集规则的变化，变化的规则会在最多10s内生效。
  - 多条日志采集规则不会创建多个 DaemonSet，但过多的日志采集规则会使得日志采集 Agent 占用的资源增加。
- **日志源**：包含指定容器标准输出、容器内文件以及节点文件。
  - 在采集容器标准输出日志时，用户可选择所有容器、或指定工作负载和指定 Pod Labels 内的容器服务日志作为日志的采集源。
  - 在采集容器文件路径日志时，用户可指定工作负载或 Pod Labels 内容器的文件路径日志作为采集源。
  - 在采集节点文件路径日志时，用户可设定日志的采集源为节点文件路径日志。
- **消费端**：用户选择日志服务 CLS 的日志集和日志主题作为消费端。
- **提取模式**：日志采集 Agent 支持将采集到的日志以单行文本、JSON、分隔符、多行文本和完全正则的形式发送至用户指定的日志主题。
- **过滤器**：开启过滤器后可以根据用户指定的规则采集部分日志，key 支持完全匹配，过滤规则支持正则匹配，如仅采集 ErrorCode = 404 的日志。

操作步骤

开启日志采集

1. 登录 容器服务控制台，选择左侧导航栏中的**运维功能管理** > **功能管理**。

2. 在“功能管理”页面上方选择地域，单击需要开启日志采集的集群右侧的**设置**。如下图所示：

3. 在“设置功能”页面，单击日志采集**编辑**，开启日志采集后确认。如下图所示：

配置日志规则

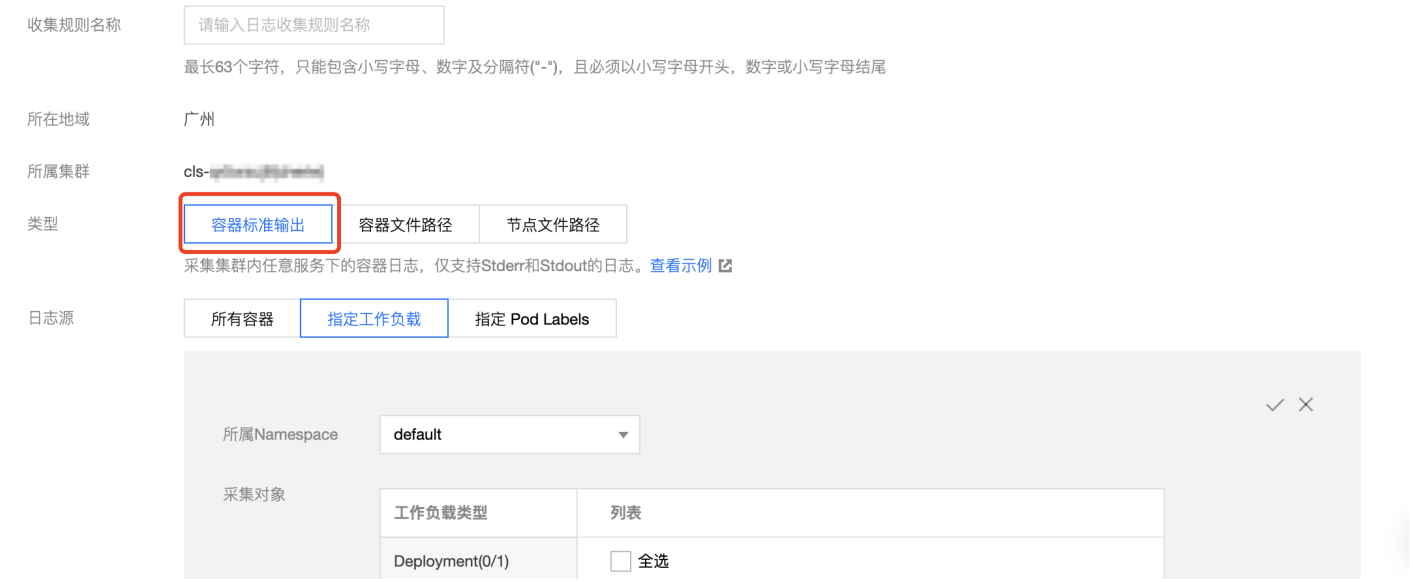
1. 登录 容器服务控制台，选择左侧导航栏中的**运维功能管理 > 日志规则**。

2. 在“日志采集”页面上方选择地域和需要配置日志采集规则的集群，单击**新建**。如下图所示：

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

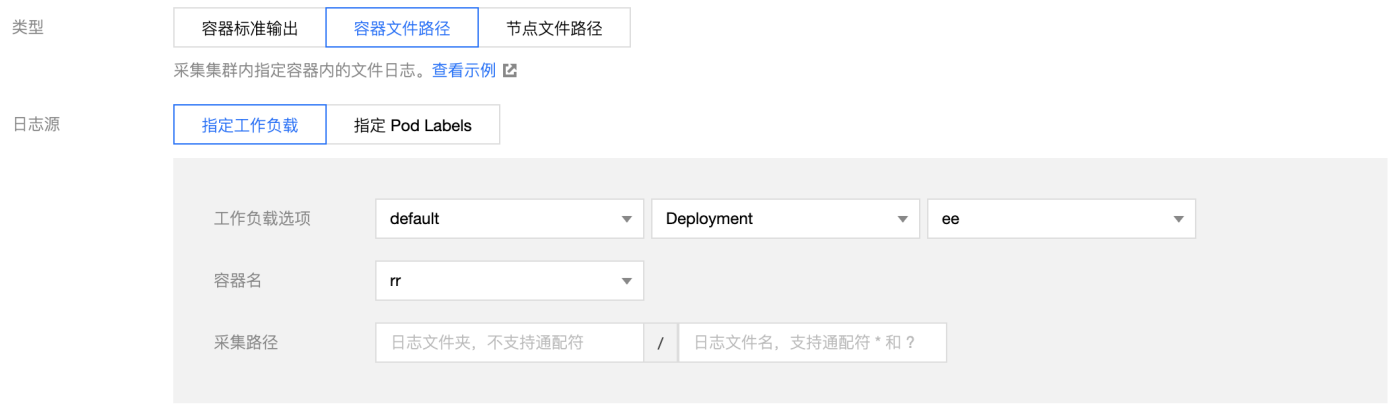
采集容器标准输出日志

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



采集容器内文件日志

选择**容器文件路径**采集类型，并配置日志源。如下图所示：



采集文件路径支持文件路径和通配规则，例如当容器文件路径为 /opt/logs/*.log，可以指定采集路径为 /opt/logs，文件名为 *.log 。

> **注意**
> 如果选择采集类型为“容器文件路径”时，对应的“容器文件路径”**不能为软链接**，否则会导致软链接的实际路径在采集器的容器内不存在，采集日志失败。

采集节点文件日志

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

> **注意：**
> 一个节点日志文件只能被一个日志主题采集。

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

说明

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

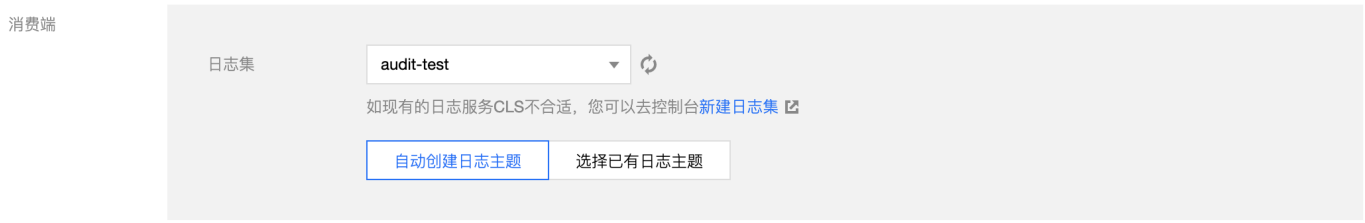
容器或 kubernetes 相关的元数据请参考下方表格：

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

4. **配置日志服务消费端。**

配置日志消费端为 CLS

选择日志集和相应的日志主题，可以选择新建和已有日志主题。如下图所示：



注意：

- **日志服务 CLS 目前只能支持同地域的容器集群进行日志采集上报。**
- **若日志集下已存在500个日志主题，则不能新建日志主题。**

配置日志消费端为 Kafka

支持用户选择写入 CKafka 或用户自建 Kafka ，当选择 CKafka 时，需要填写实例 ID 和实例 Topic ；当选择自建 Kafka 时，需按要求填写 Broker 地址和 Topic。

> **注意：**
> - 如果 Kafka 实例与节点不在同一个 VPC 下，会提示创建 Kafka 实例接入点后再进行日志投递。
> - 在集群的 daemonSet 资源中，选择 kube-system 命名空间，找到 tke-log-agent pod 下的 kafkalistener 容器，可以查询 kafka 采集器的日志。

支持在高级设置内通过指定 Key 值将日志投递到指定分区，该功能默认不开启，日志随机投放，当开启后，带有同样 Key 值的日志，将投递到相同的分区。支持输入 TimestampKey（默认@timestamp）和指定时间戳格式。如下图所示：

5. 单击**下一步**，选择日志提取模式。如下图所示：

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

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

解析模式	说明	相关文档
单行 – 完全正则	指将一条完整日志按正则方式提取多个 key-value 的日志解析模式，您需先输入日志样例，其次输入自定义正则表达式，系统将根据正则表达式里的捕获组提取对应的 key-value。支持自动生成正则表达式。	单行 – 完全正则格式
多行 – 完全正则	适用于日志文本中一条完整的日志数据跨占多行（例如 Java 程序日志），可按正则表达式提取为多个 key-value 键值的日志解析模式，您需先输入日志样例，其次输入自定义正则表达式，系统将根据正则表达式里的捕获组提取对应的 key-value。支持自动生成正则表达式。	多行–完全正则格式
JSON	JSON 格式日志会自动提取首层的 key 作为对应字段名，首层的 value 作为对应的字段值，以该方式将整条日志进行结构化处理，每条完整的日志以换行符 \n 为结束标识符。	JSON 格式
分隔符	指一条日志数据可以根据指定的分隔符将整条日志进行结构化处理，每条完整的日志以换行符 \n 为结束标识符。日志服务在进行分隔符格式日志处理时，您需要为每个分开的字段定义唯一的 key，无效字段即无需采集的字段可填空，不支持所有字段均为空。	分隔符格式

6. 根据需求开启过滤器并配置规则，并单击**完成**，完成创建。如下图所示：

更新日志规则

1. 登录 容器服务控制台，选择左侧导航栏中的**运维功能管理 > 日志规则**。
2. 在“日志采集”页面上方选择地域和需要更新日志采集规则的集群，单击右侧的**编辑收集规则**。如下图所示：

3. 根据需求更新相应配置，单击**完成**，完成更新。

> ⚠ 注意：
> **日志集和日志主题不可更新。**

通过 YAML 使用 CRD 配置日志采集

最近更新时间：2022-06-09 11:38:29

操作场景

用户不仅可以 使用控制台配置日志采集，还可通过自定义资源定义（CustomResourceDefinitions，CRD）的方式配置日志采集。CRD 支持采集容器标准输出、容器文件和主机文件，支持多种日志采集格式。支持投递到 CLS 和 CKafka 等不同消费端。

前提条件

已在容器服务控制台 的 **功能管理** 中开启日志采集，详情参见 开启日志采集。

创建 CRD 投递日志到 CLS

您只需要定义 LogConfig CRD 即可创建采集配置，log-agent 根据 LogConfig CRD 的变化修改相应的日志服务 CLS 日志主题，并设置绑定的机器组。CRD 的格式如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig ## 默认值
metadata:
name: test ## CRD资源名，在集群内唯一
spec:
clsDetail:
# 注意：topic指定后不允许修改
# 自动创建日志主题，需要同时指定日志集和主题的name
logsetName: test ## CLS日志集的name，若无该name的日志集，会自动创建，若有，会在该日志集下创建日志主题
topicName: test ## CLS日志主题的name，若无该name的日志主题，会自动创建

# 选择已有日志主题
topicId: xxxxxx-xx-xx-xx-xxxxxxxx ## CLS日志主题的ID，日志主题需要在CLS中提前创建，且没有被其它采集配置占用
logType: minimalist_log ## 日志采集格式，json_log代表 json 格式，delimiter_log代表分隔符格式，minimalist_log代表单行全文格式，multiline_log代表多行全文格式，fullregex_log代表完全正则格式
extractRule: ## 提取、过滤规则
...
inputDetail:
type: container_stdout ## 采集日志的类型，包括container_stdout（容器标准输出）、container_file（容器文件）、host_file（主机文件）

containerStdout: ## 容器标准输出
namespace: default ## 采集容器的kubernetes命名空间，如果不指定，代表所有命名空间
allContainers: false ## 是否采集指定命名空间中的所有容器的标准输出
container: xxx ## 采集日志的容器名，为空时，代表采集所有符合容器的日志名
includeLabels: ## 采集包含指定label的Pod
k8s-app: xxx ## 只采pod标签中配置"k8s-app=xxx"的pod产生的日志，与workloads、allContainers=true不能同时指定
workloads: ## 要采集的容器的Pod所属的kubernetes workload
- namespace: prod ## workload的命名空间
name: sample-app ## workload的名字
kind: deployment ## workload类型，支持deployment、daemonset、statefulset、job、cronjob
container: xxx ## 要采集的容器名，如果不指定，代表workload Pod中的所有容器

containerFile: ## 容器内文件
namespace: default ## 采集容器的kubernetes命名空间，必须指定一个命名空间
container: xxx ## 采集日志的容器名，为 * 时，代表采集所有符合容器的日志名
includeLabels: ## 采集包含指定label的Pod
```

```
k8s-app: xxx ## 只采pod标签中配置"k8s-app=xxx"的pod产生的日志，与workload不能同时指定
workload: ## 要采集的容器的Pod所属的kubernetes workload
name: sample-app ## workload的名字
kind: deployment ## workload类型，支持deployment、daemonset、statefulset、job、cronjob
logPath: /opt/logs ## 日志文件夹，不支持通配符
filePattern: app_*.log ## 日志文件名，支持通配符 * 和 ？，* 表示匹配多个任意字符，？表示匹配单个任意字符

hostFile: ## 主机文件
logPath: /opt/logs ## 日志文件夹，支持通配符
filePattern: app_*.log ## 日志文件名，支持通配符 * 和 ？，* 表示匹配多个任意字符，？表示匹配单个任意字符
customLablels:
k1: v1
```

⚠ 如果选择采集类型为“容器文件路径”时，对应的“容器文件路径”**不能为软链接**，否则会导致软链接的实际路径在采集器的容器内不存在，采集日志失败。

配置 CLS 日志解析格式

单行全文格式

单行全文日志是指一行日志内容为一条完整的日志。日志服务在采集的时候，将使用换行符 \n 来作为一条日志日志的结束符。为了统一结构化管理，每条日志都会存在一个默认的键值 __CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。详情请参见单行全文格式。

假设一条日志原始数据为：

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

LogConfig 配置参考示例如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
clsDetail:
topicId: xxxxxx-xx-xx-xx-xxxxxxxx
# 单行日志
logType: minimalist_log
```

采集到日志服务的数据为：

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

多行全文格式

多行全文日志是指一条完整的日志数据可能跨占多行（例如 Java stacktrace）。该情况下无法使用换行符 \n 作为日志的结束标识符，为了使日志系统明确区分每条日志，采用首行正则的方式进行匹配，当某行日志匹配预先设置的正则表达式，即为一条日志的开头，而下一行首出现则作为该条日志的结束标识符。多行全文也会设置一个默认的键值 __CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。详情请参见 多行全文格式。

假设一条多行日志原始数据为：

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
clsDetail:
topicId: xxxxxx-xx-xx-xx-xxxxxxxx
#多行日志
logType: multiline_log
extractRule:
#只有以日期时间开头的行才被认为是新一条日志的开头，否则就添加换行符\n并追加到当前日志的尾部
beginningRegex: \d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2},\d{3}\s.+
```

采集到日志服务的数据为：

```
\_\_CONTENT__:2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:\njava.lang.NullPointerException\n at com.test.logging.FooFactory.createFoo(FooFactory.java:15)\n at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

单行-完全正则格式

完全正则格式通常用来处理结构化的日志，指将一条完整日志按正则方式提取多个 key-value 的日志解析模式。详情请参见 完全正则格式。

假设一条日志原始数据为：

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703 "http://127.0.0.1/course/explore?filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum" "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0" 0.354 0.354
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
clsDetail:
topicId: xxxxxx-xx-xx-xx-xxxxxxxx
# 完全正则格式
logType: fullregex_log
extractRule:
# 正则表达式，会根据()捕获组提取对应的value
logRegex: (\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
beginningRegex: (\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
# 提取的key列表，与提取的value的一一对应
keys: ['remote_addr','time_local','request_method','request_url','http_protocol','http_host','status','request_length','body_bytes_sent','http_referer','http_user_agent','request_time','upstream_response_time']
```

采集到日志服务的数据为：

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
```

```
http_referer: http://127.0.0.1/course/explore?filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

多行-完全正则格式

多行-完全正则模式适用于日志文本中一条完整的日志数据跨占多行（例如 Java 程序日志），可按正则表达式提取为多个 key-value 键值的日志解析模式。若不需要提取 key-value，请参阅多行全文格式进行配置。详情请参见 多行-完全正则格式。

假设一条日志原始数据为：

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
clsDetail:
topicId: xxxxxx-xx-xx-xx-xxxxxxxx
#多行-完全正则格式
logType: multiline_fullregex_log
extractRule:
#行首完全正则表达式，只有以日期时间开头的行才被认为是新一条日志的开头，否则就添加换行符\n并追加到当前日志的尾部
beginningRegex: \[\d+-\d+-\w+:\d+:\d+,\d+\]\s\[\w+\]\s.*
#正则表达式，会根据()捕获组提取对应的value
logRegex: \[(\d+-\d+-\w+:\d+:\d+,\d+)\]\s\[(\w+)\]\s(.*)
# 提取的 key 列表，与提取的 value 的一一对应
keys:
- time
- level
- msg
```

根据提取的 key，采集到日志服务的数据为：

```
time： 2018-10-01T10:30:01,000`
level： INFO`
msg：java.lang.Exception: exception happened
at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

JSON 格式

JSON 格式日志会自动提取首层的 key 作为对应字段名。首层的 value 作为对应的字段值，以该方式将整条日志进行结构化处理，每条完整的日志以换行符 \n 为结束标识符。详情请参见 JSON 格式。

假设一条 JSON 日志原始数据为：

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34 +0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost":"unix:/tmp/php-cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","request":"POST /event/dispatch HTTP/1.1","xff":"-","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
clsDetail:
topicId: xxxxxx-xx-xx-xx-xxxxxxxx
# JSON格式日志
logType: json_log
```

采集到日志服务的数据为：

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

分隔符格式

分隔符日志是指一条日志数据可以根据指定的分隔符将整条日志进行结构化处理，每条完整的日志以换行符 \n 为结束标识符。日志服务在进行分隔符格式日志处理时，您需要为每个分开的字段定义唯一的 key。详情请参见 分隔符格式。

假设原始日志为：

```
10.20.20.10 ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/1.1 ::: 127.0.0.1 ::: 200 ::: 647 ::: 35 ::: http://127.0.0.1/
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
```

```
clsDetail:
topicId: xxxxxx-xx-xx-xx-xxxxxxxx
#分隔符日志
logType: delimiter_log
extractRule:
#分隔符
delimiter: ':::'
#提取的key列表，与被分割的字段一一对应
keys: ['IP','time','request','host','status','length','bytes','referer']
```

采集到日志服务的数据为：

```
IP: 10.20.20.10
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

创建 CRD 投递日志到 CKafka

当前支持通过配置 CRD 采集 TKE 上的 Pod 日志至自建 Kafka 或者 CKafka，需要按如下配置自行定义日志源及日志消费端，CRD 配置完成后，日志采集器会按规则进行日志采集。
CRD 具体配置如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig ## 默认值
metadata:
name: test ## CRD资源名，在集群内唯一
spec:
kafkaDetail:
kafkaType: ## 必填，kafka 类型，“ckafka” 表示ckafka，“” 表示自建 kafka
type: string
instanceId: ## ckafka实例ID
type: string
brokers: xxxxxx ## 必填，broker地址，一般是域名:端口，多个地址以“,”分隔
topic: xxxxxx ## 必填，topic名称
messageKey: ## 选填，指定pod字段作为key上传到指定分区
valueFrom:
fieldRef:
fieldPath: metadata.name
timestampKey: ## 时间戳的key，默认是@timestamp
timestampFormat: ## 时间戳的格式，默认是double
inputDetail:
type: container_stdout ## 采集日志的类型，包括container_stdout（容器标准输出）、container_file（容器文件）

containerStdout: ## 容器标准输出
namespace: default ## 采集容器的kubernetes命名空间，如果不指定，代表所有命名空间
allContainers: false ## 是否采集指定命名空间中的所有容器的标准输出
container: xxx ## 采集日志的容器名，此处可不填
includeLabels: ## 采集包含指定label的Pod
```

```
k8s-app: xxx ## 只采pod标签中配置"k8s-app=xxx"的pod产生的日志，与workloads、allContainers=true不能同时指定
workloads: ## 要采集的容器的Pod所属的kubernetes workload
- namespace: prod ## workload的命名空间
name: sample-app ## workload的名字
kind: deployment ## workload类型，支持deployment、daemonset、statefulset、job、cronjob
container: xxx ## 要采集的容器名，如果填空，代表workload Pod中的所有容器

containerFile: ## 容器内文件
namespace: default ## 采集容器的kubernetes命名空间，必须指定一个命名空间
container: xxx ## 采集日志的容器名，此处可填*
includeLabels: ## 采集包含指定label的Pod
k8s-app: xxx ## 只采pod标签中配置"k8s-app=xxx"的pod产生的日志，与workload不能同时指定
workload: ## 要采集的容器的Pod所属的kubernetes workload
name: sample-app ## workload的名字
kind: deployment ## workload类型，支持deployment、daemonset、statefulset、job、cronjob
logPath: /opt/logs ## 日志文件夹，不支持通配符
filePattern: app_*.log ## 日志文件名，支持通配符 * 和 ？，* 表示匹配多个任意字符，？表示匹配单个任意字符


...
```

采集日志的类型

容器标准输出

示例1：采集 default 命名空间中的所有容器的标准输出

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
inputDetail:
type: container_stdout
containerStdout:
namespace: default
allContainers: true
...
```

示例2：采集 production 命名空间中属于 ingress-gateway deployment 的 pod 中的容器的标准输出

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
inputDetail:
type: container_stdout
containerStdout:
allContainers: false
workloads:
- namespace: production
name: ingress-gateway
kind: deployment
...
```

示例3：采集 production 命名空间下 pod 标签中包含 “k8s-app=nginx” 的 pod 中的容器的标准输出

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
inputDetail:
type: container_stdout
containerStdout:
namespace: production
allContainers: false
includeLabels:
k8s-app: nginx
...
```

容器文件

示例1：采集 production 命名空间下属于 ingress-gateway deployment 的 pod 中的 nginx 容器中 /data/nginx/log/ 路径下名为 access.log 的文件

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
topicId: xxxxxx-xx-xx-xx-xxxxxxxx
inputDetail:
type: container_file
containerFile:
namespace: production
workload:
name: ingress-gateway
type: deployment
container: nginx
logPath: /data/nginx/log
filePattern: access.log
...
```

示例2：采集 production 命名空间下 pod 标签包含 “k8s-app=ingress-gateway” 的 pod 中的 nginx 容器中 /data/nginx/log/ 路径下名为 access.log 的文件

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
inputDetail:
type: container_file
containerFile:
namespace: production
includeLabels:
k8s-app: ingress-gateway
container: nginx
logPath: /data/nginx/log
filePattern: access.log
...
```

主机文件

示例：采集主机 /data/ 路径下所有 .log 文件

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
inputDetail:
type: host_file
hostFile:
logPath: /data
filePattern: *.log
...
```

元数据（Metadata）

容器标准输出（container_stdout）以及容器文件（container_file），除原始的日志内容外，还需携带容器场景的元数据（例如产生日志的容器 ID）一起上报至日志服务。方便用户查看日志时追溯来源或根据容器标识、特征（例如容器名及 labels）进行检索。
元数据如下表：

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

日志组件版本说明

最近更新时间：2022-03-15 10:15:10

日志组件介绍

日志组件是当用户开启容器服务内的日志服务时，腾讯云日志服务部署于用户集群内每个标准节点上的组件，用于采集容器服务产生的业务日志并写入腾讯云上的消费端，目前支持写入 CLS 和 Kafaka 。

