

容器服务

可观测配置



腾讯云

【版权声明】

©2013–2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。

您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

可观测配置

运维可观测性

容器服务可观测体系概述

日志管理

日志采集概述

采集容器日志到 CLS

使用 CRD 配置日志采集

使用 CRD 进行日志采集配置指导

LogConfig json 格式说明

日志组件版本升级

审计管理

审计日志

审计仪表盘

事件管理

事件日志

事件仪表盘

监控管理

基础监控与告警

监控告警概述

查看监控数据

监控及告警指标列表

控制面组件监控

查看 TKE 集群控制面组件监控

kube-apiserver 组件指标说明

kube-scheduler 组件指标说明

kube-controller-manager 组件指标说明

用户自建 Prometheus 采集控制面监控

应用性能管理

告警管理

健康检查

风险推送

成本洞察和优化

成本洞察

Node Map

Workload Map

成本优化

Request 智能推荐

可观测配置

运维可观测性

容器服务可观测体系概述

最近更新时间：2025-06-04 18:05:02

容器服务可观测架构概述

可观测性（Observability）是指通过系统输出的外部信息（如日志、指标、追踪数据等）来推断系统内部状态的能力。它强调在无需侵入系统或预定义监控规则的情况下，仍能实时理解系统的运行状态、诊断问题并分析未知行为。

Kubernetes 的可观测性体系是一套针对容器化、动态、分布式环境的监控与诊断框架，旨在通过多维度数据（日志、指标）的采集、关联和分析，实现对集群、节点、Pod、容器及应用的全面状态感知。

本文介绍腾讯云容器服务 TKE 可观测能力，以帮助您对容器服务可观测性能力有一个全局的了解。以下分别是指标、日志、链路追踪的可观测性能力介绍。

指标可观测性

指标可观测性是容器化环境监控的核心支柱，它通过采集、存储和分析随时间变化的数值型数据（Metrics），提供对系统运行状态、资源利用率和应用性能的量化洞察。指标能够量化 TKE 集群整体的资源利用率、调度效率、控制平面健康状态以及分布式系统的协同性能，旨在确保集群高效稳定运行。

可观测能力	说明	参考文档
云监控	容器服务 TKE 新建的集群默认安装并集成云监控。可从容器服务控制台上开箱即用的获取集群、容器、Pod 等维度的基础指标。	查看监控数据
腾讯云 Prometheus 监控服务	Prometheus 监控服务（TencentCloud Managed Service for Prometheus，TMP）是基于开源 Prometheus 构建的高可用、全托管的服务，与容器服务（TKE）高度集成，兼容开源生态丰富多样的应用组件，结合腾讯云可观测平台的告警功能和 Prometheus Alertmanager 能力，提供多种开箱即用的预置监控大盘，为您提供免搭建的高效运维能力，减少开发及运维成本。	Prometheus 监控
自建 Prometheus 监控	提供自建 Prometheus 监控 TKE 集群的方案。	自建 Prometheus 监控 TKE 集群

日志可观测性

日志可观测性是容器化环境中记录、分析和诊断离散事件的核心能力。它通过捕获系统、应用和组件生成的文本流，提供上下文丰富的运行证据，是排查复杂问题的“终极武器”。

Kubernetes 事件日志是集群内部的审计日志，记录了从节点、Pod、服务到控制器等核心对象的关键状态变更和异常信息。它提供了理解集群内部运作机制和诊断系统级问题的黄金线索，是 TKE 可观测性的重要组成部分。

可观测能力	说明	参考文档
日志监控	应用日志监控不仅是故障排查的“最后一道防线”，更是理解系统行为、优化性能和保障安全的核心手段。通过腾讯云日志服务提供的一站式日志服务平台，提供了从日志采集、日志存储到日志检索，图表分析、监控告警、日志投递等多项服务，团队可以实现从被动响应到主动预防的转变，为业务稳定性和用户体验提供坚实基础。	采集容器日志到 CLS
事件监控	事件监控是 Kubernetes 从事件角度出发的另一种监控方式，可以弥补资源监控在实时性、准确性和场景上的缺陷。开发者可以通过获取事件，实时诊断集群的异常与问题。推荐使用腾讯云日志服务 CLS（Cloud Log Service）产品实现事件的监控。	事件日志