日志服务相关组件如下：

名称	资源类型	说明
tke-log-agent	DeamonSet	每个 log-agent 的 Pod 中包含一个 controller 容器和一个 loglistener sidecar 容器，负责采集节点上所有容器产生的日志。
cls-provisioner	Deployment	每个集群一个实例，负责将 CRD 配置转换成 loglistener 可以理解的采集配置与 CLS 通信。
logconfigs.cls.cloud.tencent.com	CRD	-

log-agent 版本迭代记录

v1.0.8.1

类别	内容
Feature	-
Bugfix	修复 topic 替换时 topic id 为空，导致 logconifg 被删除重建的情况；topic name 前后一致。

v1.0.8

类别	内容
Feature	• 默认屏蔽采集 kube-system 下 loglistener 日志。 • 创建索引策略修改：只有自动创建 topic 的时候创建默认索引，其他场景不修改 topic 索引。 • 支持 kafka 采集器在消息中增加 metadata 信息。 • 支持 kafka 采集器解析方式 单行全文，JSON，多行全文。
Bugfix	• 修复在 workload 场景下采集标准输出，无法指定 container 问题。 • 添加 docker client, 获取 Storage Driver 如果配置文件没有，通过 client 去获取 info 信息拿到 Storage Driver。 • 修复采集容器文件场景下，指定 metadatalabel 错误问题。 • 修正获取 kubelet 根目录方案。 • 修复删除旧采集配置前缀设置错误，导致采集配置匹配错误。 • 修复当前 kafka 采集器中设定消息 timestampKey timestamp 失效的问题。

v1.0.7

类别	内容
Feature	• cls-provisioner 创建 topic 时，支持指定创建键值索引，包括索引名字、类型、分词以及是否开启统计；如果不支持则默认开启 pod_name,namespace,container_name 索引。 • 支持指定 metadatalabels, 将指定的 pod label 写入元数据采集，如果不支持，采集所有 pod label 为元数据。 • 支持自定义 CLS 云 API 服务后端地址。
Bugfix	-

V1.0.6

类别	内容
Feature	log-agent 支持用户自定义修改 kubelet 根目录和 docker 根目录。

类别	内容
Bugfix	-

V1.0.5

类别	内容
Feature	• 日志采集配置支持 label != 操作（exclude labels）。 • 支持日志服务只采集增量日志。 • 日志采集配置支持多选 namespace 和排除 namespace。 • log-agent 支持配置同 Key 不同 Value 的 pod labels。 • loglistener 支持参数可配置。
Bugfix	• 修复 log-agent 使用 configmap 作为 source 时的已知问题。 • 修复部分条件下采集器配置为空导致校验失败的问题。 • 修复删除日志规则时，采集器删除配置失败的问题。 • 解决 logConfig 配置的兼容性问题。

V1.0.1

类别	内容
Feature	• cls-provisioner 访问 CLS 的接口切换到云 API。 • 支持 TKE 日志采集投递到 ckafka，详情见 配置日志消费端为 ckafka。
Bugfix	-

V0.2.28

类别	内容
Feature	-
Bugfix	修复一个 Pod 对应多个 logconfig 问题。

V0.2.27

类别	内容
Feature	-
Bugfix	修复用户在 topic 上设置的提取模式在部分场景下被覆盖的问题。

V0.2.26

类别	内容
Feature	-
Bugfix	修复删除 stdout 类型的采集配置时，部分情况下无法创建 metadata 的问题。

V0.2.25

类别	内容
Feature	-
Bugfix	• 修复部分情况下 log-agent panic 问题。 • 修复 workload 缓存导致软连接删除问题。 • 修复 metadata 文件创建失败问题。

V0.2.24

类别	内容
Feature	-
Bugfix	• 修复在 pod 中的 container restart 的过程中，metadata 被误删除的问题。 • log-agent 启动前自动清理 LogAgentRootDir，避免脏数据污染。 • 修复极端场景导致的 log-agent 组件 panic。 • 修复 log-agent 重复挂载 /data 目录导致的启动失败。

日志组件版本升级

最近更新时间：2022-03-15 10:14:23

操作场景

容器服务运维中心提供日志组件版本升级的功能，若您已开启日志采集，腾讯云容器服务当前支持您在容器服务控制台的**运维功能管理**中，查看当前组件版本和进行组件版本的手动升级操作。

升级须知

- 升级属于**不可逆**操作。
- 仅支持向上升级容器服务提供的组件版本，默认升级至当前最新版本。
- 升级时，控制台将自动升级配套的 loglistener 版本和 provisioner 版本，并且将自动更新用户集群内的 CRD 资源，以便获得最新的日志功能。
- 版本详情请查看 日志组件版本迭代记录。

> 说明：
> 控制台当前仅支持标准集群升级，弹性集群暂不支持该功能。

操作步骤

1. 登录 容器服务控制台，单击左侧导航栏中**运维功能管理**。
2. 在“运维功能管理”页面中，在集群列表上方的选择地域和集群类型。若您的集群开启了日志采集功能，并且组件为可更新状态，控制台会提示“组件可升级”，如下图所示：

3. 选择您的集群，单击“设置”进入设置功能页面，并在“日志采集”栏，单击“编辑”。
4. 在“日志采集”详情中点击“升级组件”完成日志组件升级。如下图所示：

审计管理
集群审计

最近更新时间：2022-05-17 17:16:10

日志服务 CLS 为容器服务 TKE 产生的所有审计、事件数据提供**免费服务**至2022年6月30日。请选择自动创建日志集，或在已有日志集中选择自动创建日志主题。活动详情请参见 TKE 容器服务审计与事件中心日志免费说明。

简介

集群审计是基于 Kubernetes Audit 对 kube-apiserver 产生的可配置策略的 JSON 结构日志的记录存储及检索功能。本功能记录了对 kube-apiserver 的访问事件，会按顺序记录每个用户、管理员或系统组件影响集群的活动。

功能优势

集群审计功能提供了区别于 metrics 的另一种集群观测维度。开启集群审计后，Kubernetes 可以记录每一次对集群操作的审计日志。每一条审计日志是一个 JSON 格式的结构化记录，包括元数据（metadata）、请求内容（requestObject）和响应内容（responseObject）三个部分。其中元数据（包含了请求的上下文信息，例如谁发起的请求、从哪里发起的、访问的 URI 等信息）一定会存在，请求和响应内容是否存在取决于审计级别。通过日志可以了解到以下内容：

- 集群里发生的活动。
- 活动的发生时间及发生对象。
- 活动的触发时间、触发位置及观察点。
- 活动的结果以及后续处理行为。

阅读审计日志

```
{
"kind":"Event",
"apiVersion":"audit.k8s.io/v1",
"level":"RequestResponse",
"auditID":0a4376d5-307a-4e16-a049-24e017******,
"stage":"ResponseComplete",
// 发生了什么
"requestURI":"/apis/apps/v1/namespaces/default/deployments",
"verb":"create",
// 谁发起的
"user":{
"username":"admin",
"uid":"admin",
"groups":[
"system:masters",
"system:authenticated"
]
},
// 从哪里发起
"sourceIPs":[
"10.0.6.68"
],
"userAgent":"kubectl/v1.16.3 (linux/amd64) kubernetes/ald64d8",
// 发生了什么
"objectRef":{
"resource":"deployments",
"namespace":"default",
```

```
"name":"nginx-deployment",
"apiGroup":"apps",
"apiVersion":"v1"
},
// 结果是什么
"responseStatus":{
"metadata":{
},
"code":201
},
// 请求及返回具体信息
"requestObject":Object{...},
"responseObject":Object{...},
// 什么时候开始/结束
"requestReceivedTimestamp":"2020-04-10T10:47:34.315746Z",
"stageTimestamp":"2020-04-10T10:47:34.328942Z",
// 请求被接收/拒绝的原因是什么
"annotations":{
"authorization.k8s.io/decision":"allow",
"authorization.k8s.io/reason":""
}
}
```

TKE 集群审计策略

审计级别（level）

和一般日志不同，kuberenetes 审计日志的级别更像是一种 verbose 配置，用来标示记录信息的详细程度。一共有4个级别，可参考以下表格内容：

参数	说明
None	不记录。
Metadata	记录请求的元数据（例如：用户、时间、资源、操作等），不包括请求和响应的消息体。
Request	除了元数据外，还包括请求消息体，不包括响应消息体。
RequestResponse	记录所有信息，包括元数据以及请求、响应的消息体。

审计阶段（stage）

记录日志可以发生在不同的阶段，参考以下表格内容：

参数	说明
RequestReceived	一收到请求就记录。
ResponseStarted	返回消息头发送完毕后记录，只针对 watch 之类的长连接请求。
ResponseComplete	返回消息全部发送完毕后记录。
Panic	内部服务器出错，请求未完成。

TKE 审计策略

TKE 默认收到请求即会记录审计日志，且大部分的操作会记录 RequestResponse 级别的审计日志。但也会存在如下情况：

- get、list 和 watch 会记录 Request 级别的日志。
- 针对 secrets 资源、configmaps 资源或 tokenreviews 资源的请求会在 Metadata 级别记录。

以下请求将不会进行记录日志：

- system:kube-proxy 发出的监视 endpoints 资源、services 资源或 services/status 资源的请求。
- system:unsecured 发出的针对 kube-system 命名空间中 configmaps 资源的 get 请求。
- kubelet 发出的针对 nodes 资源或 nodes/status 资源的 get 请求。
- system:nodes 组中的任何身份发出的针对 nodes 资源或 nodes/status 资源的 get 请求。
- system:kube-controller-manager、system:kube-scheduler 或 system:serviceaccount:endpoint-controller 发出的针对 kube-system 命名空间中 endpoints 资源的 get 和 update 请求。
- system:apiserver 发出的针对 namespaces 资源、namespaces/status 资源或 namespaces/finalize 资源的 get 请求。
- 对与 /healthz*、/version 或/swagger* 匹配的网址发出的请求。

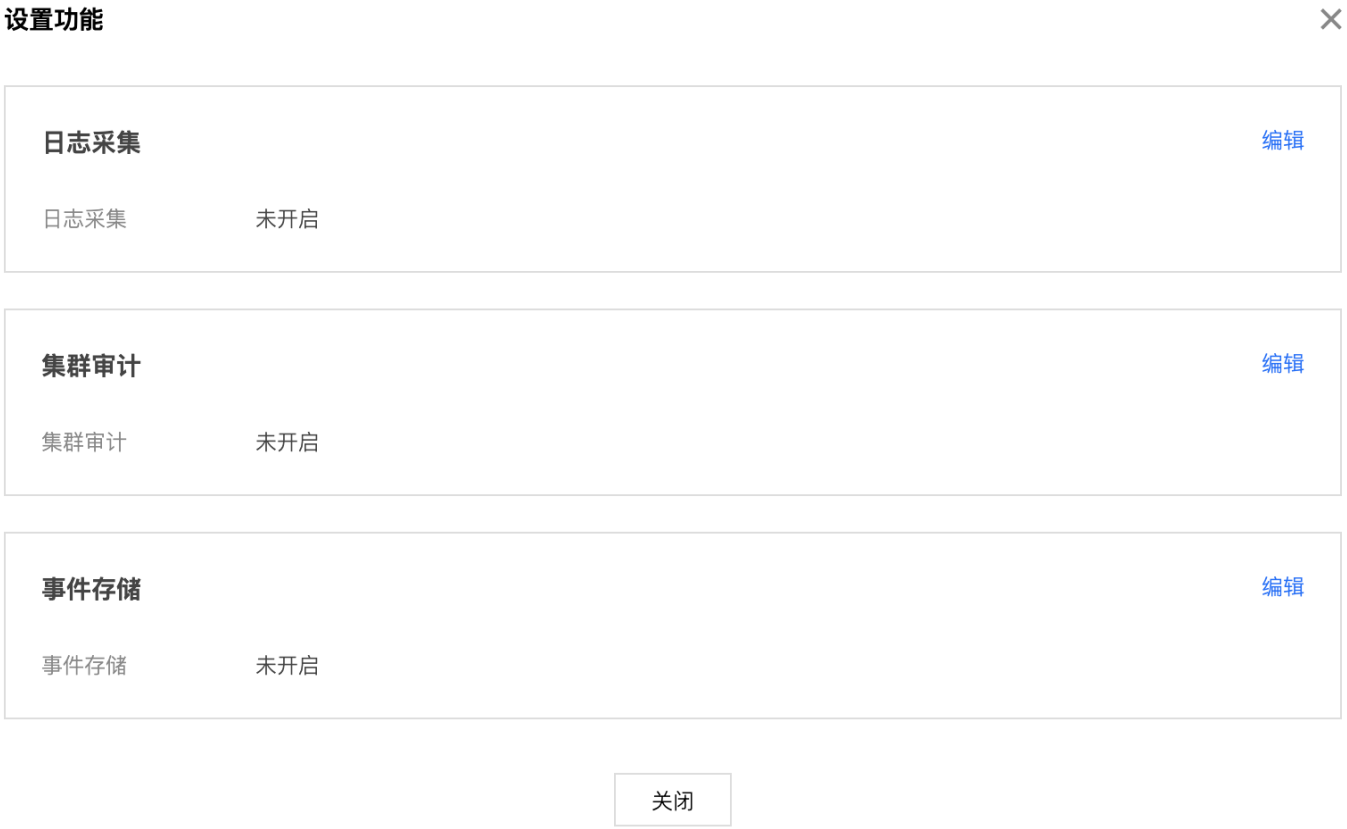
操作步骤

开启集群审计

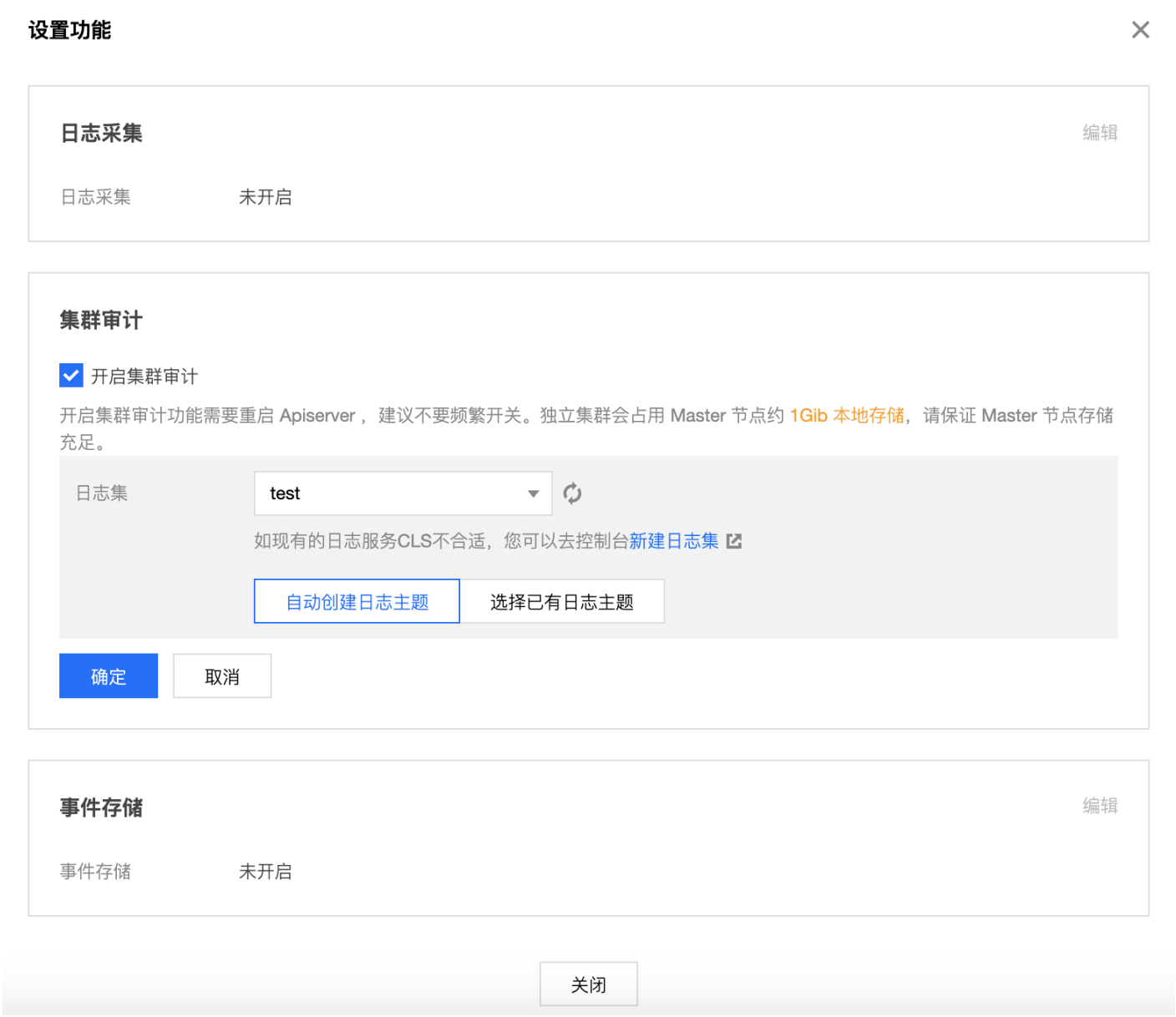
> ⚠ 注意：
>
> - 开启集群审计功能需要重启 kube-apiserver ，建议不要频繁开关。
> - 独立集群会占用 Master 节点约1Gib本地存储，请保证 Master 节点存储充足。

1. 登录 腾讯云容器服务控制台 。
2. 选择左侧导航栏中的**运维功能管理**，进入**功能管理**页面。
3. 在“功能管理”页面上方选择地域，单击希望开启集群审计的集群右侧的**设置**。如下图所示：

4. 在弹出的“设置功能”窗口，单击“集群审计”功能右侧的**编辑**。



5. 勾选**开启集群审计**，选择存储审计日志的日志集和日志主题，推荐选择**自动创建日志主题**。



6. 单击**确定**即可开启集群审计功能。

审计仪表盘

最近更新时间：2022-06-14 11:39:23

操作场景

容器服务 TKE 为用户提供了开箱即用的审计仪表盘。在集群开启集群审计功能后，TKE 将自动为该集群配置审计总览、节点操作总览、K8S 对象操作概览、聚合检索仪表盘。还支持用户自定义配置过滤项，同时内置 CLS 的全局检索，方便用户观测和检索各类集群操作，以便于及时发现和定位问题。

功能介绍

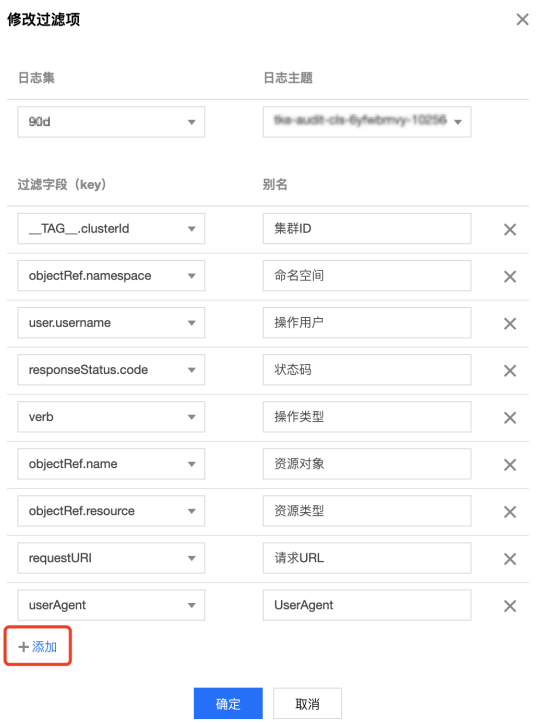
审计检索中配置了五个大盘，分别是“审计总览”、“节点操作总览”、“K8S对象操作概览”、“聚合检索”、“全局检索”。请按照以下步骤进入“审计检索”页面，开始使用对应功能：

1. 登录 容器服务控制台 。
2. 开启集群审计功能，详情请参见 集群审计。
3. 选择导航栏左侧**日志管理 > 审计日志**，进入“审计检索”页面。

审计总览

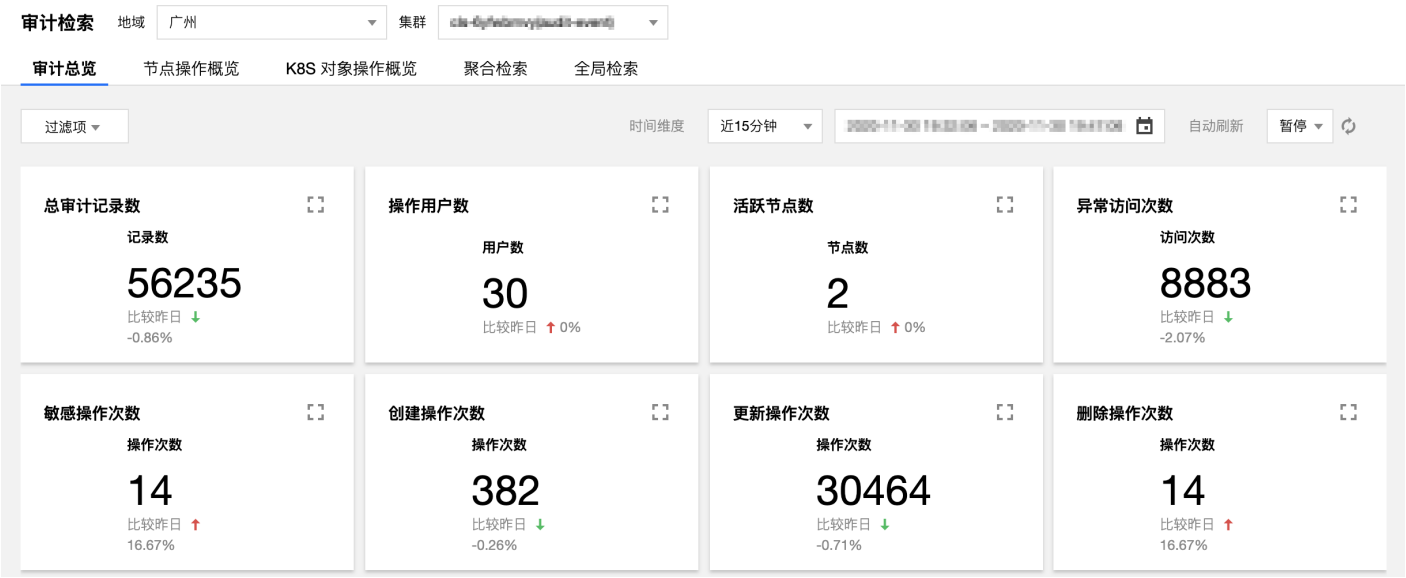
当您想观测整个集群 APIserver 操作时，可在“审计总览”页面设置过滤条件，查看核心审计日志的汇总统计信息，并展示一个周期内的数据对比。例如，核心审计日志的统计数、分布情况、重要操作趋势等。
在过滤项中，您可根据自己需求进行个性化配置（最多可设置10个过滤项），如下图所示：

您可修改过滤项自定义字段，如下图所示：

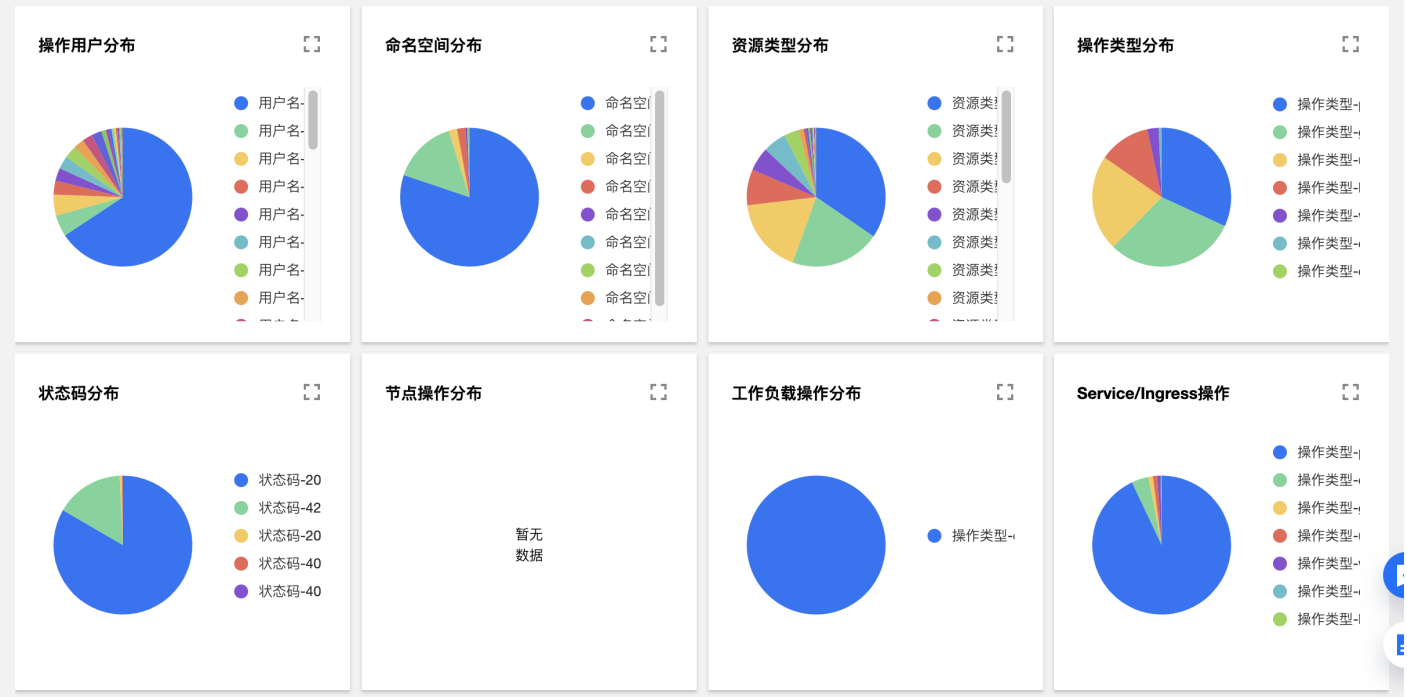


您还可在该页面中查看更多统计信息，如下所示：

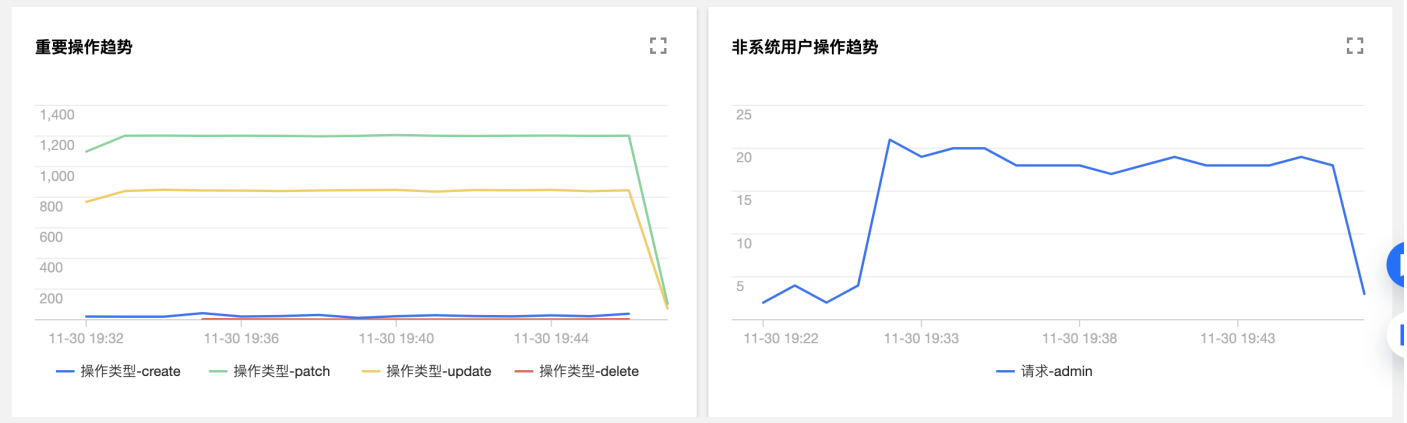
- 核心审计日志的统计数仪表盘：



- 分布情况仪表盘：

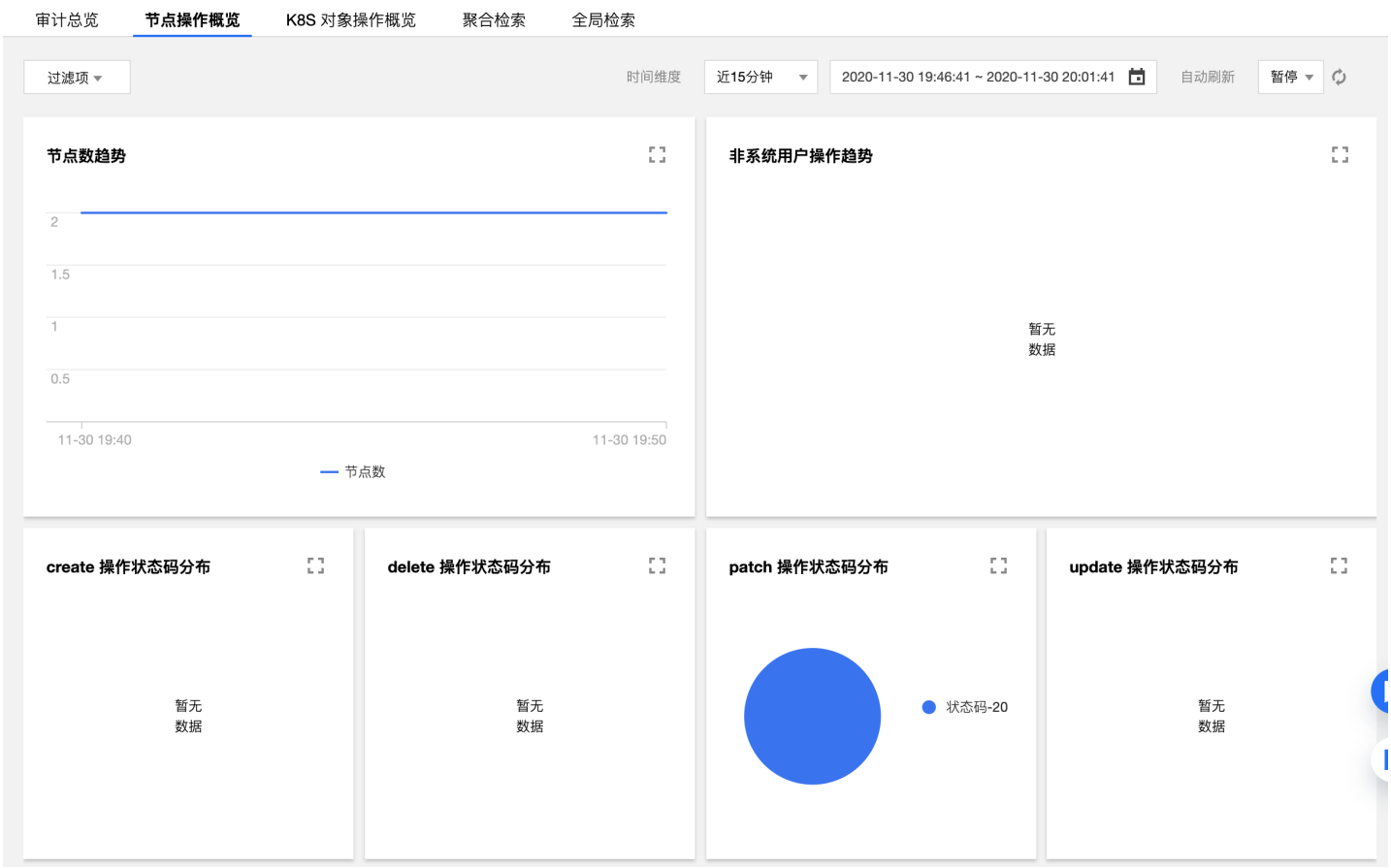


- 重要操作趋势仪表盘：



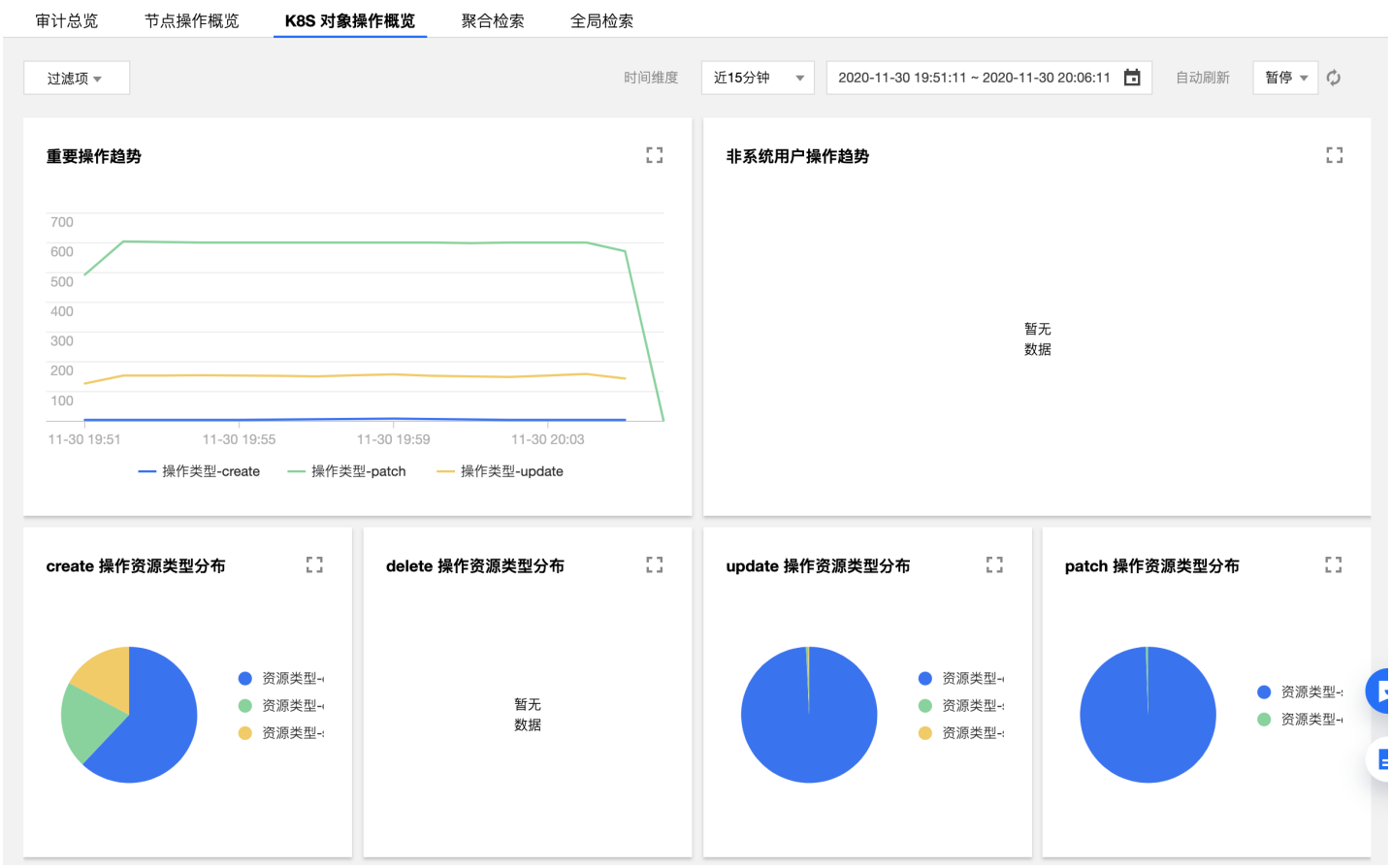
节点操作总览

当您需要排查节点相关问题时，可在“节点操作总览”页面设置过滤条件，查看各类节点操作相关的仪表，包括 create、delete、patch、update、封锁、驱逐等。如下图所示：



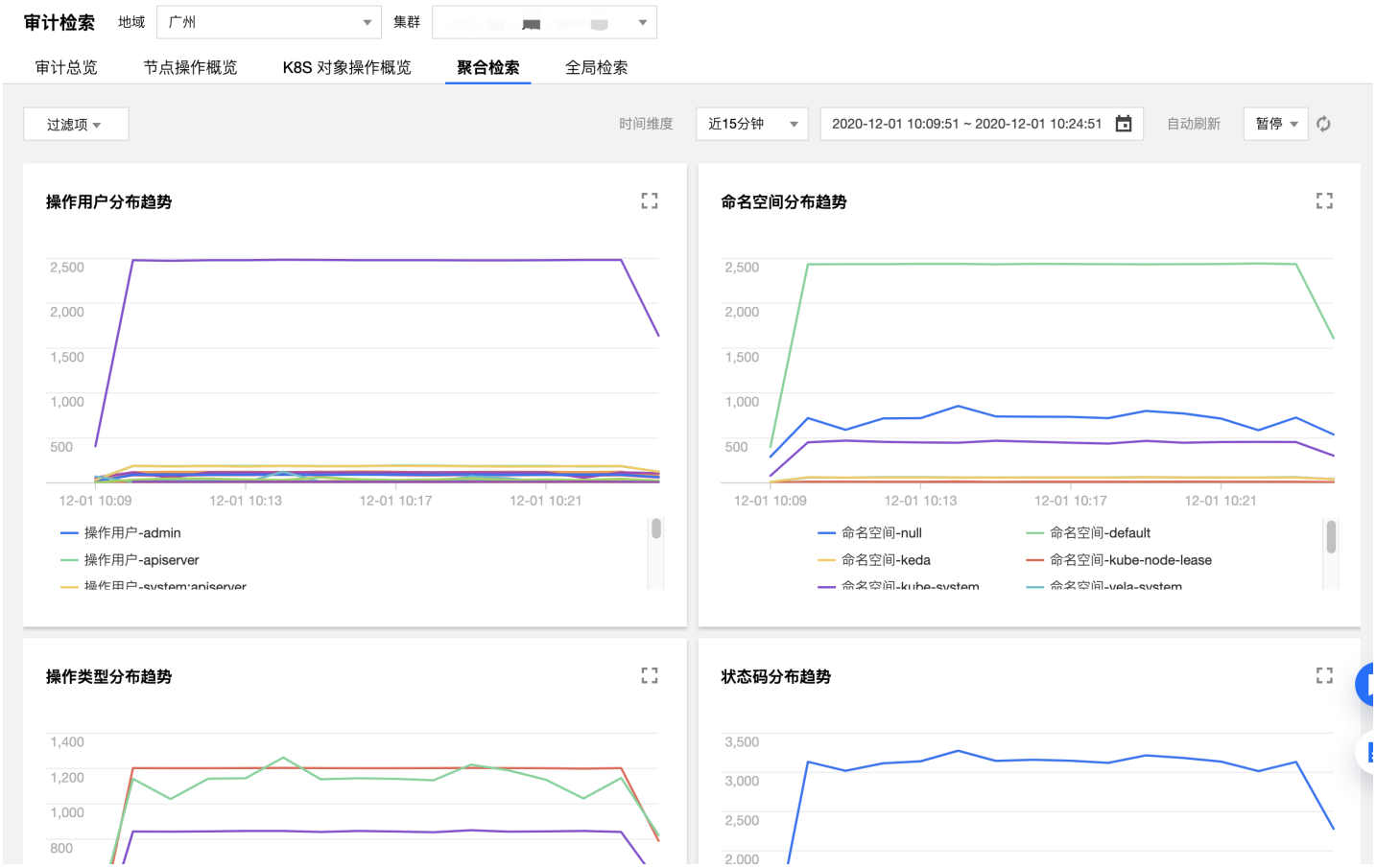
K8S 对象操作概览

当您需要排查 K8S 对象（例如某个工作负载）的相关问题时，可在 “K8S 对象操作概览”页面设置过滤条件，查看各类 K8S 对象的操作概览、对应的操作用户、相应的审计日志列表，以查找更多的细节。如下图所示：



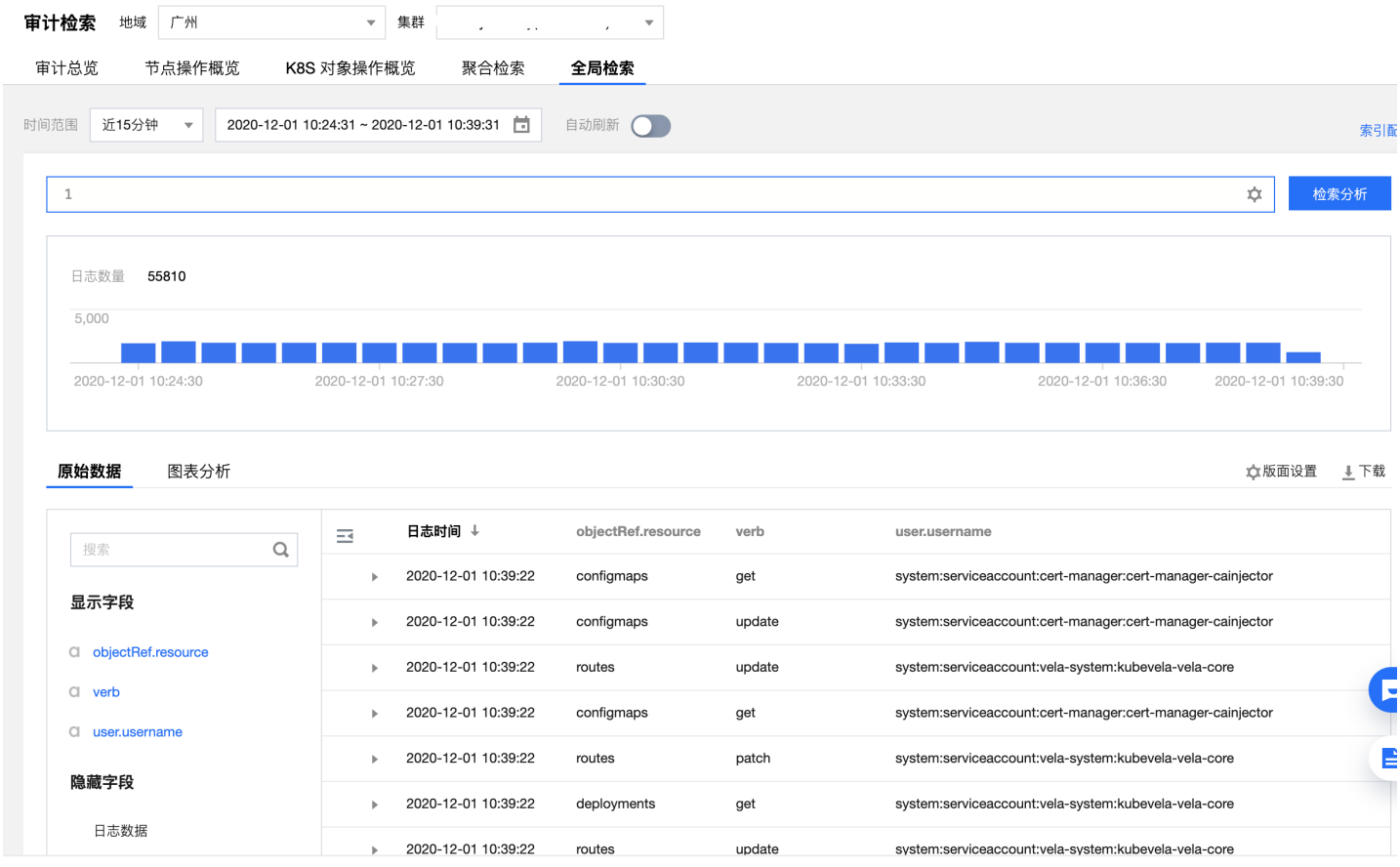
聚合检索

当您想观测某个维度下审计日志的分布趋势，可在“聚合检索”页面设置过滤条件，查看各类重要操作的时序图。纬度包括操作用户、命名空间、操作类型、状态码、资源类型以及相应的审计日志列表。如下图所示：



全局检索

全局检索仪表盘中内嵌了 CLS 的检索分析页面，方便用户在容器服务控制台 也能快速检索全部审计日志。如下图所示：



基于仪表盘配置告警

您可以通过以上预设的仪表盘配置告警，达到您所设置的条件则触发告警。操作详情如下：

1. 单击需要配置告警的仪表盘右侧的**快速添加告警**。
2. 在 **日志服务控制台**>**告警策略** 中新建告警策略。详情可参见 配置告警策略。

事件管理
事件仪表盘

最近更新时间：2022-04-18 11:44:23

操作场景

容器服务 TKE 为用户提供了开箱即用的事件仪表盘。在集群开启事件存储功能后，TKE 将自动为集群配置各类事件总览大盘和异常事件的聚合检索分析仪表盘。还支持用户自定义配置过滤项，同时内置 CLS 的事件全局检索，实现在容器服务控制台 全面观测、查找、分析、定位问题的能力。

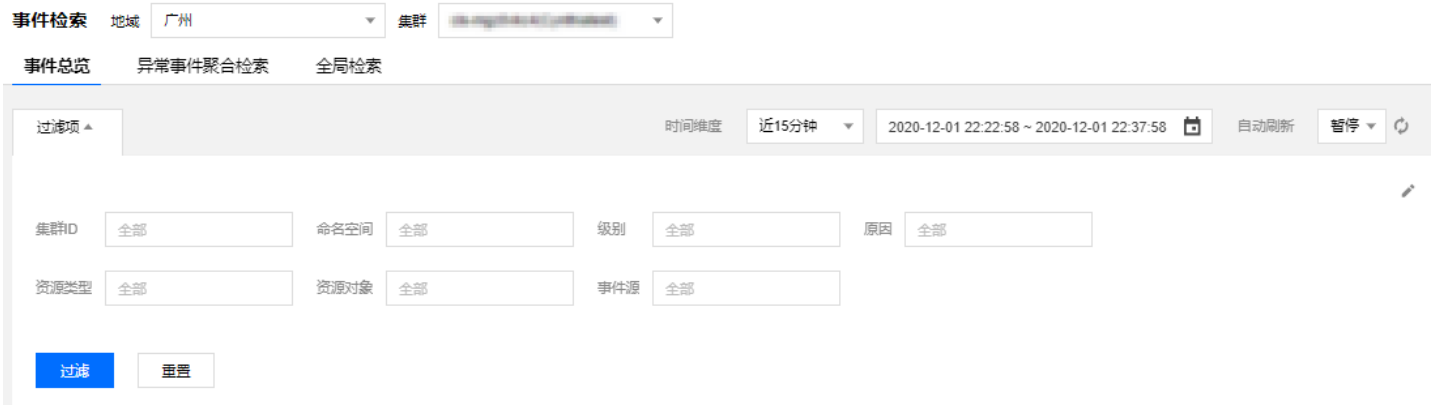
功能介绍

事件检索中配置了三个大盘，分别是“事件总览”、“异常事件聚合检索”、“全局检索”。请按照以下步骤进入“事件检索”页面，开始使用对应功能：

1. 登录 容器服务控制台 。
2. 开启“事件存储功能”，详情请参见 事件存储。
3. 选择导航栏左侧**集群运维** > **事件检索**，进入“事件检索”页面。

事件总览

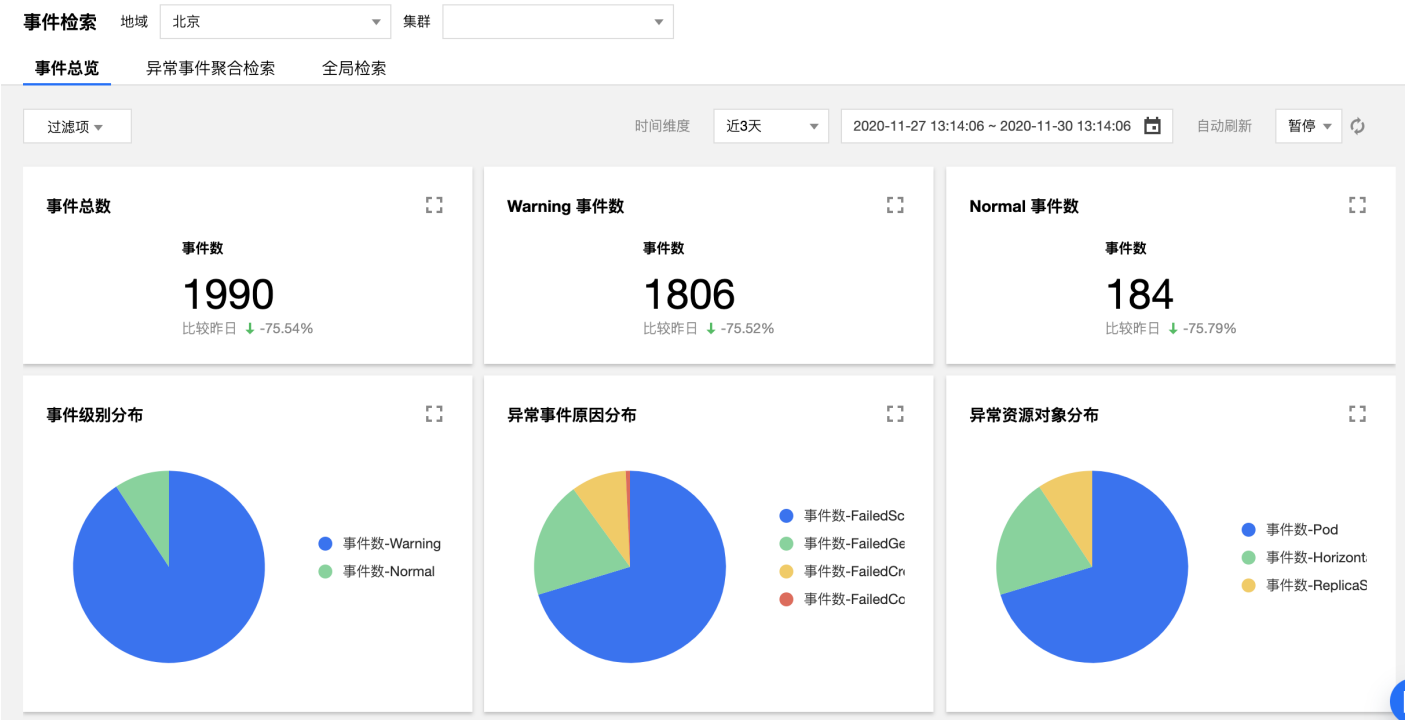
在“事件总览”页面，可根据时间、命名空间、级别、原因、资源类型、资源对象等维度过滤事件，查看核心事件的汇总统计信息，并展示一个周期内的数据对比。例如，事件总数及分布情况、节点异常、Pod OOM、重要事件趋势等仪表盘以及异常 TOP 事件列表。
在过滤项中，您可根据自己需求进行个性化配置（最多可设置10个过滤项）。如下图所示：