链路追踪可观测性

链路追踪可观测性旨在通过采集、分析和可视化应用的运行时数据，帮助开发者快速定位性能瓶颈、异常根因，并优化用户体验。

可观测能力	说明	参考文档
腾讯云应用性能监控 APM	腾讯云应用性能监控 APM 是一款应用性能管理平台，基于实时的多语言应用探针全量采集技术，为您提供分布式应用性能分析和故障自检能力，全方位保障系统的可用性和	应用性能管理

稳定性。通过调用链分析、接口分析、数据库调用监控等能力协助您在复杂的业务系统快速定位性能问题，降低 MTTR（平均故障恢复时间）。实时了解并追踪应用性能情况，提升用户体验。

相关文档

- 指标监控，请参见 [基础资源监控](#)、[监控及告警指标列表](#)、[应用性能监控](#)、[查看 TKE 集群控制面组件监控](#)。
- 日志采集，请参见 [日志采集](#)、[事件日志](#)、[审计日志](#)。
- Prometheus 监控，请参见 [Prometheus 监控服务](#)、[腾讯云 Prometheus 一键关联监控容器服务](#)。

日志管理

日志采集概述

最近更新时间：2025-04-17 17:36:02

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。您可参考以下步骤为集群开启日志采集功能。

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。

基本概念

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全文、多行全文、单行 – 完全正则、多行 – 完全正则、JSON、分隔符、组合解析的形式发送至用户指定的日志主题。

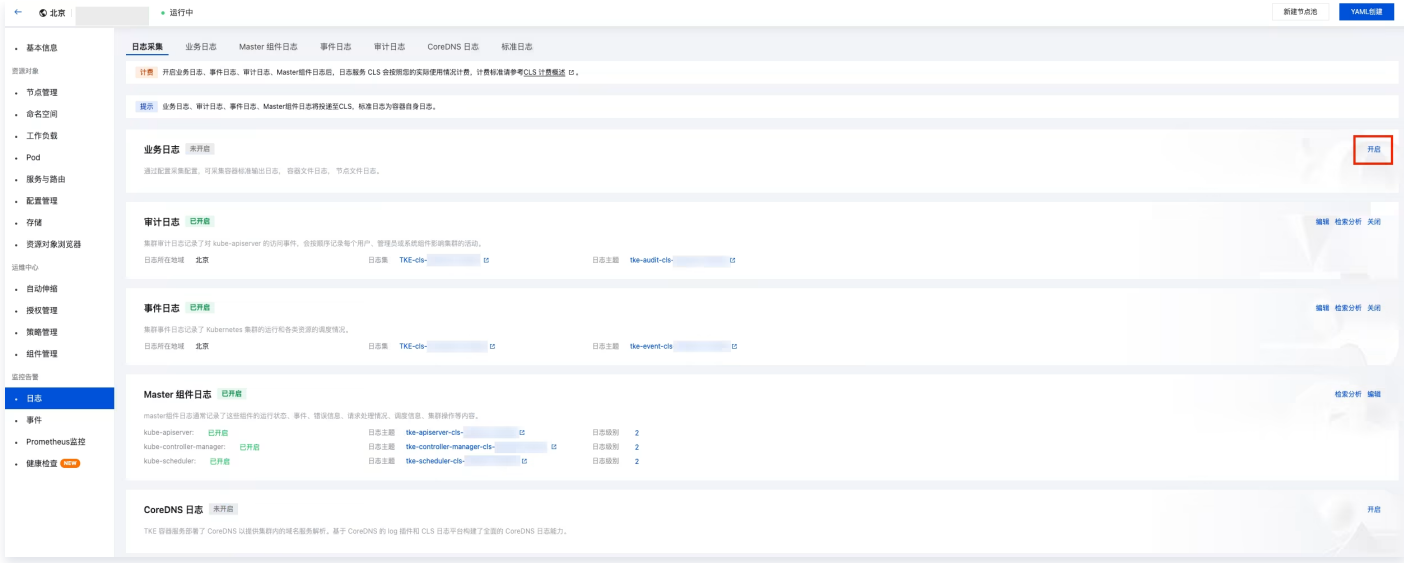
过滤器

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。

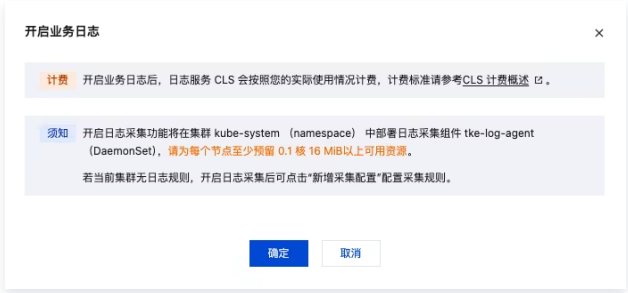
操作步骤

开启日志采集

- 登录 [容器服务控制台](#)，在左侧导航栏中选择**集群**。
- 在集群列表中，单击目标集群 ID，进入集群详情页。
- 选择左侧导航栏中的**日志**，在日志采集页签中单击**业务日志**右侧的**开启**。如下图所示：



4. 在开启业务日志中，单击确定。如下图所示：



5. 开启后，您可以在日志采集页查看日志采集状态。如下图所示：



关闭日志采集

1. 登录 [容器服务控制台](#)，在左侧导航栏中选择**集群**。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航栏中的**日志**，在日志采集页签中单击**业务日志**右侧的**关闭**。如下图所示：

←北京运行中

新建节点池YAML创建

• 基本信息

资源对象

• 节点管理

• 命名空间

• 工作负载

• Pod

• 服务与路由

• 配置管理

• 存储

• 资源对象浏览器

运维中心

• 自动伸缩

• 授权管理

• 策略管理

• 组件管理

监控告警

• 日志

日志采集业务日志Master 组件日志事件日志审计日志CoreDNS 日志标准日志

计费 开启业务日志、事件日志、审计日志、Master组件日志后，日志服务 CLS 会按照您的实际使用情况计费。计费标准请参考CLS 计费概述。

提示 业务日志、审计日志、事件日志、Master组件日志将投递至CLS，标准日志为容器自身日志。

业务日志 已开启

新增采集配置 关闭

通过配置采集配置，可采集容器标准输出日志，容器文件日志，节点文件日志。

日志采集组件版本 1.1.19 当前已是最新版本

采集规则名称类型消费端类型提取模式创建时间操作

容器标准输出CLSJSON2025-02-27 17:00:00检索分析 编辑 删除

节点文件路径CLSJSON2025-03-26 11:18:28检索分析 编辑 删除

第 1 页20 条 / 页

审计日志 已开启

编辑 检索分析 关闭

集群审计日志记录了对 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。

日志所在地域 北京 日志集 TKE-clr- 日志主题 tke-audit-clr-

4. 在关闭业务日志采集中，勾选“我已悉悉关闭须知，确认关闭当前集群业务日志采集”，单击确定。如下图所示：

关闭业务日志采集

关闭采集须知

• 点击关闭后，将停止采集当前所选集群业务日志，您所配置的采集规则将不再生效

• 关闭后，仅停止新增日志的投递。存量日志将持续存储在日志主题中直至过期，期间将持续产生收取存储费用

• 若需删除日志主题，请前往日志服务 - 日志主题 删除。

我已悉悉关闭须知，确认关闭当前集群业务日志采集

确定 取消

采集容器日志到 CLS

最近更新时间：2025-09-11 16:39:02

本文将介绍如何在容器服务控制台配置日志采集规则并投递到 [腾讯云日志服务 CLS](#)。

使用限制

配置日志采集规则前，需了解 Loglistener 采集数据时在文件采集、配置、资源、性能、错误处理等方面的能力与限制，以及相关使用说明。详情请参见 [LogListener 限制说明](#)。