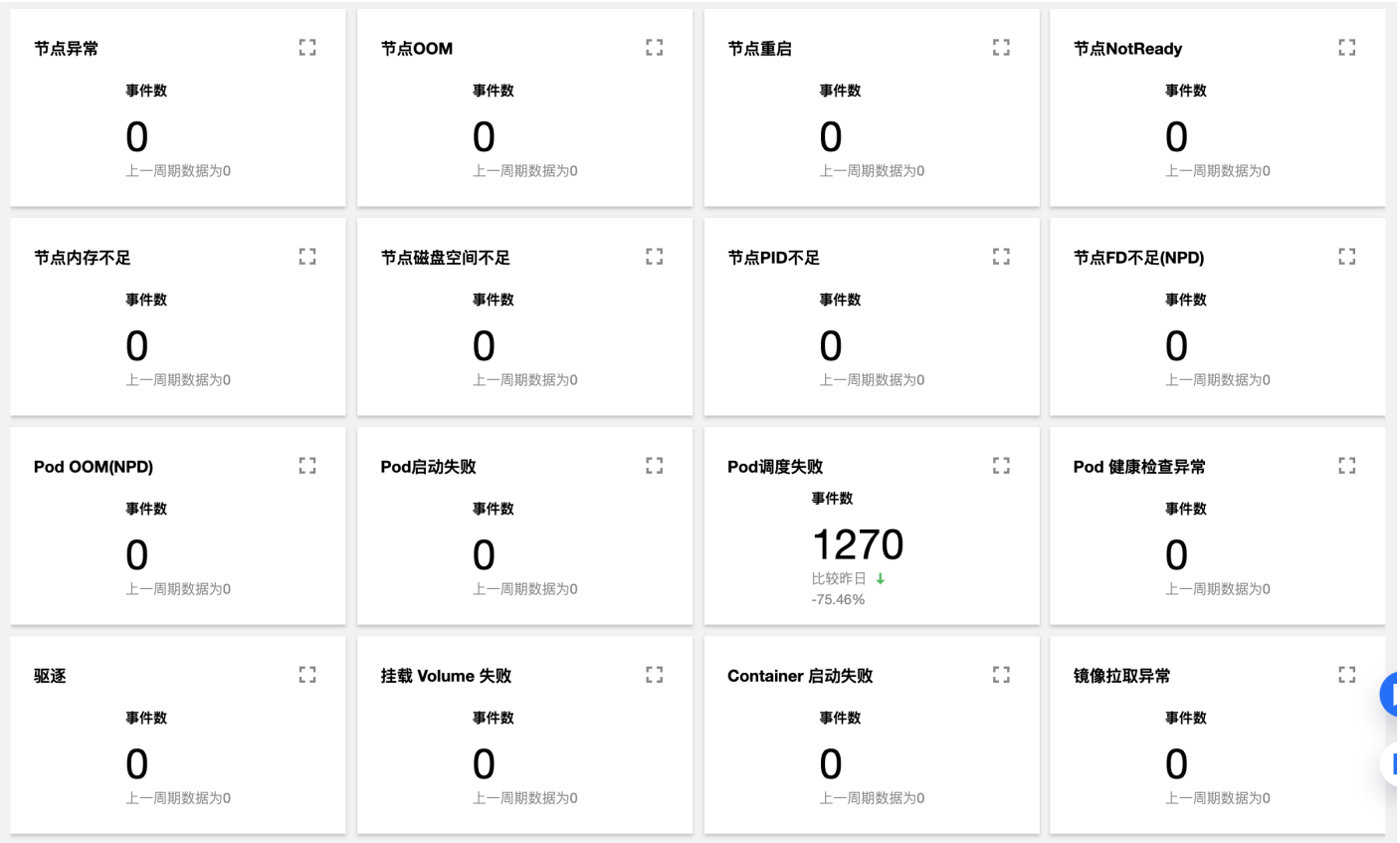
您可修改过滤项自定义字段，如下图所示：

您还可在该页面中查看更多统计信息，如下所示：

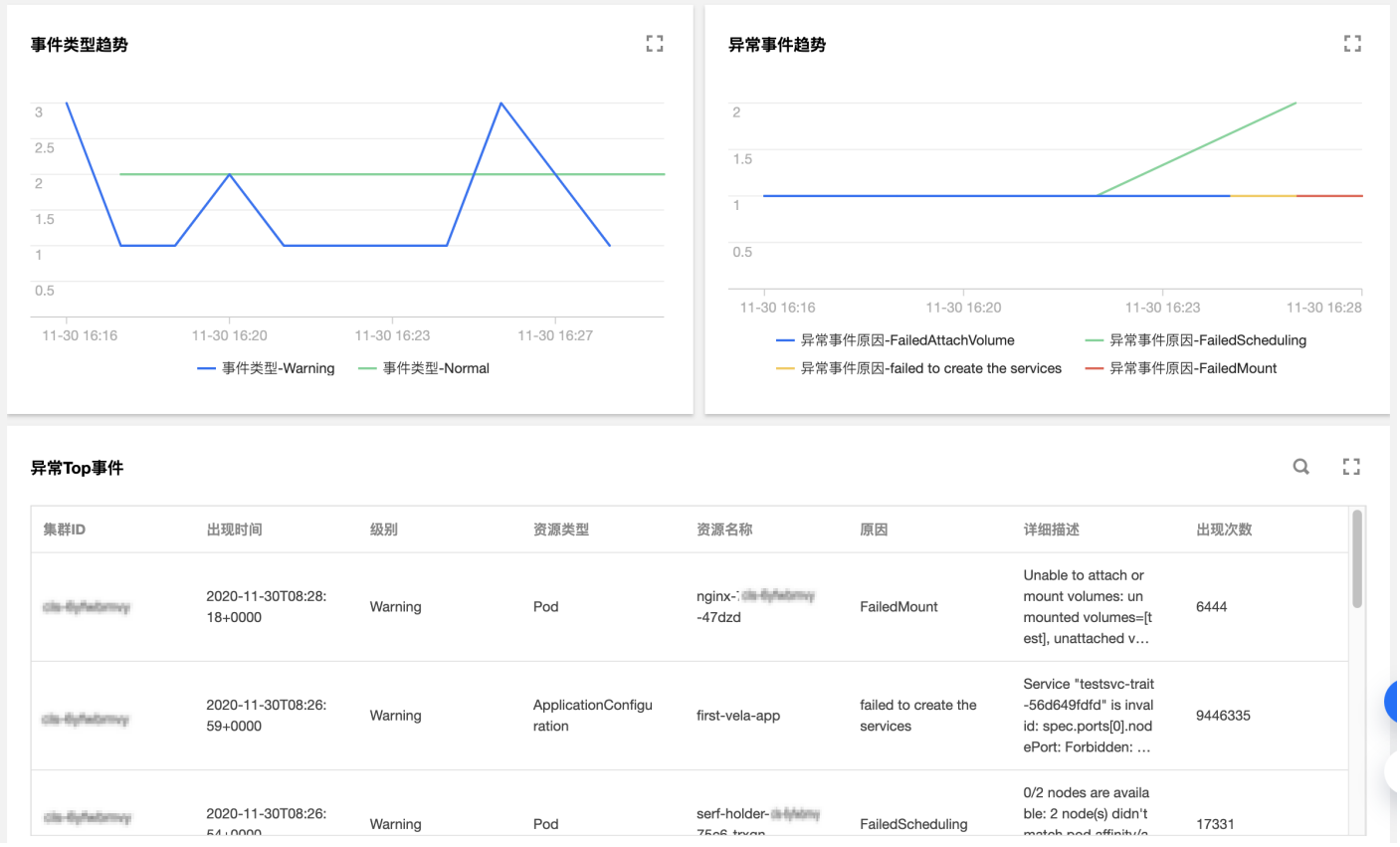
- 事件总数及级别分布情况，异常事件的原因、对象分布情况检索如下图所示：



- 各类常见事件的汇总信息检索如下图所示：

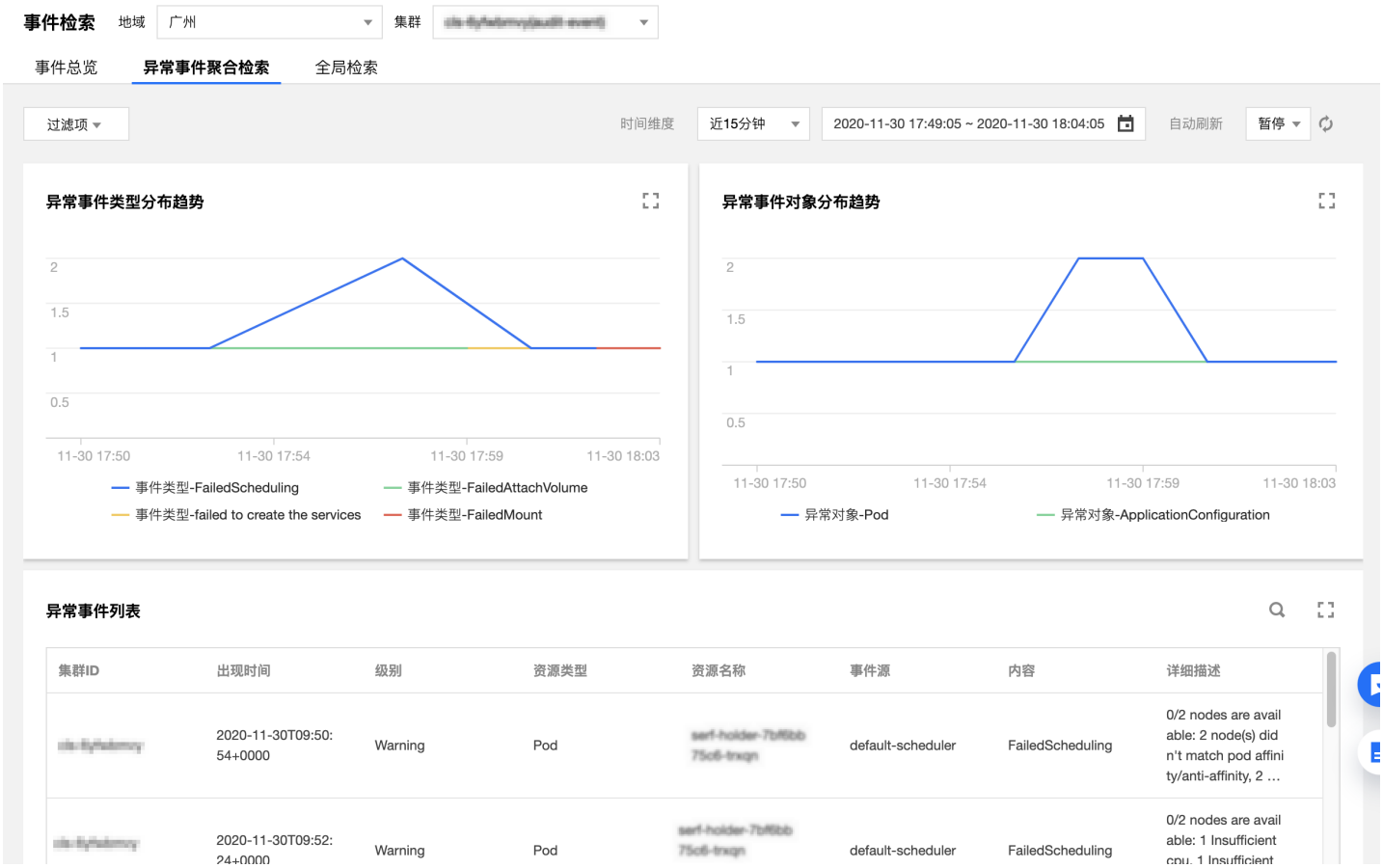


- 事件趋势及异常 TOP 事件列表检索如下图所示：



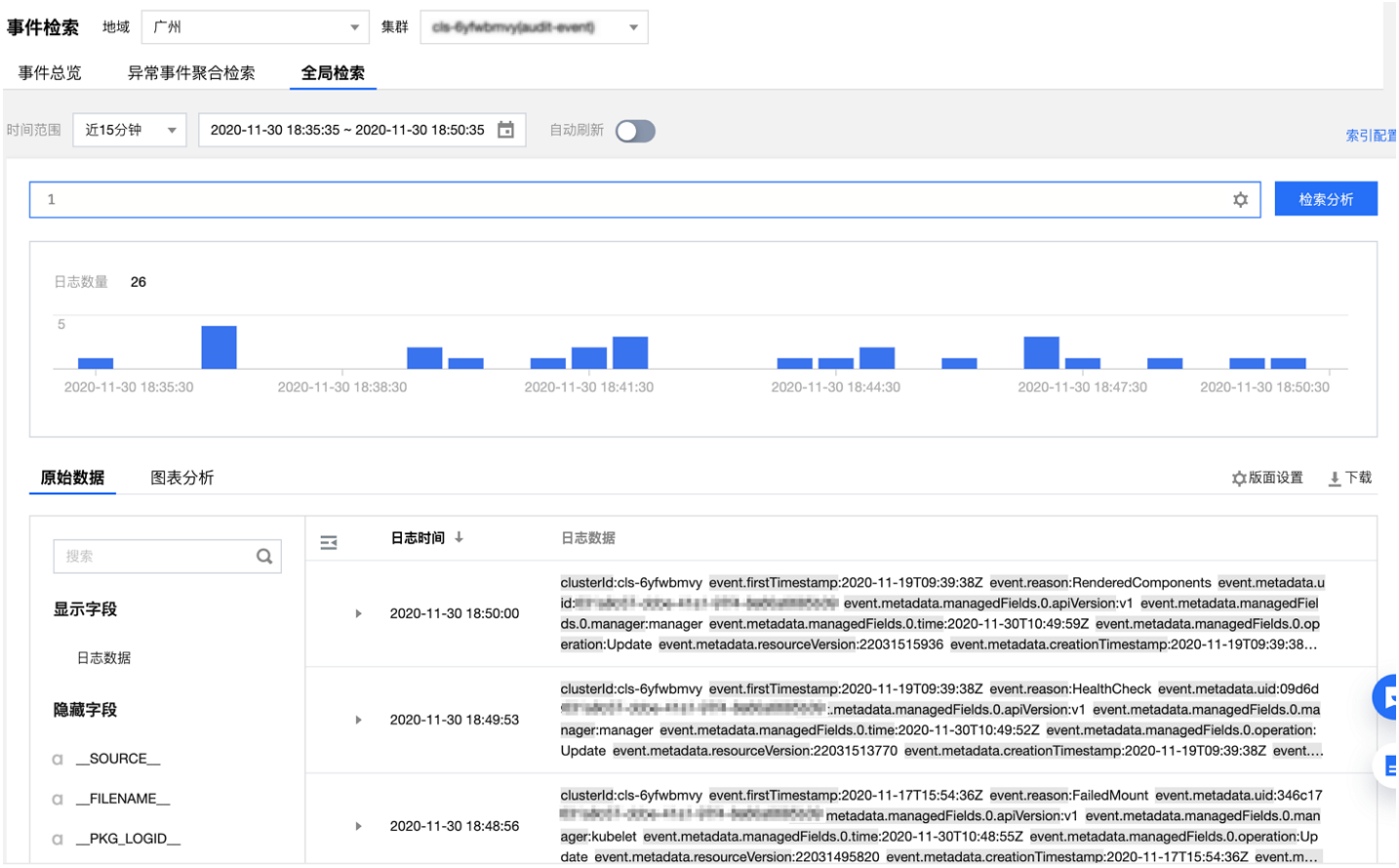
异常事件聚合检索

在“异常事件聚合检索”页面设置过滤条件，查看某个时间段内各类异常事件的 reason 和 object 分布趋势。在趋势下方展示了可供搜索的异常事件列表，帮助您快速定位到问题。如下图所示：



全局检索

全局检索仪表盘中内嵌了 CLS 的检索分析页面，方便用户在容器服务控制台 也能快速检索全部事件。如下图所示：



基于仪表盘配置告警

您可以通过以上预设的仪表盘配置告警，达到您所设置的条件则触发告警。操作详情如下：

1. 单击需要配置告警的仪表盘右侧的**快速添加告警**。
2. 在 **日志服务控制台**>**告警策略** 中新建告警策略。详情可参见 **配置告警策略**。

事件存储

最近更新时间：2022-06-09 10:44:20

日志服务 CLS 为容器服务 TKE 产生的所有审计、事件数据提供**免费服务**至2022年06月30日。请选择自动创建日志集，或在已有日志集中选择自动创建日志主题。活动详情请参见 TKE 容器服务审计与事件中心日志免费说明。

操作场景

Kubernetes Events 包括了 Kubernetes 集群的运行和各类资源的调度情况，对维护人员日常观察资源的变更以及定位问题均有帮助。TKE 支持为您的所有集群配置事件持久化功能，开启本功能后，会将您的集群事件实时导出到配置的存储端。TKE 还支持使用腾讯云提供的 PAAS 服务或开源软件对事件流水进行检索。本文档指导您如何开启集群事件持久化存储。

操作步骤

开启事件存储

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，选择**运维功能管理**。
3. 在“功能管理”页面上方选择地域，单击需要开启事件存储的集群右侧的**设置**。如下图所示：

4. 在“设置功能”页面，单击事件存储**编辑**。
5. 在“事件存储”编辑页面，勾选**开启事件存储**，并配置日志集和日志主题。如下图所示：

注意：
一个日志集最多只能有10个日志主题。若选择自动创建日志主题，请保证该日志集下未满10个日志主题。

单击确定，即可开启事件存储。

更新日志集或日志主题

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，选择**运维功能管理**。
3. 在“功能管理”页面上方选择地域，单击需要开启事件存储的集群右侧的**设置**。如下图所示：

4. 在“设置功能”页面，单击事件存储**编辑**。
5. 在“事件存储”编辑页面，重新选择日志集和日志主题。单击**确定**即可更新日志集和日志主题。

关闭事件存储

1. 登录 容器服务控制台 。
2. 在左侧导航栏中，，选择**运维功能管理**。
3. 在“功能管理”页面上方选择地域，单击需要开启事件存储的集群右侧的**设置**。如下图所示：

4. 在“设置功能”页面，单击事件存储**编辑**。
5. 在“事件存储”编辑页面，取消勾选**开启事件存储**。如下图所示，

单击确定，即可关闭事件存储。

健康检查

最近更新时间：2022-01-17 14:52:08

操作场景

集群健康检查功能是腾讯云容器服务（Tencent Kubernetes Engine，TKE）为集群提供检查各个资源状态及运行情况的服务，检查报告将详细展示组件、节点、工作负载的状态和配置的检查内容。若出现异常项，可进行异常详情描述，并自动分析异常级别、异常原因、异常影响和修复建议等。

> ⚠ 注意：
> 在健康检查过程中，您的集群内会自动新建 namespace tke-cluster-inspection，并安装一个 Daemonset 进行节点信息采集，检查结束后均会被自动删除。

主要检查项目

检查类别	检查项	检查内容	仅独立集群
资源状态	kube-apiserver 的状态	检测组件是否正在运行，如果组件以 Pod 形式运行，则检测其24小时内是否重启过。	是
	kube-scheduler 的状态		是
	kube-controller-manager 的状态		是
	etcd 的状态		是
	kubelet 的状态		否
	kube-proxy 的状态		否
	dockerd 的状态		否
	master 节点的状态	检测节点状态是否 Ready 且无其他异常情况，如内存不足，磁盘不足等。	是
	worker 节点的状态	检测节点状态是否 Ready 且无其他异常情况，如内存不足，磁盘不足等。	否
	各个工作负载的状态	检测工作负载当前可用 Pod 数是否符合其期望目标 Pod 数。	否
运行情况	kube-apiserver 的参数配置	根据 master 节点配置检测以下参数： • max-requests-inflight：给定时间内运行的非变更类请求的最大值。 • max-mutating-requests-inflight：给定时间内运行的变更类请求的最大值。	是
	kube-scheduler 的参数配置	根据 master 节点配置检测以下参数： • kube-api-qps：请求 kube-apiserver 使用的 QPS。 • kube-api-burst：和 kube-apiserver 通信的时候最大 burst 值。	是
	kube-controller-manager 的参数配置	根据 master 节点配置检测以下参数： • kube-api-qps：请求 kube-apiserver 使用的 QPS。 • kube-api-burst：和 kube-apiserver 通信的时候最大 burst 值。	是
	etcd 的参数配置	根据 master 节点配置检测以下参数： quota-backend-bytes：存储大小。	是
	master 节点的配置合理性	检测当前 master 节点配置是否足以支撑当前的集群规模。	是

检查类别	检查项	检查内容	仅独立集群
	node 高可用	检测目前集群是否是单节点集群； 检测当前集群节点是否支持多可用区容灾。 即当一个可用区不可用后，其他可用区的资源总和是否足以支撑当前集群业务规模。	否
	工作负载的 Request 和 Limit 配置	检测工作负载是否有未设置资源限制的容器，配置资源限制有益于完善资源规划、Pod 调度、集群可用性等。	否
	工作负载的反亲和性配置	检测工作负载是否配置了亲和性或者反亲和性，配置反亲和性有助于提高业务的高可用性。	否
	工作负载的 PDB 配置	检测工作负载是否配置了 PDB，配置 PDB 可避免您的业务因驱逐操作而不可用。	否
	工作负载的健康检查配置	检测工作负载是否配置了健康检查，配置健康检查有助于发现业务异常。	否
	HPA-IP 配置	当前集群剩余的 Pod IP 数目是否满足 HPA 扩容的最大数。	否

操作步骤

1. 登录 容器服务控制台，选择左侧导航栏中的**集群运维** > **健康检查**。
2. 进入“健康检查”页面，选择需要健康检查的集群，并为其选择合适的检查方式。

   健康检查的三种方式分别为批量检查、立即检查和自动检查。
   - **批量检查**：适用于同时检查多个集群。
   - **立即检查**：适用于只检查一个集群。
   - **自动检查**：适用于需要周期性检查的集群。选择需要周期检查的集群，单击**自动检查**。如下图所示：

在“自动检查设置”弹窗中，可根据您的需求设置开启状态、检查周期和时刻。如下图所示：

3. 选择好检查方式之后，等待检查完成，可查看检查进度。如下图所示：

4. 检查完成后，可单击**查看结果**查看检查报告。如下图所示：

在检查报告页面，选择**资源状态**和**运行情况**分别查看资源状态和异常情况，单击**检查内容**可展示具体的检查内容，单击**异常**可查看异常级别、异常描述、异常原因、异常影响和修复建议。如下图所示：

监控与告警
监控告警概述

最近更新时间：2022-04-18 14:15:10

概述

腾讯云容器服务 TKE 提供集群、节点、工作负载、Pod、Container 5个层面的监控数据收集和展示功能。良好的监控环境为腾讯云容器服务高可靠性、高可用性和高性能提供重要保证。通过告警配置您可以为不同资源收集不同维度的监控数据，方便掌握资源的使用状况，轻松定位故障。

收集监控数据有助于您建立容器集群性能的正常标准。通过在不同时间、不同负载条件下测量容器集群的性能并收集历史监控数据，您可以较为清楚地了解容器集群和服务运行时的正常性能，并能快速根据当前监控数据判断服务运行时是否处于异常状态，及时找出解决问题的方法。例如，您可以监控服务的 CPU 利用率、内存使用率和磁盘 I/O。

监控

容器服务的监控功能使用指引请参见 查看监控数据。
目前覆盖的监控指标请参见 监控及告警指标列表。

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告警，告警配置指引请参见 设置告警。
目前覆盖的告警指标请参见 监控及告警指标列表。