操作步骤

创建日志采集规则

1. 登录 [容器服务控制台](#)，在左侧导航栏中选择**集群**。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航栏中的**日志**，在日志采集页签中单击**业务日志**右侧的**新增采集配置**。如下图所示：



4. 在**新建日志采集规则**页面中配置收集规则名称，您可以自定义日志收集规则名称。
5. 参考如下配置日志服务消费端：
 - **类型**：选择 **CLS**。
 - **日志所在地域**：CLS 支持跨地域日志投递，您可通过下拉框选择日志投递的目标地域。
 - **日志集**：根据日志所在地域展示您已经创建的日志集，若现有日志集不合适，您可以在 [日志服务控制台](#) 新建日志集。操作详情见 [创建日志集](#)。
 - **日志主题**：选择日志集下对应的日志主题，支持“自动创建”和“选择已有”日志主题两种模式。

- 说明：**
- CLS 不支持国内和境外地域之间跨地域日志投递；针对 CLS 未开通日志服务的地域仅支持就近投递，如深圳集群采集的容器日志仅支持投递到广州，天津集群采集的容器日志仅支持投递到北京；详情可通过控制台查看。
 - 一个日志主题目前仅支持配置一类日志，即日志、审计、事件不可采用同一个 topic，会产生覆盖情况，请确保所选日志主题没有被其他采集配置占用；若日志集下已存在500个日志主题，则不能新建日志主题。

6. 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**。

容器标准输出日志

- ④ 说明：
采集集群内任意服务下的容器日志，仅支持 Stderr 和 Stdout 的日志。

日志源支持**所有容器**、**指定工作负载**、**指定 Pod Labels** 三种类型。如下图所示：

所有容器

日志源

所有容器 指定工作负载 指定 Pod Labels

命名空间指定方式 **指定命名空间** 排除命名空间

所属Namespace default, kube-node-lease, k... ▼

① kube-system命名空间下会默认采集loglistener等系统组件的日志，若组件异常可能会存在大量日志造成额外成本，建议不采集该命名空间下日志。

全部命名空间：包含当前以及未来创建的所有命名空间。
全选当前命名空间：即选中当前所有命名空间，不包含未来创建的命名空间。

- 命名空间指定方式：选择命名空间指定方式，支持**指定命名空间**和**排除命名空间**。
- 所属 Namespace：指定命名空间时，支持勾选**全部命名空间**或**全选当前命名空间**。
 - 全部命名空间：包含当前以及未来创建的所有命名空间。
 - 全选当前命名空间：即选中当前所有命名空间，不包含未来创建的命名空间。

需注意，kube-system 命名空间下会默认采集 loglistener 等系统组件的日志，若组件异常可能会存在大量日志造成额外成本，建议不采集该命名空间下日志。

指定工作负载

日志源

所有容器 **指定工作负载** 指定 Pod Labels

所属Namespace default ▼

采集对象

工作负载类型	☐ 列表	Q 请输入关键词进行搜索
Deployment(0/7)	☐ dify-tke-dify-api	所有容器 ▼
DaemonSet(未加载)	☐ dify-tke-dify-plugin-d...	所有容器 ▼
StatefulSet(未加载)	☐ dify-tke-dify-proxy	所有容器 ▼
CronJob(未加载)	☐ dify-tke-dify-sandbox	所有容器 ▼
Job(未加载)	☐ dify-tke-dify-web	所有容器 ▼
	☐ dify-tke-dify-worker	所有容器 ▼
	☐ nlinx	所有容器 ▼

最少选择一个工作负载

添加Namespace

指定 Pod Labels

日志源

所有容器

指定工作负载

指定 Pod Labels

命名空间指定方式

指定命名空间

排除命名空间

所属Namespace

default, kube-node-lease, k...

全部命名空间：包含当前以及未来创建的所有命名空间
全选当前命名空间：即选中当前所有命名空间，不包含未来创建的命名空间

Pod Label=

▼

删除

新增

收集规则收集的日志会带上metadata，并上报到消费端
标签名称及标签值只能包含字母、数字及分隔符("-", "_", ".", "/"), 且必须以字母、数字开头和结尾
支持匹配同一个key下多个value值的pod，例填写environment = production,qa表示当key为environment，value值为production或qa时，均会被匹配。注意输入多个value值时请使用逗号隔开。

容器名称

等于

▼

请输入容器名称

若要采集符合以上Label的全部容器日志，此处可为空。

- 命名空间指定方式：选择命名空间指定方式，支持指定命名空间和排除命名空间。
- 所属 Namespace：指定命名空间时，支持勾选全部命名空间或全选当前命名空间。
 - 全部命名空间：包含当前以及未来创建的所有命名空间。
 - 全选当前命名空间：即选中当前所有命名空间，不包含未来创建的命名空间。
- Pod Label：新增需匹配的 Pod 标签。
- 容器名称：新增容器名称，支持等于或不等于。若要采集符合以上 Label 的全部容器日志，该项可为空。

容器内文件日志

❗ 说明：
采集集群内指定容器内的文件日志。

- 日志源支持指定工作负载、指定 Pod Labels 两种类型。
- 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`。如下图所示：

类型

容器标准输出

容器文件路径

节点文件路径

采集集群内指定容器内的文件日志。[查看示例](#)

日志源

指定工作负载

指定 Pod Labels

工作负载选项

default

▼

Deployment

▼▼

容器名▼

采集路径

日志文件夹，支持通配符 * 和 ?

/**/

日志文件名，支持通配符 * 和 ?

删除

新增采集路径

日志文件夹以开头，文件名以非开头。采集路径不支持软链接，不可包含逗号。多采集路径模式下，文件夹不支持使用 * 或 ? 通配符

采集路径黑名单

☐ 黑名单配置可在采集时忽略指定的目录和文件，目录和文件名可以是完整匹配，也支持通配符模式匹配，需要 LogListener-2.3.9 及以上版本
黑名单配置可在采集时忽略指定的目录和文件，该功能仅支持消费端类型为cls。

类型

容器标准输出

容器文件路径

节点文件路径

采集集群内指定容器内的文件日志。[查看示例](#)

日志源

指定工作负载

指定 Pod Labels

命名空间指定方式

指定命名空间

排除命名空间

所属Namespace

请选择

① kube-system命名空间下会默认采集loglistener等系统组件的日志，若组件异常可能会存在大量日志造成额外成本，建议不采集该命名空间下日志。

Pod Label

新增

收集规则收集的日志会带上metadata，并上报到消费端

容器名称

等于

请输入容器名称

若要采集符合以上Label的全部容器日志，此处填*。

采集路径

日志文件夹，支持通配符 * 和 ?

/**/

日志文件名，支持通配符 * 和 ?

删除

新增采集路径

日志文件夹以/开头，文件名以非/开头。采集路径不支持软链接，不可包含逗号。多采集路径模式下，文件夹不支持使用*或?通配符

采集路径黑名单

☐

黑名单配置可在采集时忽略指定的目录和文件，目录和文件名可以是完整匹配，也支持通配符模式匹配，需要LogListener-2.3.9及以上版本

黑名单配置可在采集时忽略指定的目录和文件，该功能仅支持消费端类型为cls。

⚠ 注意：

“容器文件路径” 不能为符号链接或硬链接，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。

节点文件日志

① 说明：

采集集群内指定节点路径的文件。

- 采集路径支持以文件路径和通配规则的方式填写，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`。
- 您可根据实际需求自定义添加 Key-Value 形式的 metadata，metadata 将会添加到日志记录中。

类型

容器标准输出

容器文件路径

节点文件路径

采集集群内指定节点路径的文件。[查看示例](#)

日志源

采集路径

日志文件夹，支持通配符 * 和 ?