相关说明

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 云监控控制台 提供的基础资源监控（如云服务器、块存储、负载均衡等）使用，以保证更细的指标覆盖。

若腾讯云容器服务提供的基础监控能力无法满足您的诉求，您可以使用腾讯云推出的 云原生监控 服务。云原生监控服务致力于提供轻量、稳定、高可用的 Prometheus 监控服务。保留原生 Prometheus 的特性，支持采集自定义指标，支持多集群监控，支持千万级指标上报，提供基于 Grafana 的优秀可视化能力和默认面板，提供稳定的多渠道的告警能力，无侵入式架构几乎不占用您的集群资源，高度自由化的配置方式助您构建云原生场景下最适合自己的监控平台。

查看监控数据

最近更新时间：2022-04-22 17:19:41

操作场景

腾讯云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。

1. 选择需要查看的集群ID/名称，进入该集群的管理页面。
2. 展开**节点管理**，即可查看节点和 Master&Etcd 节点的监控信息。

- 选择**节点 > 监控**，即可进入**节点监控**页面，查看监控信息。如下图所示：

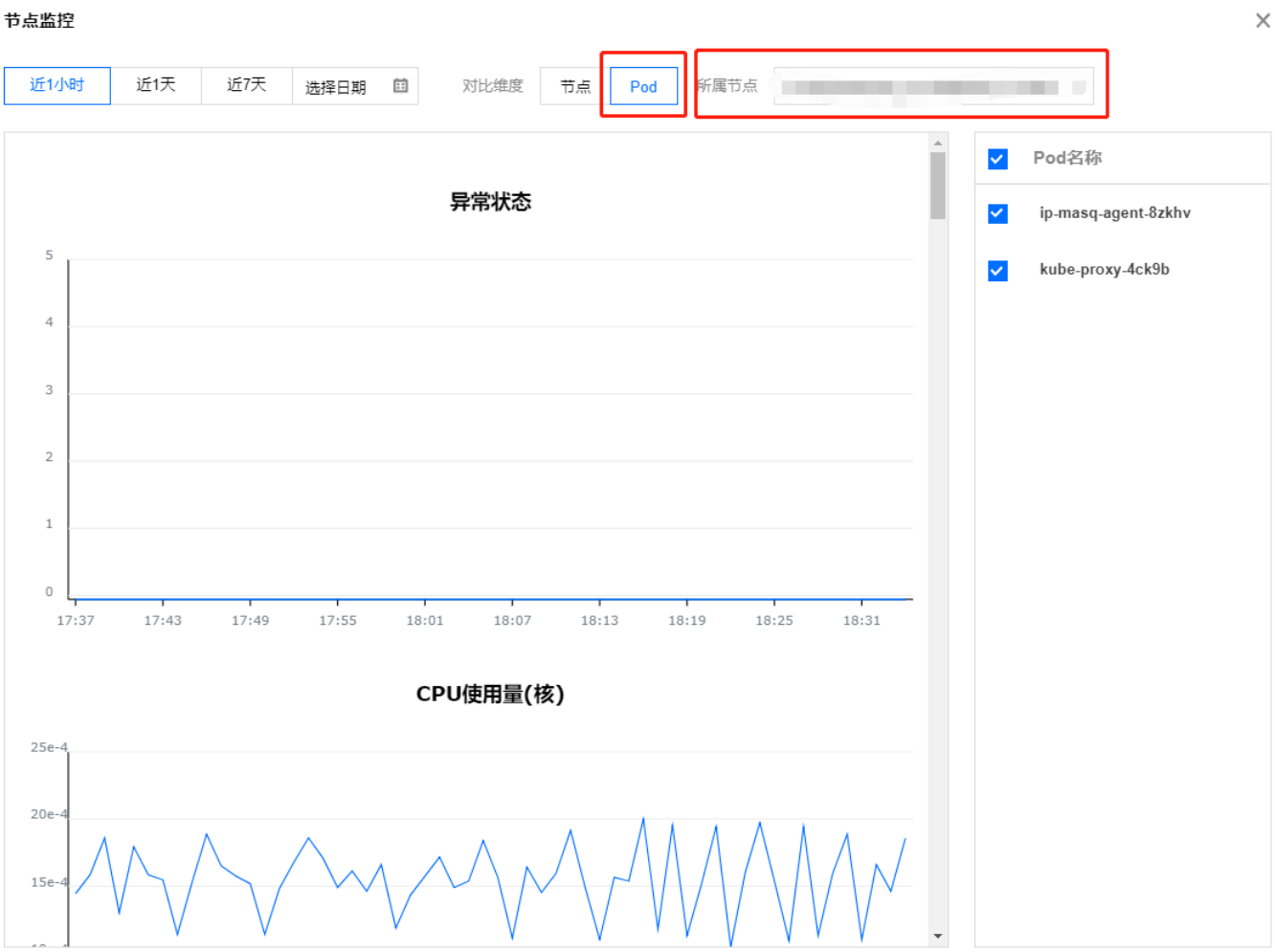
- 选择**Master&Etcd > 监控**，即可进入**Master&Etcd 监控**页面，查看监控信息。如下图所示：

查看节点内 Pod 指标

您可通过以下两种方式查看节点内 Pod 指标。

1. 选择需要查看的集群ID/名称，进入该集群的管理页面。
2. 选择**节点管理 > 节点**，进入节点列表页面。
3. 根据实际需求，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 的监控指标对比图。如下图所示：

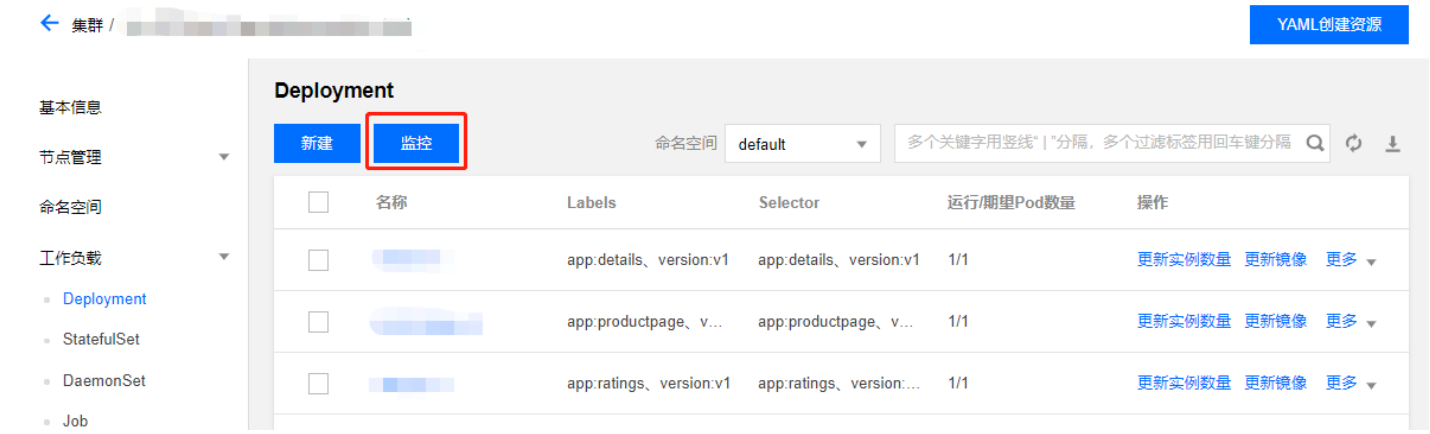


- 在节点详情页面中查看 Pod 指标。
  a. 选择需要查看的节点ID/节点名，进入该节点的管理页面。
  b. 选择**Pod 管理**页签，单击**监控**，即可查看到该节点内 Pod 的监控指标对比图。如下图所示：

查看工作负载指标

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

3. 单击**监控**，即可查看该工作负载的监控信息。如下图所示：



查看工作负载内 Pod 指标

1. 选择需要查看的集群ID/名称，进入该集群的管理页面。
2. 选择**工作负载 > 任意类型工作负载**。例如，选择**Deployment**，进入 Deployment 管理页面。
3. 选择需要查看的工作负载名称，进入该工作负载的管理页面。
4. 选择**Pod 管理**页签，单击**监控**，即可查看该工作负载内所有 Pod 的监控指标对比图。如下图所示：

查看 Pod 内 Container 指标

1. 参照查看工作负载内 Pod 指标的步骤1–步骤3，进入工作负载详情页。
2. 选择**Pod 管理**页签，单击**监控**，进入**Pod 监控**页面。

3. 单击**Container**，将**所属 Pod**选择为您想查看的 Pod，即可查看该 Pod 内 Container 的监控指标对比图。如下图所示：

监控及告警指标列表

最近更新时间：2022-04-22 17:19:35

目前容器服务提供了以下维度的监控告警指标，所有指标均为统计周期内的平均值。

集群监控及告警指标

指标	单位	说明
Pod 数量	个	集群中 Pod 个数
Node 数量	个	集群中 Node 个数
CPU 总配置	核	集群的 CPU 总配置量
CPU 使用量	核	集群的 CPU 使用量
CPU 利用率	%	集群的 CPU 利用率
CPU 使用量（弹性容器）	核	弹性容器的 CPU 使用量（若使用节点池的虚拟节点）
块设备读取大小	Mbytes	集群硬盘的使用总量
块设备读取次数	次	集群硬盘读取总次数
块设备写入大小	Mbytes	集群硬盘写入数据量
块设备写入次数	次	集群硬盘写入总次数
内存总和	Gbytes	集群内存总量
内存使用量	Mbytes	集群内存使用量总和
内存利用率	%	集群内存利用率
内存使用量（弹性容器）	Mbytes	弹性容器的内存使用量（若使用节点池的虚拟节点）
内存使用量（弹性容器，不含 Cache）	Mbytes	弹性容器的内存（不含 Cache）使用量（若使用节点池虚拟节点）
网络入流量	Mbytes	集群网络入流量
网络带宽	Mbps	集群网络带宽
网络入包量	个/s	集群网络入包量
网络出流量	Mbytes	集群网络出流量
网络出包量	个/s	集群网络出包量
GPU 内存总量	Gbytes	集群 GPU 内存总量
GPU 内存使用量	Mbytes	集群 GPU 内存总使用量
GPU 总量	卡	集群 GPU 总量
GPU 使用量	卡	集群整体的 CPU 利用率
显存利用率	%	GPU 显存利用率
GPU 利用率	%	集群 GPU 利用率

Master&Etcd 和普通节点监控及告警指标

指标	单位	说明
Pod 重启次数	次	节点内所有 Pod 的重启次数之和

指标	单位	说明
Node 状态	-	节点的状态，正常或异常
CPU 利用率	%	节点内所有 Pod 的 CPU 使用量占节点总量之比
CPU 分配量	核	节点内所有 Pod 的 CPU 分配量总和
内存利用率	%	节点内所有 Pod 的工作集内存使用量占节点总量之比
内存分配量	Mbps	节点内所有 Pod 的内存分配量总和
内网入带宽	Mbps	节点内所有 Pod 的内网入方向带宽之和
内网出带宽	Mbps	节点内所有 Pod 的内网出方向带宽之和
外网入带宽	Mbps	节点内所有 Pod 的外网入方向带宽之和
外网出带宽	Mbps	节点内所有 Pod 的外网出方向带宽之和
TCP 连接数	个	节点保持的 TCP 连接数
GPU 使用量	卡	节点内所有 Pod 的 GPU 使用量之和
GPU 内存使用量	Mbps	节点内所有 Pod 的 GPU 内存使用量之和
GPU 内存利用率	%	节点内所有 Pod 的 GPU 内存使用量占节点 GPU 内存总量之比
GPU 利用率	%	节点内所有 Pod 的 GPU 使用量占节点 GPU 总量之比
Node 的 eni- IP 分配量	个	Node 的弹性网卡上已分配的 IP 数量
Node 的 direct-eni 分配量	个	Node 的 direct-eni 上已分配的 IP 数量
GlobalRouter模式集群中节点 Pod CIDR 已经分配的 IP 数	个	GlobalRouter 模式的 K8S 集群中，一个节点的 Pod CIDR 中已分配的 IP 个数
GlobalRouter模式集群中节点可以分配的 IP 数	个	GlobalRouter 模式的 K8S 集群中，一个节点中总共可分配的 IP 个数

集群节点更详细的指标监控及告警请参考 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监控及告警请参考 云硬盘监控 和 云监控创建告警策略。

工作负载监控及告警指标

指标	单位	说明
工作负载异常	-	工作负载是否为异常状态，非0即为异常
Pod 数量	个	工作负载内所有 Pod 的数量和
Pod 重启次数	次	工作负载内所有 Pod 的重启次数之和
CPU 使用量	核	工作负载内所有 Pod 的 CPU 使用量
CPU 利用率	%	工作负载内所有 Pod 的 CPU 使用量占总量之比
内存使用量	Mbytes	工作负载内所有 Pod 的内存使用量之和
内存使用量（不含 Cache）	Mbytes	工作负载内所有 Pod 的内存使用量（不含 Cache）之和
内存使用量（working_set）	Mbytes	工作负载内所有 Pod 的工作集内存使用量之和
内存利用率	%	工作负载内所有 Pod 的内存使用量占总量之比
内存利用率（不含 Cache）	%	工作负载内所有 Pod 的内存使用量（不含 Cache）占所有 Pod 内存总量之比
内存利用率（working_set）	%	工作负载内所有 Pod 的工作集内存使用量占总量之比

指标	单位	说明
网络入带宽	bps	工作负载内所有 Pod 的入方向带宽之和
网络出带宽	bps	工作负载内所有 Pod 的出方向带宽之和
网络入流量	B	工作负载内所有 Pod 的入方向流量之和
网络出流量	B	工作负载内所有 Pod 的出方向流量之和
网络入包量	个/s	工作负载内所有 Pod 的入方向包数之和
网络出包量	个/s	工作负载内所有 Pod 的出方向包数之和
块设备读取大小	Mbytes	工作负载内所有 Pod 的块设备读取大小之和
块设备读取次数	次	工作负载内所有 Pod 的块设备读取次数之和
块设备写入大小	Mbytes	工作负载内所有 Pod 的块设备写入大小之和
块设备写入次数	次	工作负载内所有 Pod 的块设备写入次数之和
GPU 使用量	卡	工作负载内所有 Pod 的 GPU 使用量之和
GPU 内存使用量	Mbps	工作负载内所有 Pod 的 GPU 内存使用量之和
GPU 内存利用率	%	工作负载内所有 Pod 的 GPU 内存使用量与 GPU 内存总量之比
GPU 利用率	%	工作负载内所有 Pod 的 GPU 使用量与 GPU 总量之比

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 监控及告警指标

指标	单位	说明
Pod 重启次数	次	Pod 的重启次数
异常状态	-	Pod 的状态，正常或异常
CPU 使用量	核	Pod 的 CPU 使用量
CPU 利用率（占节点）	%	Pod 的 CPU 使用量占节点总量之比
CPU 利用率（占 Request）	%	Pod 的 CPU 使用量和设置的 Request 值之比
CPU 利用率（占 Limit）	%	Pod 的 CPU 使用量和设置的 Limit 值之比
内存使用量	Mbytes	Pod 中 Container 的内存使用量（含缓存）之和（来源：container_memory_usage_bytes）
内存使用量（不包含 Cache）	Mbytes	Pod 中 Container 的内存使用量（不含缓存）之和（来源：container_memory_usage_bytes - container_memory_cache）
内存使用量（working_set）	Mbytes	Pod 中 Container 的工作集内存使用量（来源：container_memory_working_set_bytes）
内存利用率（占节点）	%	Pod 中 Container 的内存使用量（含缓存）占节点总量之比
内存利用率（占节点，不包含 Cache）	%	Pod 中 Container 的内存使用量（不含缓存）占节点总量之比
内存利用率（占节点，working_set）	%	Pod 中 Container 的工作集内存使用量占节点总量之比
内存利用率（占 Request）	%	Pod 中 Container 的内存使用量和设置的 Request 值之比
内存利用率（占 Request，不包含 Cache）	%	Pod 中 Container 的内存使用量（不含缓存）和设置的 Request 值之比

指标	单位	说明
内存利用率（占 Request，working_set）	%	Pod 中 Container 的工作集内存使用量与设置的 Request 值之比
内存利用率（占 Limit）	%	Pod 中 Container 的内存使用量和设置的 Limit 值之比
内存利用率（占 Limit，不包含 Cache）	%	Pod 中 Container 的内存使用量（不含缓存）和设置的 Limit 值之比
内存利用率（占 Limit，working_set）	%	Pod 中 Container 的工作集内存使用量与设置的 Limit 值之比
网络入带宽	Mbps	Pod 的入方向带宽之和
网络出带宽	Mbps	Pod 的出方向带宽之和
网络入流量	Mbytes	Pod 的入方向流量之和
网络出流量	Mbytes	Pod 的出方向流量之和
网络入包量	个/s	Pod 的入方向包数之和
网络出包量	个/s	Pod 的出方向包数之和
Pod TCP 连接数	个	Pod 的 TCP 连接数
块设备读取大小	Mbytes	Pod 的块设备读取大小
块设备读取次数	次	Pod 的块设备读取次数
块设备写入大小	Mbytes	Pod 的块设备写入大小
块设备写入次数	次	Pod 的块设备写入次数
rootfs使用量	字节	Pod 中 rootfs 使用量
GPU 申请量	卡	Pod 中 GPU 申请量
GPU 内存利用率（占节点）	%	Pod 中 GPU 内存使用量占节点 GPU 内存总量之比
GPU 内存利用率（占 request）	%	Pod 中 GPU 内存使用量占 GPU 内存申请量之比
GPU 利用率（占节点）	%	Pod 中 GPU 使用量占节点 GPU 总量之比
GPU 利用率（占 request）	%	Pod 中 GPU 使用量占 GPU 申请量之比
GPU 内存申请量	Mbytes	Pod 中 GPU 内存申请量
GPU 内存使用量	Mbytes	Pod 中 GPU 内存使用量
GPU 使用量	卡	Pod 中 GPU 使用量
GPU 显存使用率	%	Pod 中 GPU 显存使用量占显存总量的百分比
GPU 编码资源使用率	%	Pod 中 GPU 编码资源使用率
GPU 解码资源使用率	%	Pod 中 GPU 解码资源使用率
GPU 流处理器使用率	%	Pod 中 GPU 流处理器使用率

Container 监控及告警指标

指标	单位	说明
CPU 使用量	核	Container 的 CPU 使用量
CPU 利用率（占节点）	%	Container 的 CPU 使用量占节点总量之比

指标	单位	说明
CPU 利用率（占 Request）	%	Container 的 CPU 使用量和设置的 Request 值之比
CPU 利用率（占 Limit）	%	Container 的 CPU 使用量和设置的 Limit 值之比
内存使用量	Mbytes	Container 的内存使用量，含缓存（来源：container_memory_usage_bytes）
内存使用量（不包含 Cache）	Mbytes	Container 的内存使用量，不含缓存（来源：container_memory_usage_bytes - container_memory_cache）
内存使用量（working_set）	Mbytes	Container 的工作集内存使用量（来源：container_memory_working_set_bytes）
内存利用率（占节点）	%	Container 的内存使用量（含缓存）占节点总量之比
内存利用率（占节点，不包含 Cache）	%	Container 的内存使用量（不含缓存）占节点总量之比
内存利用率（占节点，working_set）	%	Container 的工作集内存使用量占节点总量之比
内存利用率（占 Request）	%	Container 的内存使用量和设置的 Request 值之比
内存利用率（占 Request，不包含 Cache）	%	Container 的内存使用量（不含缓存）和设置的 Request 值之比
内存利用率（占 Request，working_set）	%	Container 的工作集内存使用量与设置的 Request 值之比
内存利用率（占 Limit）	%	Container 的内存使用量和设置的 Limit 值之比
内存利用率（占 Limit，不包含 Cache）	%	Container 的内存使用量（不含缓存）和设置的 Limit 值之比
内存利用率（占 Limit，working_set）	%	Container 的工作集内存使用量与设置的 Limit 值之比
块设备读带宽	B/s	Container 从硬盘读取数据的吞吐量
块设备写带宽	B/s	Container 把数据写入硬盘的吞吐量
块设备读 IOPS	次/s	Container 从硬盘读取数据的 IO 次数
块设备写 IOPS	次/s	Container 把数据写入硬盘的 IO 次数

tke-monitor-agent 组件说明

最近更新时间：2022-04-26 11:10:35

组件介绍

为了提升容器服务基础监控及告警服务的稳定性，腾讯云升级了基础监控服务架构。新版基础监控会在用户集群的 kube-system 命名空间下部署一个 DaemonSet，名称为 tke-monitor-agent，并创建对应的认证授权 K8s 资源对象 ClusterRole、ServiceAccount、ClusterRoleBinding，名称均为 tke-monitor-agent。

组件作用

该组件会采集每个节点上容器、Pod、节点、以及官方组件的监控数据，该数据源用于控制台基础监控指标展示、指标告警和基于基础指标的 HPA 服务。部署该组件，可极大程度改善之前因基础监控运行不稳定导致的监控数据无法正常获取的问题，获得更稳定的监控、告警及 HPA 服务。

组件影响

- 部署该组件不会影响集群服务的正常运行。
- 如果您的**节点资源分配不合理**或者**节点负载过高、节点资源不够**，部署基础监控组件时可能会导致监控组件 DaemonSet tke-monitor-agent 对应的 Pod 处于 **Pending**、**Evicted**、**OOMKilled**、**CrashLoopBackOff** 状态，这属于正常现象。对于 DaemonSet tke-monitor-agent 对应 Pod 出现的意外状态描述如下：
  - **Pending** 状态：表示集群的节点上没有足够的资源进行 Pod 的调度，尝试将 DaemonSet tke-monitor-agent 的资源申请量设置为0，可将 Pod 调度上去（详情见 Pod 处于 pending 状态的排错指南）。
  - **Evicted** 状态：DaemonSet tke-monitor-agent 的 Pod 如果处于此状态，可能是您的节点资源不够或者节点本身负载就已经过高，可通过如下方式去查看具体的原因，并进行排查和解决：
    - 执行 kubectl describe pod -n kube-system <podName>，通过 Message 字段的描述信息来查看具体被驱逐的原因。
    - 执行 kubectl describe pod -n kube-system <podName>，通过 Events 字段描述的信息来查看具体被驱逐的原因。
  - **CrashLoopBackOff** 或者 **OOMKilled** 状态：可以通过 kubectl describe pod -n kube-system <podName> 查看是否为 OOM，如果是，可以通过提升 memory limits 的数值解决，limits 值最多不超过100M，如果设置为100M仍然出现 OOM，请 提交工单 来寻求帮助。
  - **ContainerCreating** 状态：执行命令 kubectl describe pod -n kube-system <pod 名称>，查看 Events 字段。若显示如下内容：Failed to create pod sandbox: rpc error: code = Unknown desc = failed to create a sandbox for pod "<pod 名称 >": Error response from daemon: Failed to set projid for /data/docker/overlay2/xxx-init: no space left on device，则表明容器数据盘已满，清理节点上数据盘后即可恢复。

    > 说明：
    > 如果以上描述未解决您的疑问，请 提交工单 来寻求帮助。

- 监控组件 DaemonSet（名称为 tke-monitor-agent）所管理的每个 Pod 的资源耗费情况和节点上运行的 Pod 数量和容器数量成正相关，下图为压测示例，内存和 CPU 占用量均很小。
  数据规模
  节点上有220个 Pod，每个 Pod 有3个容器。
  资源消耗

内存（峰值）	CPU（峰值）
40MiB 左右	0.01C

- CPU 使用量压测结果如下：

- 内存使用量压测结果如下：

云原生监控
云原生监控概述

最近更新时间：2022-04-27 10:14:08

温馨提示

感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。

TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

产品简介

腾讯云云原生监控服务（Tencent Prometheus Service，TPS）是针对云原生服务场景进行优化的监控和报警解决方案，全面支持开源 Prometheus 的监控能力，为用户提供轻量、稳定、高可用的云原生 Prometheus 监控服务。借助 TPS，您无需自行搭建 Prometheus 监控系统，也无需关心数据存储、数据展示、系统运维等问题，只需简单配置即可享受支持多集群的高性能云原生监控服务。

Prometheus 简介

Prometheus 是一套开源的系统监控报警框架，其彻底颠覆了传统监控系统的测试和告警模型，是一种基于中央化的规则计算、统一分析和告警的新模型。作为云原生计算基金会 Cloud Native Computing Foundation 中受欢迎度仅次于 Kubernetes 的项目，Prometheus 依靠其强劲的单机性能、灵活的 PromSQL、活跃的社区生态，逐渐成为云原生时代最核心的监控组件。