/**/

日志文件名，支持通配符 * 和 ?

删除

新增采集路径

日志文件夹以/开头，文件名以非/开头，log-agent将监听所填文件夹下的所有层级匹配上的日志文件。采集路径不支持软链接，不可包含逗号。多采集路径模式下，文件夹不支持使用*或?通配符

采集路径黑名单

☐

黑名单配置可在采集时忽略指定的目录和文件，目录和文件名可以是完整匹配，也支持通配符模式匹配，需要LogListener-2.3.9及以上版本

黑名单配置可在采集时忽略指定的目录和文件，该功能仅支持消费端类型为cls。

⚠ 注意：

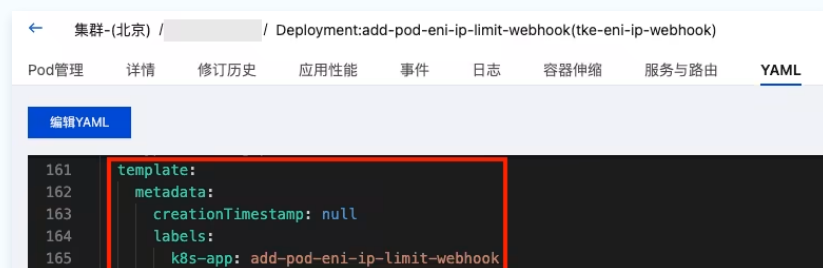
- “节点文件路径” 不能为软链接或硬链接，否则会导致软链接的实际路径在采集器不存在，采集日志失败。

- 一个节点日志文件只能被一个日志主题采集。

❗ 说明：

- 请确保 Pod 的 label 配置正确，即 Pod 的 label，而非工作负载的 label。您可以按照以下步骤查询：

- 1.1 登录 [容器服务控制台](#)，选择左侧导航栏中的集群。
- 1.2 在集群管理页面，选择集群 ID，进入集群的基本信息页面。
- 1.3 在工作负载中，选择具体的工作负载名称，进入工作负载详情页。
- 1.4 在 YAML 页签中，查看 template 下的 labels。如下图所示：



- 对于“容器的标准输出”及“容器内文件”（不包含“节点文件路径”即 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_ip	日志所属 pod 的 IP 地址。
cluster_id	日志所属的集群 ID
pod_label_{label name}	日志所属 pod 的 label（例如一个 pod 带有两个 label: app=nginx, env=prod，则在上传的日志中会附带两个 metadata: pod_label_app:nginx, pod_label_env:prod）。

7. 元数据、索引配置：

- 默认元数据：CLS 默认将元数据 `pod_name`、`namespace`、`container_name` 设为索引用于日志检索。
- 自定义元数据：支持自定义元数据及日志索引。

❗ 说明：

自定义元数据和基于元数据开启索引创建后不能修改，您可前往 [日志服务控制台](#) 修改。

8. 配置采集策略。您可以选择全量或者增量。

- 全量：全量采集指从日志文件的开头开始采集。
- 增量：增量采集指从距离文件末尾 1M 处开始采集（若日志文件小于 1M，则等同于全量采集）。

⚠ 注意：**全量采集场景**

- 采集规则生效延时：为确保采集到 Pod 启动过程中的所有关键日志，请配置全量采集策略。全量采集策略能够在采集规则完全生效前，无差别地收集所有日志数据，有效避免日志丢失现象。