Prometheus 优势

- 支持强大的多维数据模型。
- 内置灵活的查询语言 PromQL。
- 支持全面监控。
- 拥有良好的开放性。
- 支持通过动态服务或静态配置发现采集目标。

开源 Prometheus 不足

- 原生 Prometheus 为单点架构，不提供集群化功能，单机性能瓶颈使其无法作为大规模集群下的监控方案。
- 无法便捷地实现动态的扩缩容和负载均衡。
- 部署使用技术门槛高。

云原生监控与开源 Prometheus 对比

对比项	云原生监控	开源 Prometheus
场景	针对容器云原生场景优化	面向多种场景
量级	超轻量级	内存占用高
稳定性	高于原生	无法保证
可用性	高	低
数据存储能力	无限制	受限于本地磁盘
超大集群监控	支持	不支持
数据可视化	基于 Grafana 提供优秀的可视化能力	原生的 Prometheus UI 可视化能力有限
开源生态	完全兼容	原生支持
使用门槛	低	高

对比项	云原生监控	开源 Prometheus
成本	低	高

产品优势

完全兼容 Prometheus 配置和核心 API，保留 Prometheus 原生特性和优势
支持自定义多维数据模型。
内置灵活的查询语言 PromQL。
支持通过动态服务或静态配置发现采集目标。
兼容核心 PrometheusAPI。

支持超大规模集群的监控
在针对单机 Prometheus 的性能压测中，当 Series 数量超过300万（每个 Label 以及值的长度固定为10个字符）时，Prometheus 的内存增长非常明显，需要20GB及以上的 Memory，因此需要使用较大内存的机器来运行。
腾讯云通过自研的分片技术和对象存储 COS 提供的无上限数据存储服务，支持超大规模集群的监控。

支持在同一实例里进行多集群监控
支持同一监控实例内关联多个集群。

支持模板化管理配置
针对多实例多集群的监控，云原生监控服务支持配置监控模板，用户可以使用模板一键完成对多集群的统一监控。

超轻量、无侵入式的监控
腾讯云云原生监控相较于开源 Prometheus 更加轻量化，开源 Prometheus 需要占用用户16GB – 128GB内存，但云原生监控部署在用户集群内的只有非常轻量的 Agent，监控100个节点的集群约只占用20M内存，且无论集群多大，也不会超过1G的内存占用。
当用户关联集群后，云原生监控会自动在用户的集群内部署 Agent，用户无需安装任何组件即可开始监控业务，超轻量级的 Agent 不会对用户集群内的业务和组件产生任何影响。

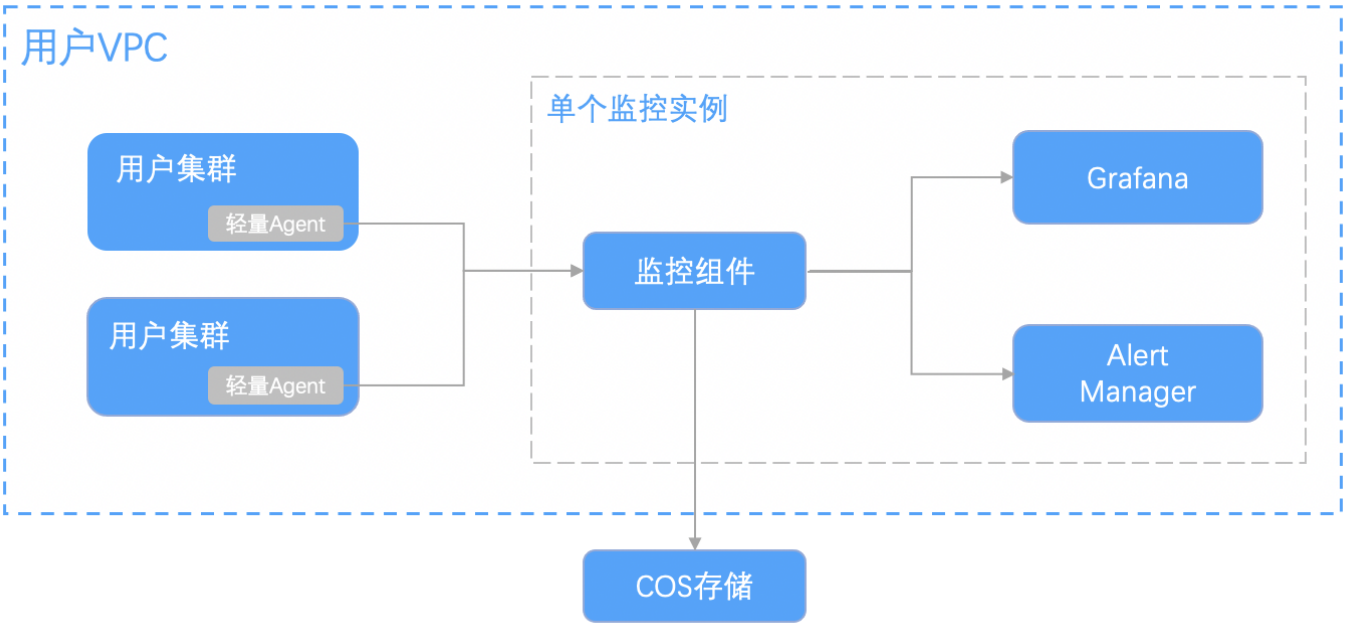
支持实时动态扩缩，满足弹性需求
腾讯云云原生监控采用腾讯云自研的分片和调度技术，可以针对采集任务进行实时的动态扩缩，满足用户的弹性需求，同时支持负载均衡。

高可用性
采用技术手段，保障数据不断点，不缺失，为用户提供高可用的监控服务。

接入成本低
控制台支持产品化的配置文件编写，用户无需精通 Prometheus 即可轻松使用。针对有 Prometheus 实际使用经验的用户，腾讯云也提供原生 YAML 文件提交配置的方式，方便用户自定义高级功能完成个性化监控。

产品架构

腾讯云云原生监控作为超轻量、高可用、无侵入式的监控系统，在用户集群内仅包含一个轻量级的 Agent。其中，位于用户 VPC 内的监控组件负责数据的采集、远端存储、查询等操作。Grafana 负责数据的展示，AlertManager 负责告警。产品架构如下图所示：



云原生监控支持多集群监控、支持同一 VPC 网络中非集群内业务的监控、支持超大集群的监控并实时进行监控组件的扩缩容，保障高可用的监控服务。

在用户关联集群后，云原生监控将默认添加社区主流的采集配置，用户在不做任何个性化配置的基础上能够开箱即用。

此外，云原生监控为每个监控实例内置独立的 Grafana 账户，不仅提供丰富的预设面板，也为用户提供高度自由化的监控自定义能力，用户可完成基于业务的定制化监控而无需关心监控基础资源的管理和调度、监控性能的瓶颈，以最少的成本享受最优质的监控服务。

使用流程

用户需要登录腾讯云账号，进入 云原生监控控制台，在引导下完成腾讯云对象存储 COS 的授权。之后您便可以按照以下流程使用：

1. 创建监控实例。详情可参见 监控实例管理。
2. 关联集群。在新创建的监控实例下完成集群关联操作，此时系统会自动在用户集群内完成 Agent 的部署，在用户的 VPC 内完成监控组件的部署，用户无需进行任何插件的安装。详情可参见 关联集群。
3. 配置采集规则。成功关联集群后用户可以按照实际需求灵活配置数据采集规则，并按需要配置告警规则，配置完成后即可打开 Grafana 查看监控数据。详情可参见 配置采集规则 和 告警配置。

关键概念解释

- **监控实例：**一个监控实例对应一整套监控服务，拥有独立的可视化页面，一个监控实例下可以关联同一 VPC 下的多个集群并完成对多个集群的统一监控。
- **集群：**通常指用户在腾讯云上的 TKE 或 EKS 集群。
- **关联集群：**指将监控实例与用户的集群进行关联的操作。
- **采集规则：**指用户自定义的监控数据采集的规则。

- **Job：**在 Prometheus 中，一个 Job 即为一个采集任务，定义了一个 Job 工作负载下所有监控目标的公共配置，多个 Job 共同组成采集任务的配置文件。
- **Target：**指通过静态配置或者服务发现得到的需要进行数据采集的采集对象。例如，当监控 Pod 时，其 Target 即为 Pod 中的每个 Container。
- **Metric：**用于记录监控指标数据，所有 Metrics 皆为时序数据并以指标名字作区分，即每个指标收集到的样本数据包含至少三个维度（指标名、时刻和指标值）的信息。
- **Series：**一个 Metric+Label 的集合，在监控面板中表现为一条直线。

应用场景

腾讯云云原生 Prometheus 主要针对容器云原生业务场景进行监控，除了实现容器和 Kubernetes 的主流监控方案之外，还灵活支持用户按照自己的业务进行自定义监控，通过逐步完善不同场景的预设面板，不断总结行业最佳实践，来帮助用户完成监控数据的多维分析以及数据的个性化展示，云原生 Prometheus 监控致力于成为容器化场景下的最佳监控解决方案。

产品定价

目前云原生监控服务处于免费公测阶段，使用云原生监控服务时将会在用户的账户下创建 对象存储 COS、云硬盘 CBS 等存储资源，以及内外网 负载均衡 CLB 资源，按用户实际使用的云资源收费。试用 Prometheus 监控服务请前往 云原生监控控制台。

相关服务

云原生 Prometheus 监控负责容器云原生相关的监控业务，若您有其他非容器化场景下的 Prometheus 监控需求，请关注云监控 托管 Prometheus 服务。

TPS 一键迁移到 TMP

最近更新时间：2022-04-29 10:17:52

> 温馨提示
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

TPS 支持一键迁移到 TMP。您可以迁移单独的实例，也可以批量迁移单地域下的实例。每个 TPS 实例的迁移时间一般在十分钟左右。**新的 TMP 实例将以 “旧实例名 (trans-from-prom-xxx)” 命名，其中 “旧实例名” 为原 TPS 的实例名，“xxx” 为原 TPS 实例 ID。**迁移之后，您可以在新的 TMP 实例查看新的监控数据，也可以在旧的实例中查看以前的监控数据，需注意，旧实例将在服务停止时删除。

迁移步骤如下：

1. 迁移 Grafana 配置
2. 创建 Grafana 实例
3. 创建 TMP 实例
4. TMP 绑定 TPS 之前关联的集群
5. 迁移采集配置
6. 迁移告警策略
7. 迁移聚合规则

迁移注意事项

迁移后的预估费用

TPS 和 TMP 已上线“收费指标采集速率”的能力，您可以用该数值估算监控实例/集群/采集对象/指标等多个维度的预估费用：

> 注意：
>
> 仅 TMP 实例会产生费用，TPS 实例不会产生费用。

1. 登录 容器服务控制台 ，选择左侧导航栏中的 **云原生监控**。
2. 在云原生监控列表中，查看“收费指标采集速率”。该指标表示迁移到 TMP 实例的收费指标采集速率，根据用户的指标上报量和采集频率预估算出。该数值乘以 86400 则为一天的监控数据点数，根据 按量计费 可以计算预估的监控数据刊例价。

您也可以单击实例名称右侧的一键迁移，获取该 TPS 实例迁移到 TMP 之后的预估价格。或者在“关联集群”、“数据采集配置”、“指标详情”等多个页面查看到不同维度下的“收费指标采集速率”。

（旧）TPS Prometheus 数据查询地址和 Grafana 地址

如果您有相关的程序平台或系统依赖 TPS 的 Prometheus 数据查询地址和 Grafana 地址。迁移后请及时更换为 TMP 里面相应的地址。否则旧的 TPS 实例在服务停止删除后，您的 Prometheus 数据查询地址和 Grafana 地址 将失效。

1. 登录 容器服务控制台，选择左侧导航栏中的 云原生监控。
2. 单击实例 ID，进入实例的“基本信息”页，如下图所示：

（新）TMP Prometheus 数据查询地址和 Grafana 地址

> 说明：
> TMP 对查询接口增加了鉴权，例如您需要将 TMP 的监控实例对接到您自己的 Grafana 页面，TMP 实例的用户名为您腾讯云账号的 APPID，密码为下图中的 Token。具体可参考 监控数据查询。

1. 登录 容器服务控制台 ，选择左侧导航栏中的 **Prometheus 监控**。
2. 单击实例 ID ，进入实例的“基本信息”页，如下图所示：

> 注意：
> 迁移完成后，请勿在旧的 TPS 实例里面关联新的集群或采集规则，这部分新增的改变将不会自动同步到新的 TMP 实例中。

操作步骤

单实例迁移

1. 登录 容器服务控制台 ，选择左侧导航栏中的 **云原生监控**。
2. 在当前的云原生监控的实例列表页，在上方选择需要迁移的实例所在的地域。
3. 单击实例右方的**一键迁移**。
4. 在弹窗中，选择新的 TMP 实例需要的**网络**和**数据存储时间**。

- **网络**：新的 TMP 实例的 VPC 和子网默认和原来的 TPS 实例一样。若您要选择其它 VPC，请注意该 VPC 首先需要和监控的集群所在的 VPC 网络已打通。
- **数据存储时间**：默认15天，目前仅支持额外选择30天，45天。
- **标签**：非必填字段，根据实际需要选择。
- **预估费用**：如上图所示，迁移的时候会显示当前 TPS 实例，在迁移到 TMP 之后的收费指标采集速率，以及预估的一天费用。

> 注意：
> 具体费用请查看 TMP 涉及 计费方式 和相关 云资源使用情况。若费用过高，建议您 精简监控指标。

5. 单击**确定**。当 TPS 实例状态的括号中内容显示“已迁移”，表示迁移成功。
6. TPS 迁移完成后，您可以在 Prometheus 监控控制台 中选择地域，同地域下中有一个名为**“旧实例名 (trans-from-prom-xxx)” 的 TMP 新实例，其中“旧实例名”为原 TPS 的实例名，“xxx” 为原 TPS 实例 ID。**如下图所示：

> 说明：
> 迁移完成后，请勿在旧的 TPS 实例里面关联新的集群或采集规则，这部分新增的改变将不会自动同步到新的 TMP 实例中。

实例批量迁移

1. 登录 容器服务控制台 ，选择左侧导航栏中的 **云原生监控**。
2. 在当前的云原生监控的实例列表页，在上方选择需要迁移的实例所在的地域。
3. 勾选状态为“未迁移”的实例后，单击上方的“一键迁移”。

> 注意：
> - 批量迁移不支持择新 TMP 实例的 VPC 和子网，如您有类似需求，请进行**单实例迁移**。
> - 迁移前请查看 TMP 涉及 计费方式 和相关 云资源使用情况。若费用过高，建议您 精简监控指标。

4. 单击**确定**。当 TPS 实例状态的括号中内容显示“已迁移”，表示迁移成功。
5. TPS 迁移完成后，您可以在 Prometheus 监控 控制台，在同样的地域里找到一个名为**“旧实例名 (trans-from-prom-xxx)” 的 TMP 新实例，其中“旧实例名”为原 TPS 的实例名，“xxx” 为原 TPS 实例 ID。**

> 说明：
> 迁移完成后，请勿在旧的 TPS 实例里面关联新的集群或采集规则，这部分新增的改变将不会自动同步到新的 TMP 实例中。

监控实例管理

最近更新时间：2022-04-27 10:14:17

> 温馨提示
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

操作场景

您可以在容器产品控制台一键创建 Prometheus 监控实例，创建完成后可以将当前地域中的集群与此实例相关联。关联同一 Prometheus 实例中的集群可以实现监控指标的联查和统一告警。目前云原生监控功能服务支持的集群类型包括托管集群、独立集群、弹性集群以及边缘集群。您可以根据以下指引进行监控实例的创建。本文介绍如何在腾讯云容器服务控制台 中创建和管理监控实例，您可根据以下指引进行监控实例的创建。

操作步骤

服务授权

初次使用云原生监控功能服务需要授权名为 TKE_QCSLinkedRoleInPrometheusService 的服务相关角色，该角色用于授权云原生监控功能服务对 COS 存储桶的访问权限。

1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**，弹出**服务授权**窗口。
2. 单击**前往访问管理**，进入角色管理页面。
3. 单击**同意授权**，完成身份验证后即可成功授权。如下图示：

创建监控实例

1. 登录 容器服务控制台 ，单击左侧导航栏中的**云原生监控**。
2. 进入 Prometheus 监控实例列表页面，单击实例列表上方的**新建**。

3. 在“创建监控实例”页面，设置实例的基本信息。如下图所示：

- **实例名**：输入自定义的监控实例名称，不超过60个字符。
- **地域**：选择您希望部署该实例的地域。当前仅支持部署在北京、上海和广州地域。实例创建后地域无法修改，建议您根据所在地理位置选择靠近业务的地域，可降低访问延迟，提高数据上报速度。
- **网络**：选择当前地域下已有的私有网络和子网。创建后不可修改。若在该地域下没有 VPC 资源可跳转到私有网络控制台新建 VPC，详情请参见 容器及节点网络设置。
- **数据保留时间**：选择数据存储时间，可选30天/3个月/半年/1年。实例创建成功后将自动为您创建对象存储 COS 存储桶并按照实际资源使用情况计费。详情请参见 对象存储计费概述 。
- **Grafana组件**：选择是否开通 Grafana 访问。若选择开通，此处需要设置登录用户名和密码用于 Grafana 登录。实例创建后，Grafana 用户名和密码不可修改。您可以根据业务需要开通 Grafana 外网访问。
- **AlertManger**：您可通过添加□自定义的 AlertManger 地址，将实例产□的告警发往自建的 AlertManger。

> 说明：
> 实例创建成功后，监控对象可以是实例所属 VPC 下的 kubernetes 集群。如需对多地域集群或不同 VPC 下的集群监控，需要在同一 VPC 下新建实例。

4. 单击**完成**，即可完成创建。
5. 您可在“云原生监控”列表页面查看实例创建进度。当实例状态为“运行中”时，表示当前实例已成功创建并处于可用状态。如下图所示：

> 说明：
> 若实例创建花费时间过长，或显示状态为异常，可 在线咨询 联系我们。

删除监控实例

1. 登录 容器服务控制台 ，单击左侧导航栏中的**云原生监控**。
2. 进入 Prometheus 监控实例列表页面，单击期望删除实例右侧的**删除**。
3. 在弹出的“删除监控实例”窗口中，单击**确定**即可删除当前实例。如下图所示：

> **说明：**
> 删除实例时将删除已安装在集群中的监控功能组件，同时默认删除实例关联的 COS 存储桶。如需备份相关监控数据请移步对象存储控制台操作。

关联集群

最近更新时间：2022-04-27 10:14:20

> **温馨提示**
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

操作场景

本文档介绍如何在云原生监控服务中关联集群与监控实例，关联成功后即可编辑数据采集规则等配置。当前服务仅支持与实例所属同一 VPC 下的容器服务 TKE 独立集群、托管集群和弹性集群与监控实例进行关联。

前提条件

- 已登录 容器服务控制台 ，并创建独立集群。
- 已在集群所在 VPC 中 创建监控实例。

操作步骤

关联集群

> **注意：**
>
> 关联集群成功后将在集群中安装监控数据采集插件，该插件在解除关联的同时会被删除。

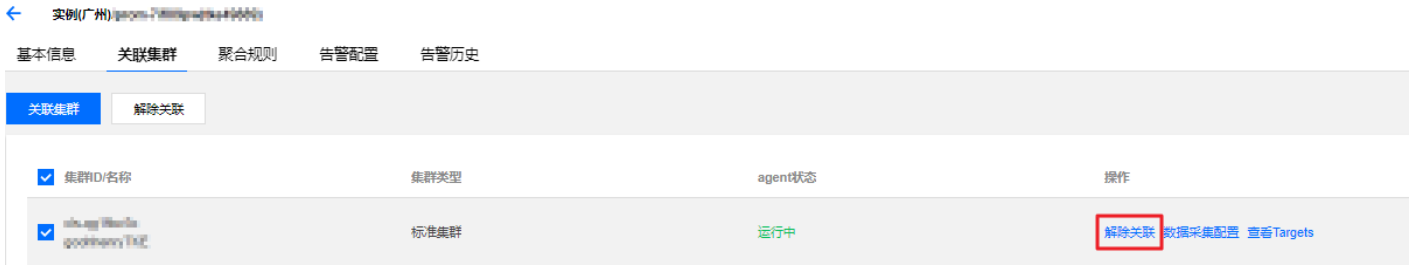
1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**。
2. 在监控实例列表页，选择需要关联集群操作的实例名称，进入该实例详情页。
3. 在“关联集群”页面，单击**关联集群**。如下图所示：

4. 在弹出的“关联集群”窗口，勾选当前 VPC 下需要关联的集群。如下图所示：

5. 单击**确定**即可将所选集群和当前监控实例关联。

解除关联

1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**。
2. 在监控实例列表页，选择解除关联的实例名称，进入该实例详情页。
3. 在“关联集群”页面，单击实例右侧的**解除关联**。如下图所示：



4. 在弹出的“解除关联集群”窗口，单击**确定**即可解除关联。

数据采集配置

最近更新时间：2022-04-27 10:14:24

> **温馨提示**
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

操作场景

本文档介绍如何为已完成关联的集群配置监控采集项。

前提条件

在配置监控数据采集项前，您需要完成以下操作：

- 已成功 创建 Prometheus 监控实例。
- 已将需要监控的集群关联到相应实例中，详情请参见 关联集群。

操作步骤

配置数据采集

1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**。
2. 在监控实例列表页，选择需要配置数据采集规则的实例名称，进入该实例详情页。
3. 在“关联集群”页面，单击实例右侧的**数据采集配置**，进入采集配置列表页。如下图所示：

4. 在“数据采集配置”页中，新增数据采集配置。云原生监控预置了部分采集配置文件，用来采集常规的监控数据。您可以通过以下两种方式配置新的数据采集规则来监控您的业务数据。**通过控制台新增配置**

监控 Service

i. 单击**新增**。

ii. 在“新建采集配置”弹窗中，填写配置信息。如下图所示：

- **监控类型**：选择 “Service监控” 。
- **名称**：填写规则名称。
- **命名空间**：选择 Service 所在的命名空间。
- **Service**：选择需要监控的 Service 名称。
- **ServicePort**：选择相应的 Port 值。
- **MetricsPath**：默认为 /metrics，您可根据需求执行填写采集接口。
- **查看配置文件**：单击“配置文档”可查看当前配置文件。如果您有 relabel 等相关特殊配置的需求，可以在配置文件内进行编辑。
- **探测采集目标**：单击**探测采集目标**，即可显示当前采集配置下能够采集到的所有 target 列表，您可通过此功能确认采集配置是否符合您的预期。

监控工作负载

i. 单击**新增**。

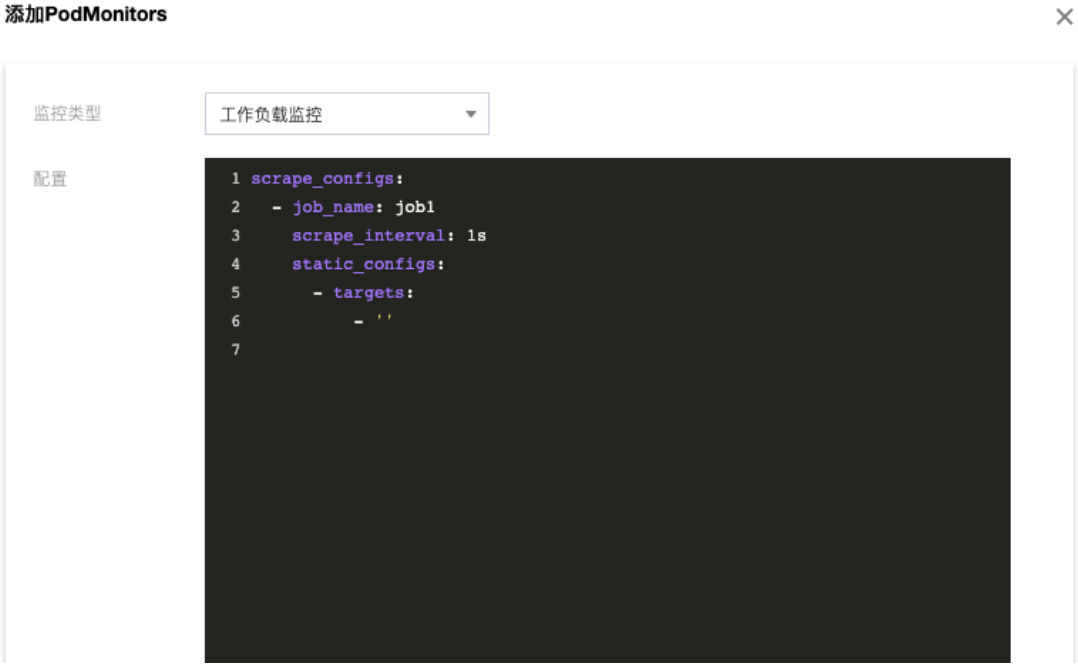
ii. 在“新建采集配置”弹窗中，填写配置信息。如下图所示：

- **监控类型**：选择 “工作负载监控” 。
- **名称**：填写规则名称。
- **命名空间**：选择工作负载所在的命名空间。
- **工作负载类型**：选择需要监控的工作负载类型。
- **工作负载**：选择需要监控的工作负载。
- **targetPort**：填写暴露采集指标的目标端口，通过端口找到采集目标。若端口填写错误将无法获取到正确的采集目标。
- **MetricsPath**：默认为 /metrics，您可根据需求执行填写采集接口。
- **查看配置文件**：单击“配置文档”可查看当前配置文件。如果您有 relabel 等相关特殊配置的需求，可以在配置文件内进行编辑。
- **探测采集目标**：单击**探测采集目标**，即可显示当前采集配置下能够采集到的所有 target 列表，您可通过此功能确认采集配置是否符合您的预期。

通过 yaml 文件新增配置

i. 单击**YAML新增**。

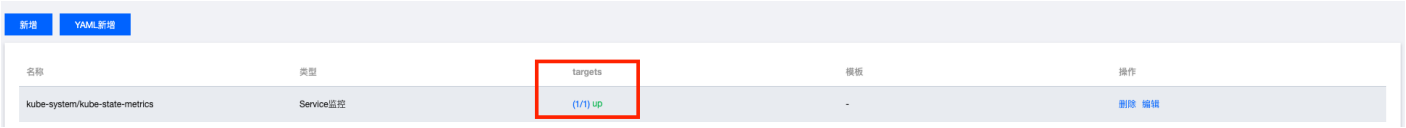
ii. 在弹窗中，选择监控类型，并填写相应配置。本文以 “添加PodMonitors” 为例，如下图所示：



您可以按照社区的使用方式通过提交相应的 yaml 来完成数据采集的配置。

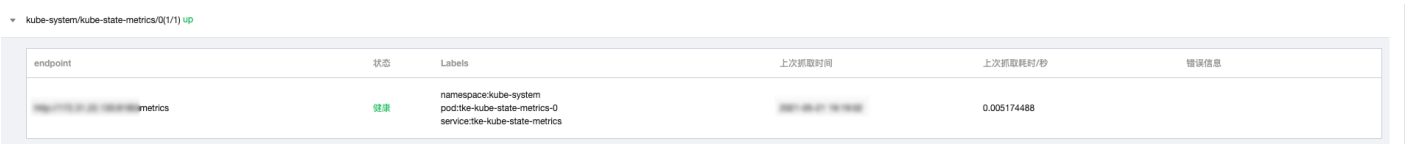
- **工作负载监控**：对应配置为 PodMonitors。
- **service 监控**：对应配置为 ServiceMonitors。
- **RawJobs 监控**：对应配置为 RawJobs。

5. 单击**确认**完成配置。
6. 在该实例的“数据采集配置”页中，查看采集目标状态。如下图所示：



targets（1/1）表示（实际抓取的 targets 数为1 / 探测的采集目标数为1）。当实际抓取数和探测数的数值相等时，显示为 up，即表示当前抓取正常。当实际抓取数小于探测数时，显示为 down，即表示有部分 endpoints 抓取失败。

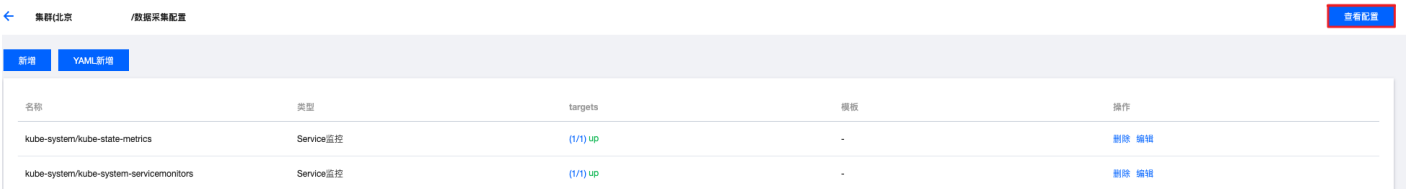
单击字段值（1/1）即可查看采集目标的详细信息。如下图所示：



您还可以在该实例的“关联集群”页中，单击集群名称右侧的**更多 > 查看采集目标**，查看该集群下所有的采集目标情况。如下图所示：

查看已有配置

1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**。
2. 在监控实例列表页，选择右上角的**查看配置**。如下图所示：

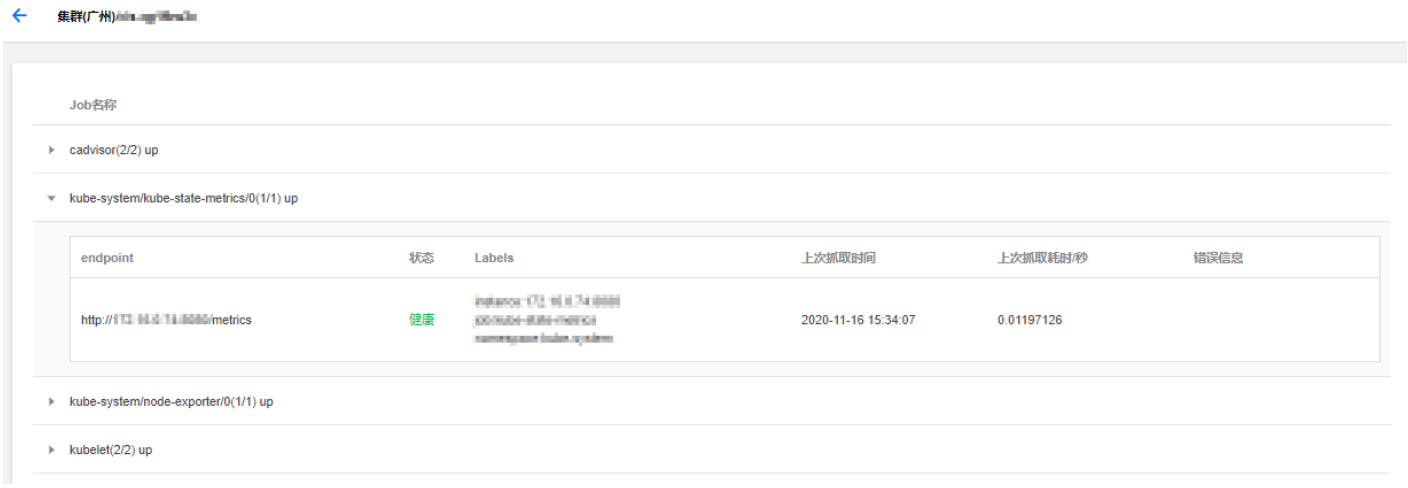


3. 在弹出的“查看配置”窗口，查看 yaml 文件中当前配置的所有监控对象。如下图所示：

查看采集目标

1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**。
2. 在监控实例列表页，选择需要查看 Targets 的实例名称，进入该实例详情页。
3. 在“关联集群”页面，单击实例右侧的**查看采集目标**。

4. 在 Targets 列表页即可查看当前数据拉取状态。如下图所示：



- 状态为“不健康”的 endpoints 默认显示在列表上方，方便及时查看。
- 实例中“采集目标”页面支持检索，可以按资源属性进行过滤。

相关操作

挂载文件到采集器

在配置采集项的时候，如果您需要为配置提供一些文件，例如证书，可以通过以下方式向采集器挂载文件，文件的更新会实时同步到采集器内。

- **prometheus.tke.tencent.cloud.com/scrape-mount = "true"**
  prom-xxx 命名空间下的 configmap 添加如上 label，其中所有的 key 会被挂载到采集器的路径 /etc/prometheus/configmaps/[configmap-name]/ 。
- **prometheus.tke.tencent.cloud.com/scrape-mount = "true"**
  prom-xxx 命名空间下的 secret 添加如上 label，其中所有的 key 会被挂载到采集器的路径 /etc/prometheus/secrets/[secret-name]/ 。

精简监控指标

最近更新时间：2022-04-28 17:53:10

> 温馨提示
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

操作场景

本文档介绍如何精简云原生监控服务 TPS 的**采集指标**，避免在迁移到 TMP 后产生不必要的费用支出。

前提条件

在配置监控数据采集项前，您需要完成以下操作：

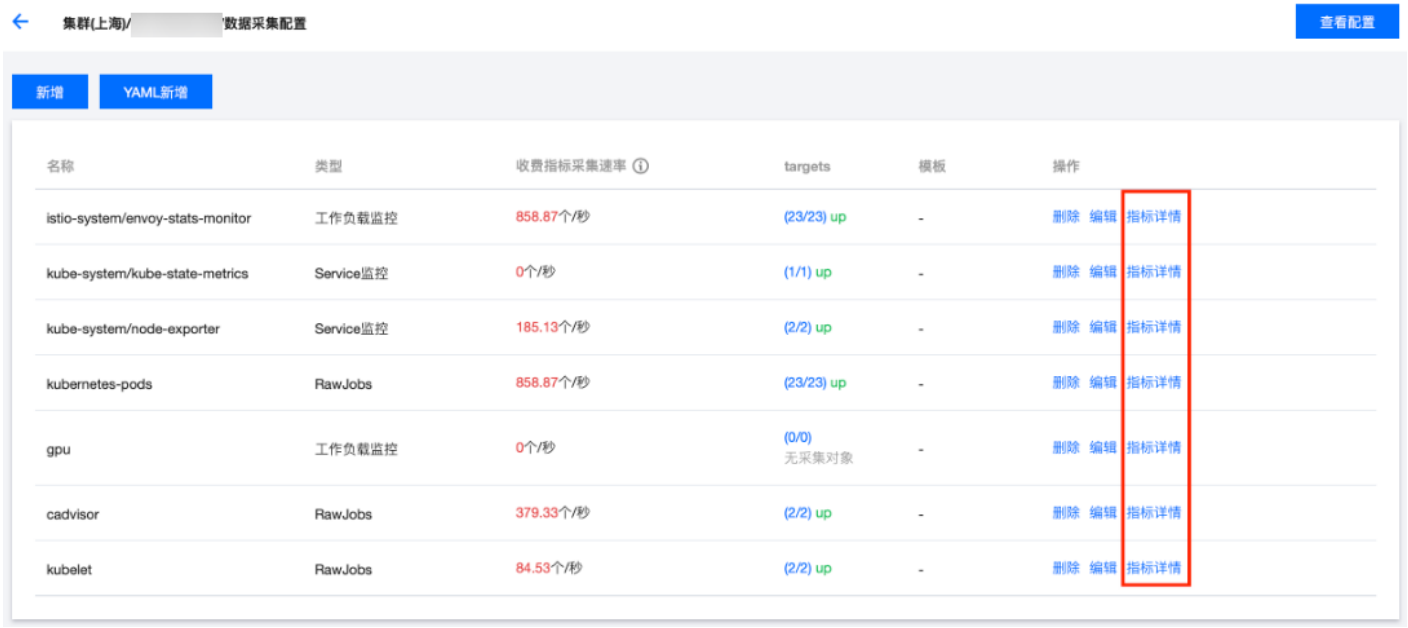
- 已登录 容器服务控制台 ，并创建独立集群。
- 已在集群所在 VPC 中 创建监控实例。

精简指标

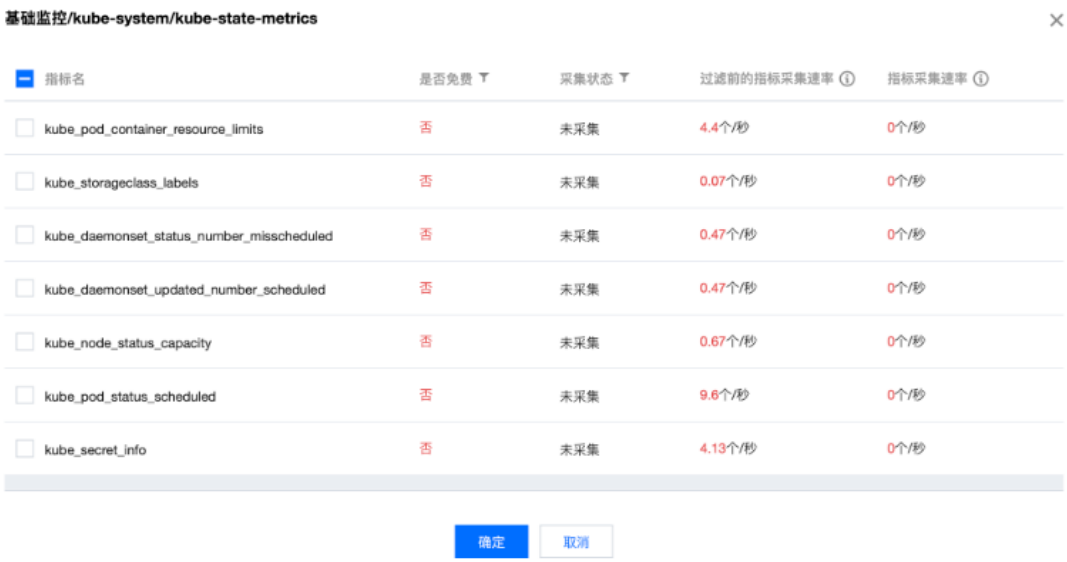
控制台精简指标

Prometheus 监控服务 TMP 提供了一百多个免费的基础监控指标，完整的指标列表可查看 按量付费免费指标。

1. 登录 容器服务控制台 ，选择左侧导航栏中的 **云原生监控**。
2. 在监控实例列表页，选择需要配置数据采集规则的实例名称，进入该实例详情页。
3. 在“关联集群”页面，单击集群右侧的**数据采集配置**，进入采集配置列表页。
4. 基础指标支持通过产品化的页面增加/减少采集对象，单击右侧的“指标详情”：



5. 在以下页面您可以查看到每个指标是否免费，指标勾选表示会采集这些指标，建议您取消勾选付费指标，以免在迁移到 TMP 后造成额外的成本。仅基础监控提供免费的监控指标，完整的免费指标详情见 按量付费免费指标。付费指标计算详情见 Prometheus 监控服务按量计费。



通过 YAML 精简指标

TMP 目前收费模式为按监控数据的点数收费，为了最大程度减少不必要的浪费，建议您针对采集配置进行优化，只采集需要的指标，过滤掉非必要指标，从而减少整体上报量。详细的计费方式和相关云资源的使用请查看 文档。

以下步骤将分别介绍如何在自定义指标的 ServiceMonitor、PodMonitor，以及原生 Job 中加入过滤配置，精简自定义指标。

1. 登录 容器服务控制台 ，选择左侧导航栏中的 **云原生监控**。
2. 在监控实例列表页，选择需要配置数据采集规则的实例名称，进入该实例详情页。
3. 在“关联集群”页面，单击集群右侧的**数据采集配置**，进入采集配置列表页。
4. 单击**编辑**。如下图所示：

ServiceMonitor 和 PodMonitor

ServiceMonitor 和 PodMonitor 的过滤配置字段相同，本文以 ServiceMonitor 为例。
ServiceMonitor 示例：

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
labels:
app.kubernetes.io/name: kube-state-metrics
app.kubernetes.io/version: 1.9.7
name: kube-state-metrics
namespace: kube-system
spec:
endpoints:
- bearerTokenSecret:
key: ""
interval: 15s # 该参数为采集频率，您可以调大以降低数据存储费用，例如不重要的指标可以改为 300s，可以降低20倍的监控数据采集量
port: http-metrics
scrapeTimeout: 15s
jobLabel: app.kubernetes.io/name
namespaceSelector: {}
selector:
matchLabels:
app.kubernetes.io/name: kube-state-metrics
```

若要采集 kube_node_info 和 kube_node_role 的指标，则需要在 ServiceMonitor 的 endpoints 列表中，加入 metricRelabelings 字段配置。注意：是 **metricRelabelings** 而不是 relabelings 。
添加 metricRelabelings 示例：

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
labels:
app.kubernetes.io/name: kube-state-metrics
app.kubernetes.io/version: 1.9.7
name: kube-state-metrics
namespace: kube-system
spec:
endpoints:
- bearerTokenSecret:
key: ""
interval: 15s # 该参数为采集频率，您可以调大以降低数据存储费用，例如不重要的指标可以改为 300s，可以降低20倍的监控数据采集量
port: http-metrics
scrapeTimeout: 15s # 该参数为采集超时时间，Prometheus 的配置要求采集超时时间不能超过采集间隔，即：scrapeTimeout <= interval
# 加了如下四行：
metricRelabelings: # 针对每个采集到的点都会做如下处理
- sourceLabels: ["__name__"] # 要检测的 label 名称，__name__ 表示指标名称，也可以是任意这个点所带的 label
regex: kube_node_info|kube_node_role # 上述 label 是否满足这个正则，在这里，我们希望__name__满足 kube_node_info 或 kube_node_role
action: keep # 如果点满足上述条件，则保留，否则就自动抛弃
jobLabel: app.kubernetes.io/name
namespaceSelector: {}
selector:
```

原生 Job

如果使用的是 Prometheus 原生的 Job，则可以参考以下方式进行指标过滤。
Job 示例：

```
scrape_configs:
- job_name: job1
scrape_interval: 15s # 该参数为采集频率，您可以调大以降低数据存储费用，例如不重要的指标可以改为 300s，可以降低20倍的监控数据采集量
static_configs:
- targets:
- '1.1.1.1'
```

若只需采集 kube_node_info 和 kube_node_role 的指标，则需要加入 metric_relabel_configs 配置。注意：是 **metric_relabel_configs** 而不是 relabel_configs 。
添加 metric_relabel_configs 示例：

```
scrape_configs:
- job_name: job1
scrape_interval: 15s # 该参数为采集频率，您可以调大以降低数据存储费用，例如不重要的指标可以改为 300s，可以降低20倍的监控数据采集量
static_configs:
- targets:
- '1.1.1.1'
# 加了如下四行：
metric_relabel_configs: # 针对每个采集到的点都会做如下处理
- source_labels: ["__name__"] # 要检测的 label 名称，__name__ 表示指标名称，也可以是任意这个点所带的 label
```

```
regex: kube_node_info|kube_node_role # 上述 label 是否满足这个正则，在这里，我们希望__name__满足 kube_node_info 或 kube_node_role
action: keep # 如果点满足上述条件，则保留，否则就自动抛弃
```

屏蔽部分采集对象

屏蔽整个命名空间的监控

TPS 关联集群后，默认会纳管集群中所有 ServiceMonitor 和 PodMonitor，若您想屏蔽某个命名空间下的监控，可以为指定命名空间添加 label： tps-skip-monitor: "true" ，关于 label 的操作请 参考。

屏蔽部分采集对象

TPS 通过在用户的集群里面创建 ServiceMonitor 和 PodMonitor 类型的 CRD 资源进行监控数据的采集，若您想屏蔽指定 ServiceMonitor 和 PodMonitor 的采集，可以为这些 CRD 资源添加 labe： tps-skip-monitor: "true" ，关于 label 的操作请 参考。

创建聚合规则

最近更新时间：2022-04-27 10:14:32

> 温馨提示
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

操作场景

本文档介绍应对复杂查询场景时如何配置聚合规则，提高查询的效率。

前提条件

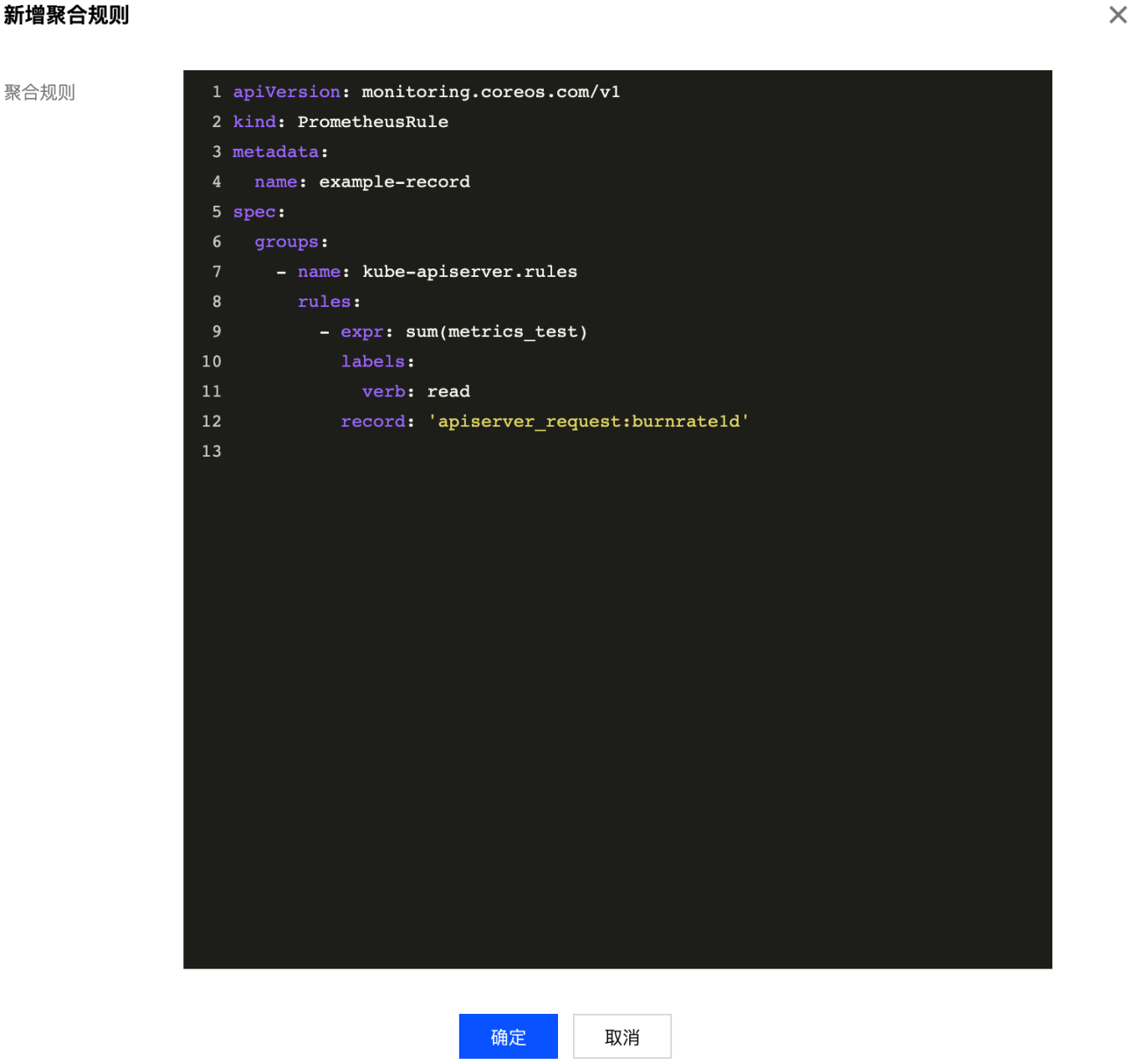
在配置聚合规则前，您需要完成以下操作：

- 已登录 容器服务控制台 ，并创建独立集群。
- 已在集群所在 VPC 中 创建监控实例。

操作步骤

1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**。
2. 在监控实例列表页，选择需要创建聚合规则的实例名称，进入该实例详情页。
3. 在“聚合规则”页面，单击**新建聚合规则**。如下图所示：

4. 在弹出的“新增聚合规则”窗口，编辑聚合规则。如下图所示：



5. 单击**确定**，即可完成创建聚合规则。

告警配置

最近更新时间：2022-04-27 10:14:36

> 温馨提示
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

操作场景

本文档介绍如何在云原生监控服务中配置告警规则。

前提条件

在配置告警前，您需要完成以下准备工作：

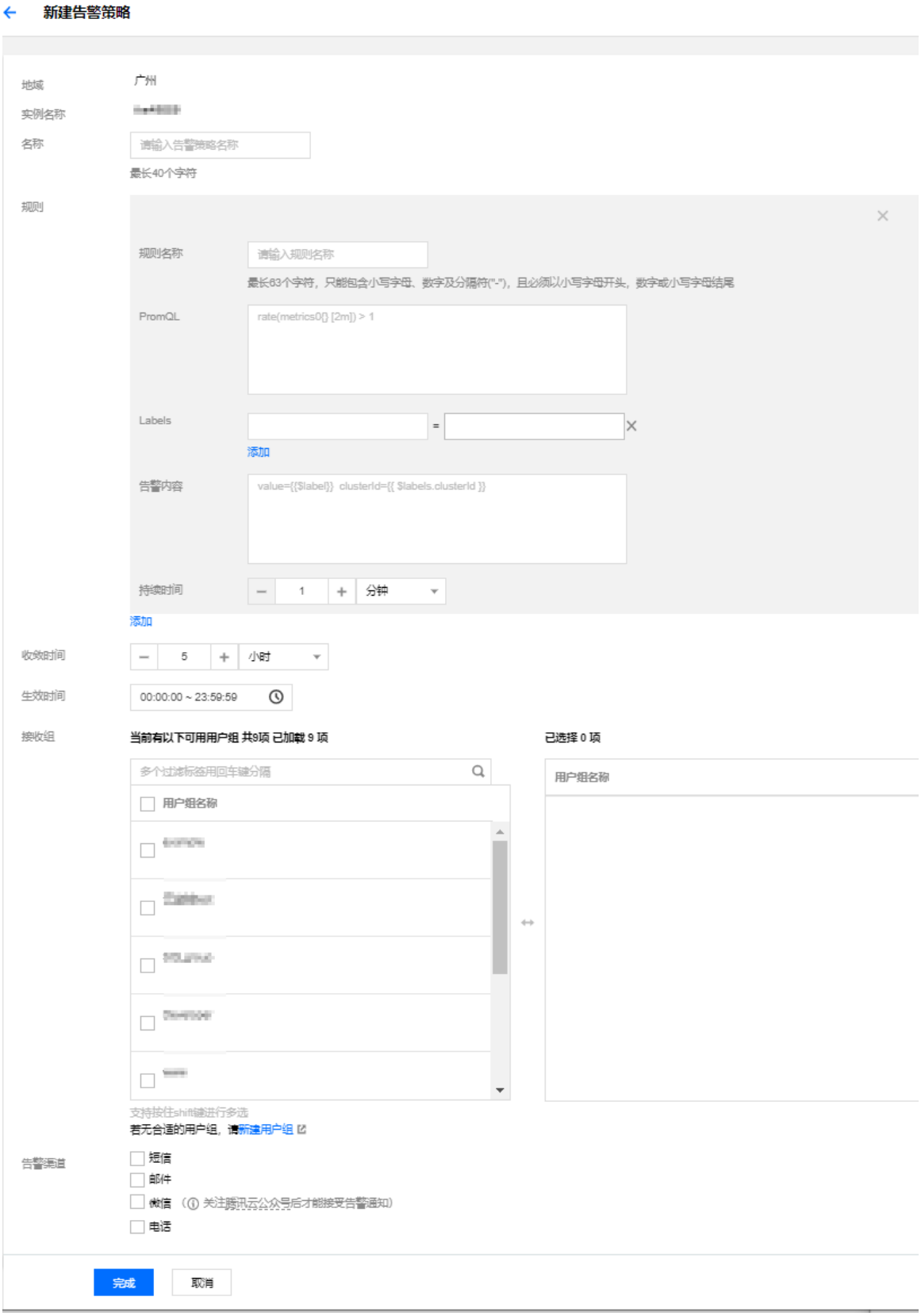
- 已成功 创建 Prometheus 监控实例。
- 已将需要监控的集群关联到相应实例中，详情请参见 关联集群。
- 已将需要采集的信息添加到集群 数据采集配置。

操作步骤

配置告警规则

1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**。
2. 在监控实例列表页，选择需要配置告警规则的实例名称，进入该实例详情页。
3. 在“告警配置”页面，单击**新建告警策略**。如下图所示：

4. 在“新建告警策略”页面，添加策略详细信息。如下图所示：



- **规则名称**：告警规则的名称，不超过40个字符。
- **PromQL**：告警规则语句。
- **持续时间**：满足上述语句所描述的条件的时间，达到该持续时间则会触发告警。
- **Label**：对应每条规则添加 Prometheus 标签。
- **告警内容**：告警触发后通过邮件或短信等渠道发送告警通知的具体内容。
- **收敛时间**：在该周期内，若多次满足告警条件，仅会发送一次通知。

- **生效时间**：一天之中可以发送告警通知的时间段。
- **接收组**：接收告警信息的联系人组。
- **告警渠道**：告警后发送告警内容的渠道。

5. 单击**完成**，即可完成新建告警策略。

> ⚠ 注意：
> 新建告警策略后，默认告警策略生效。

暂停告警

1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**。
2. 在监控实例列表页，选择需要暂停告警的实例名称，进入该实例详情页。
3. 在“告警配置”页面，单击实例右侧的**更多** > **暂停告警**。如下图所示：

4. 在弹出的“关闭告警设置”窗口单击**确定**，即可暂停告警策略。

告警历史

最近更新时间：2022-04-27 10:14:40

> 温馨提示
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

操作场景

本文档介绍如何在云原生监控功能服务中查看告警历史。

前提条件

在查看告警历史前，需要完成以下前置操作：

- 已成功 创建 Prometheus 监控实例。
- 已将需要监控的集群关联到相应实例中，详情请参见 关联集群。
- 已将需要采集的信息添加到集群 数据采集配置。
- 已 配置告警规则。

操作步骤

1. 登录 容器服务控制台 ，选择左侧导航栏中的**云原生监控**。
2. 在监控实例列表页，选择需要查看告警历史的实例名称，进入该实例详情页。
3. 在“告警历史”页面，选择时间即可查看告警历史。如下图所示：

云原生监控资源使用情况

最近更新时间：2022-04-27 10:14:45

> **温馨提示**
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

目前云原生监控服务处于免费公测阶段，使用云原生监控服务时将会在用户的账户下创建 对象存储 COS、云硬盘 CBS 等存储资源，以及内外网 负载均衡 CLB 资源，按用户实际使用的云资源收费。本文向您介绍使用云原生监控服务时资源的使用情况。

资源列表

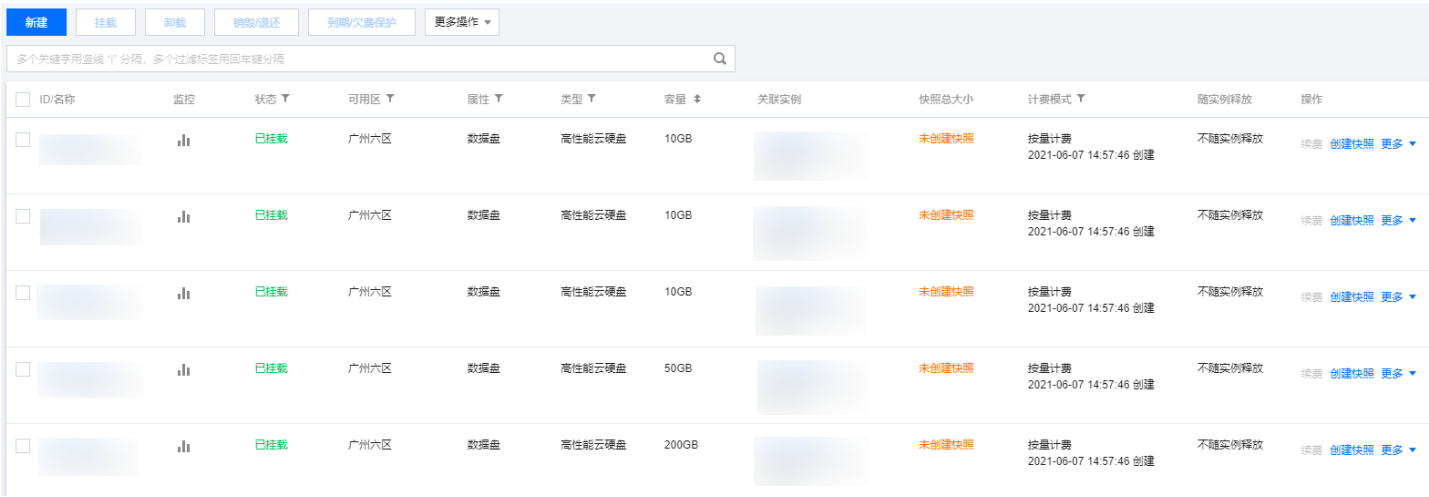
对象存储 COS

创建云原生监控实例后，会在用户的账户下开通对象存储 COS，用于指标数据的持久化存储。在 对象存储控制台 上可查看资源信息，如下图所示：

该资源按实际指标存储量和存储时间（由用户在创建实例时定义）计费，计费详情请参见对象存储 按量计费（后付费） 文档。

云硬盘 CBS

创建云原生监控实例后，会在用户的账户下购买5块高性能云硬盘，用于指标数据的临时存储。在 云硬盘控制台 可查看云硬盘资源和规格信息，如下图所示：



其中：

- 用于 Grafana 的硬盘规格为10G。
- 用于 Thanos Rule 组件的硬盘规格为50G。
- 用于 Thanos Store 组件的硬盘规格为200G。
- 用于 AlertManager 的硬盘规格为10G。
- 用于 Prometheus 的硬盘规格不固定，会按照指标的实际数据增减，约30w series（约30个节点）对应10G规格。

该资源按实际使用量计费，计费详情请参见云硬盘 价格总览 文档。

负载均衡 CLB

创建云原生监控实例后，会在用户的账户下创建2个内网 LB，每多关联一个集群，会增加一个 LB。若要使用通过外网访问 Grafana 服务，则需要创建一个相应的公网 LB，该资源会收取费用，创建的公网 LB 可在 负载均衡控制台 查看资源信息，如下图所示：

该资源按实际使用量计费，计费详情请参见负载均衡 标准账户类型计费说明 文档。

资源销毁

目前不支持用户直接在对应控制台删除资源，需要在 云原生监控控制台 销毁监控实例，对应的所有资源会一并销毁。腾讯云不主动回收用户的监控实例，若您不再使用云原生监控服务，请务必及时删除监控实例，以免发生资源的额外扣费。

关闭云原生监控

最近更新时间：2022-04-27 10:14:49

> 温馨提示
>
> 感谢您对腾讯云原生监控 TPS 的认可与信赖，为提供更优质的服务和更强大的产品能力，TPS 与原腾讯云 Prometheus 监控服务进行融合和升级，升级为 TMP。支持跨地域跨 VPC 监控，支持统一 Grafana 面板对接多监控实例实现统一查看。TMP 计费详情见 按量计费，相关云资源使用详情见 计费方式和资源使用。若您只使用基础监控的 免费指标，TMP 不会收取任何指标费用。
>
> TPS 将于2022年5月16日下线，详情见 公告。TMP 已正式发布，欢迎 了解试用。TPS 已不支持创建新实例，我们提供一键 迁移工具，帮您一键将 TPS 实例迁移到 TMP，迁移前请 精简监控指标 或降低采集频率，否则可能产生较高费用，再次感谢您对 TPS 的支持和信任。

操作场景

当您不需要再使用云原生监控服务监控集群时，可以通过云原生监控控制台删除所有监控实例，系统会自动卸载监控组件并销毁相关资源。

操作步骤

1. 登录**容器服务控制台 *，*选择左侧导航中的 *云原生监控***。
2. 在**监控实例列表**页中找到需要删除的实例，单击实例名称右侧的**删除**。如下图所示：

3. 在“删除监控实例”弹窗中确认监控实例信息后，单击**确定**。

> - 实例删除后，云原生监控控制台不再展示该实例信息。
> - 实例删除后，实例内已有的监控功能组件等资源及配置均将被删除，实例关联的集群将自动解除关联不再被监控，实例关联的 COS 存储桶将随实例一并删除，如需备份相关监控数据可以在对象存储控制台进行操作。
> - 删除操作不可逆，以上实例数据将无法恢复，请谨慎操作。

远程终端
远程终端概述

最近更新时间：2022-01-19 11:27:22

远程终端概述

远程终端帮助您快速调试容器，连接容器查看问题，支持复制粘贴、上传下载文件功能，解决用户登录容器路径长、调试难的问题。

使用帮助

- 远程终端的基本操作
- 其他容器登录方式

远程终端基本操作

最近更新时间：2022-01-17 15:13:39

远程终端连接到容器

1. 登录腾讯云容器服务控制台，选择左侧导航栏中的 **集群**。
2. 在“集群管理”页面，单击集群 ID（cls-xxx），进入集群详情页。如下图所示：

3. 在集群详情页，选择左侧导航栏中的 **节点管理** > **节点**。如下图所示：

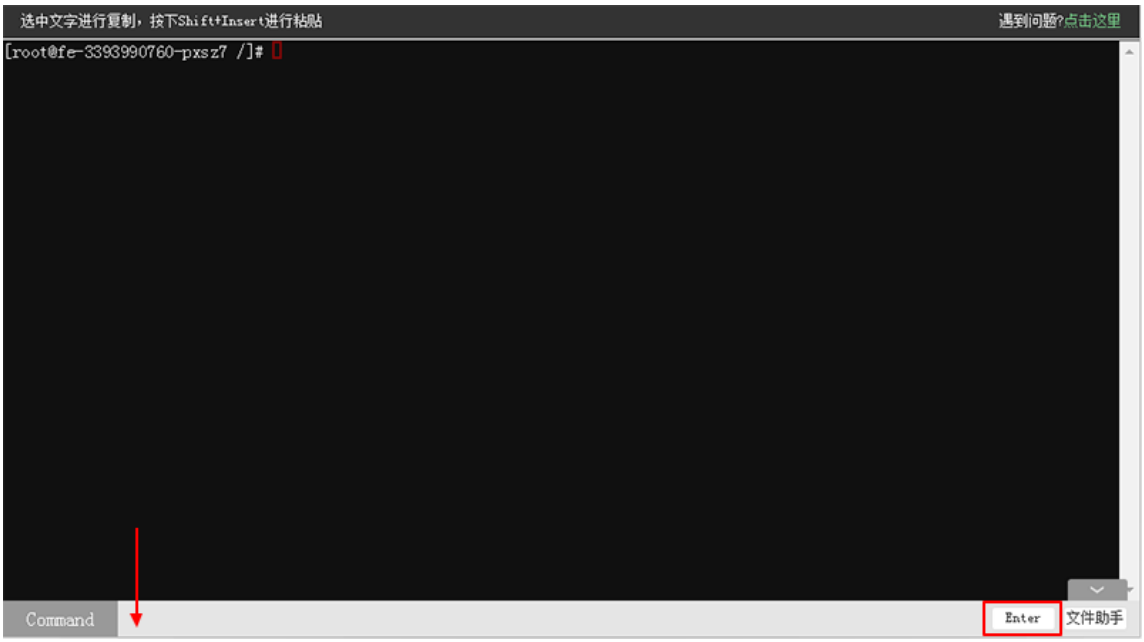
4. 在“节点列表”页面，单击节点 ID，进入 Pod 管理页面。
5. 单击实例所在行右侧的**远程登录**，登录到远程终端。如下图所示：

> **注意：**
>
> **符合以下任一条件的容器均不支持远程登录：**
>
> - 命名空间为 kube-system。
> - 容器镜像中没有内置 bash。

未安装 shell 的容器运行命令

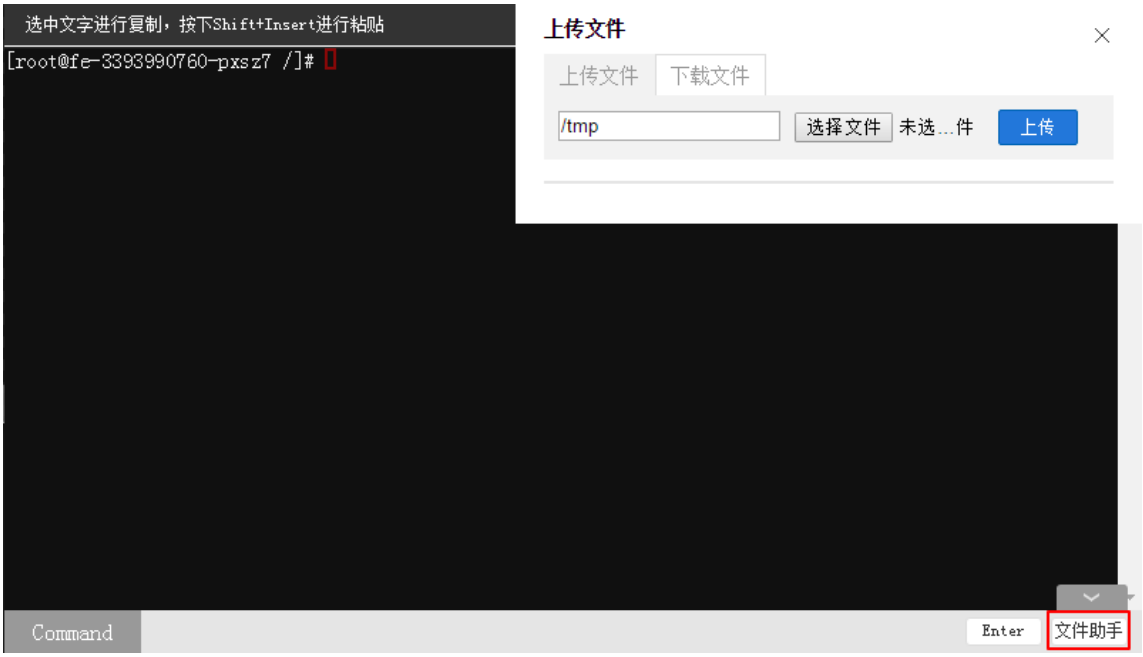
1. 进入远程终端页面。

2. 在下方输入要执行的命令，单击确认。如下图所示：



文件的上传与下载

1. 进入远程终端页面。
2. 单击下方的文件助手，选择上传或下载文件。如下图所示：
   - 上传需指定上传的文件目录。
   - 下载需指定下载的文件的路径。



其他容器登录方式

最近更新时间：2022-01-17 15:14:37

通过 web 终端登录到容器（推荐）

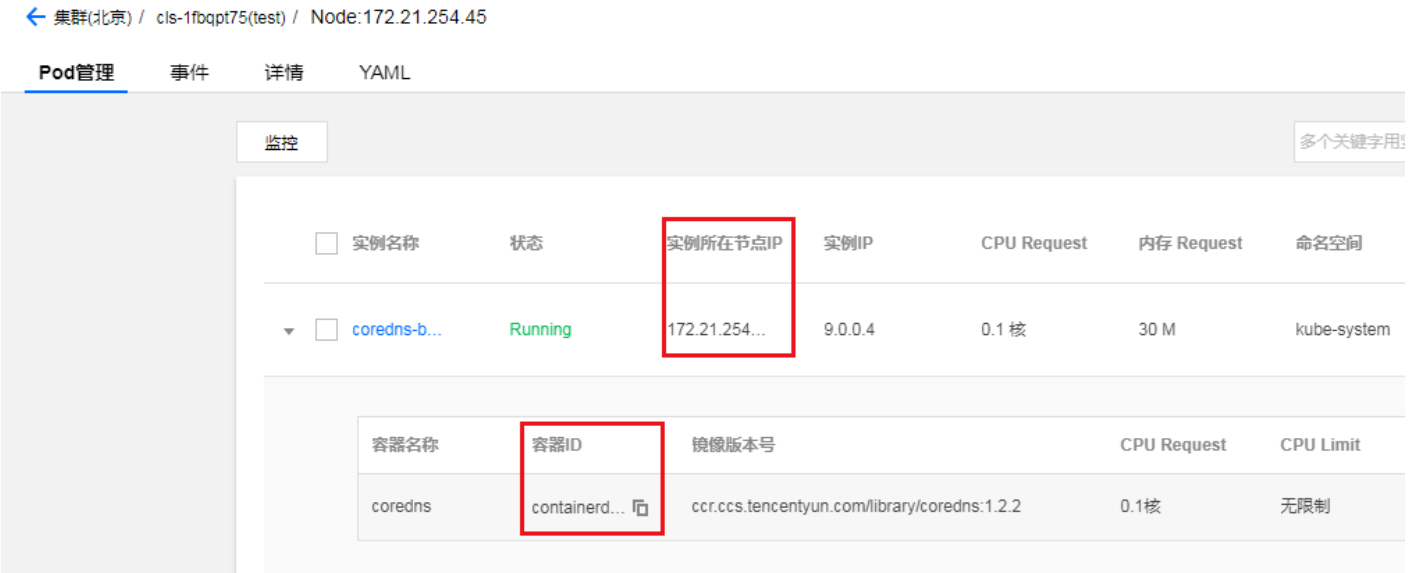
1. 登录容器服务控制台，选择左侧导航栏中的 **集群**。
2. 在“集群管理”页面，单击集群 ID（cls-xxx），进入集群详情页。
3. 在集群详情页，选择左侧导航栏中的 **节点管理** > **节点**。
4. 在“节点列表”页面，选择容器所属节点，单击进入 Pod 管理详情页，查看实例列表，选择容器并登录远程终端，如下图所示。

> **说明：**
> 更多远程终端常见问题单击 查看详情。

> **注意：**
> 符合以下任一条件的容器均不支持远程登录：
>
> - 命名空间为 kube-system。
> - 容器镜像中没有内置 bash。

通过容器所在节点登录到容器

1. 获取容器所在节点 IP 地址，容器 ID。如下图所示：



2. 登录到节点，详情查看 登录到云服务器。
3. 通过 docker ps 命令查看需登录的容器。

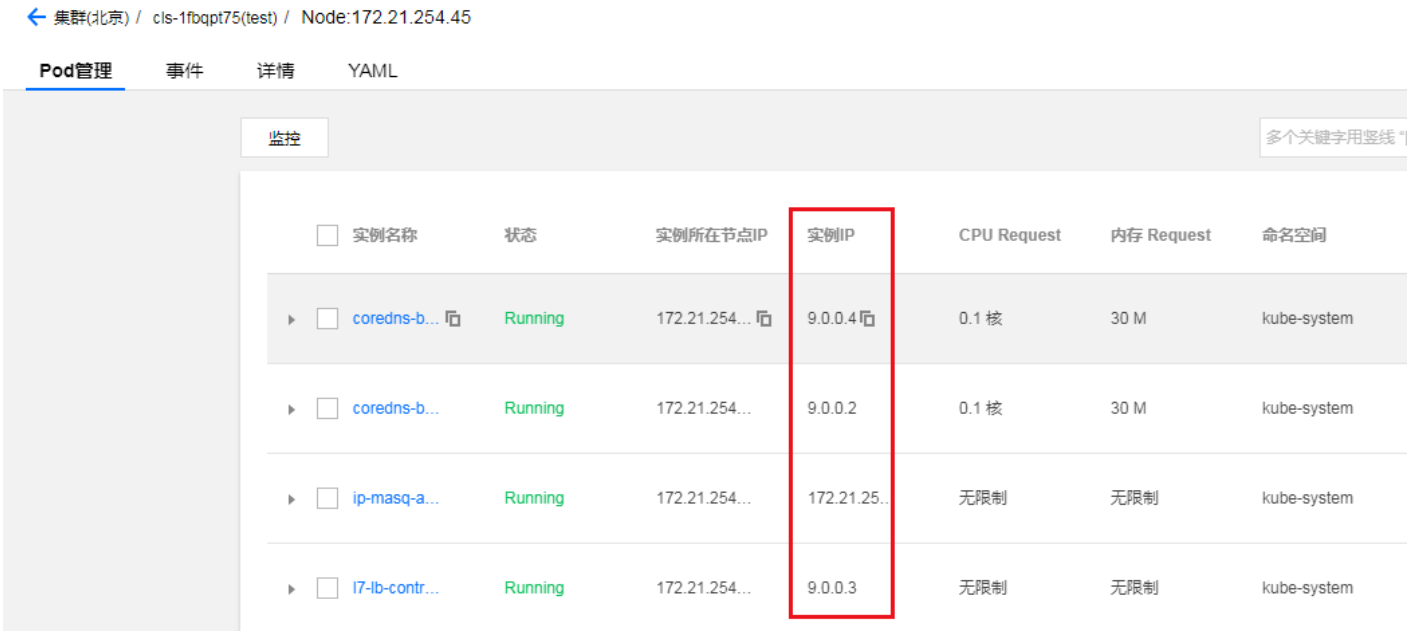
```
[root@VM_88_88_centos ~]# docker ps | grep 75b3b15af61a
75b3b15af61a nginx:latest "nginx -g 'daemon off" About a minute ago Up About a minute k8s_world.e8b44cc_world-24bn2_default_8
1a59654-aa14-11e6-8a18-52540093c40b_42c0b746
```

4. 通过 docker exec 命令登录到容器。

```
[root@VM_0_60_centos ~]# docker ps | grep 75b3b15af61a
75b3b15af61a nginx:latest "nginx -g 'daemon off" 2 minutes ago Up 2 minutes k8s_world.e8b44cc_world-24bn2_default_81a59654-aa
14-11e6-8a18-52540093c40b_6b389dd2
[root@VM_0_60_centos ~]# docker exec -it 75b3b15af61a /bin/bash
root@world-24bn2:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
```

容器已安装 SSH 服务端，通过 SSH 登录容器

1. 获取容器 IP 地址。如下图所示：



2. 登录集群内任意节点，详情查看 登录到云服务器。
3. 通过 SSH 登录到容器。