增量采集场景

- 采集路径为云硬盘 CBS：采集路径为云硬盘时建议配置增量采集策略，以免 Pod 重建产生重采。

异常场景

- 采集路径为文件存储 CFS：配置全量采集会造成重采，且多个 Pod 可能同时上报相同日志，造成日志服务重复计费，建议日志源不要使用 CFS。

9. 单击下一步，选择日志解析方式。

- 编码模式：支持 **UTF-8** 和 **GBK**。
- 提取模式：支持多种类型的提取模式，详情如下：

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
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
Nginx 日志模板	适用于 Nginx 日志。	-
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
组合解析	当您的日志结构太过复杂，涉及多种解析模式，单种解析模式（如 Nginx 模式、完整正则模式、JSON 模式等）无法满足日志解析需求时，您可以使用 LogListener 组合解析格式解析日志，此模式支持用户在控制台输入代码（JSON 格式）用来定义日志解析的流水线逻辑。您可添加一个或多个 Loglistener 插件处理配置，Loglistener 会根据处理配置顺序逐一执行。	组合解析格式

- 过滤器：LogListener 仅采集符合过滤器规则的日志，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。您可以根据需求开启过滤器并配置规则。

💡 说明：

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。

10. 单击完成，完成投递到 CLS 的容器日志采集规则创建。**更新日志规则**

- 登录 [容器服务控制台](#)，在左侧导航栏中选择**集群**。

2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航栏中的日志，在日志采集页签中单击采集规则右侧的编辑。如下图所示：

业务日志 已开启

新增采集配置 关闭

通过配置采集配置，可采集容器标准输出日志，容器文件日志，节点文件日志。

日志采集组件版本 1.1.19 当前已是最新版本

请输入日志名称

采集规则名称	类型	消费端类型	提取模式	创建时间	操作
	容器标准输出	CLS	JSON	2025-02-27 17:00:00	检索分析 编辑 删除
	节点文件路径	CLS	JSON	2025-03-26 11:18:28	检索分析 编辑 删除

第 1 页

20 条 / 页

4. 根据需求更新相应配置，单击完成。

 **注意：**
日志集和日志主题不可更新。

相关文档

您不仅可以使用控制台配置日志采集，还可以通过自定义资源（ CustomResourceDefinitions，CRD ）的方式配置日志采集。详情见 [使用 CRD 配置日志采集](#)。

使用 CRD 配置日志采集

使用 CRD 进行日志采集配置指导

最近更新时间：2025-02-21 10:23:23

操作场景

您不仅可以 [使用控制台配置日志采集](#)，还可通过自定义资源（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 [运维功能管理](#) 中开启日志采集，操作详情请参见 [开启日志采集](#)。

采集器限制

Kafka采集器限制

项目	能力与限制
采集规则	日志规则最大能够同时采集监听约256个文件，如果当前采集超出此限制，请 提交工单 。
单条日志大小	- 标准输出模式下单行最大长度为16KB，超过此长度会导致日志截断。 - 容器文件和节点文件模式下： - 日志单行最大长度小于256KB时，单日志规则最大能够同时监听采集约80个文件。 - 日志单行最大长度大于256KB且小于2MB时，单日志规则最大能够同时监听采集约10个文件。 - 日志单行最大长度不能超过2MB。如果当前采集超出此限制，请 提交工单。
标签等采集方式	如果使用标签等方式批量采集文件，需要确保当前日志规则下所有被采集容器的目标路径均存在，否则当前日志规则会出现采集失败的情况。
处理能力	- Kafka 采集日志开启 Metadata 与关闭时采集能力基本持平，同一个日志规则的所有被采集文件共享吞吐能力。 - 单日志规则单文件情况下，每日志大小为25KB时，采集器单行文本提取模式处理能力约为40MB/s，采集器 JSON 提取模式处理能力约为4.5MB/s。

CRD 介绍

结构总览

```
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:                                   ## 投递到CLS的配置
    ...
  inputDetail:                                 ## 采集数据源配置
    ...
  kafkaDetail:                                ## 投递到 ckafka 或者自建kafka配置
    ...
status:                                        ## CRD资源状态
  status: ""
  code: ""                                     ## 调用接口出错时，接口返回的错误码
  reason: ""                                  ## 出错原因
```

clsDetail 字段说明

⚠ 注意：

topic 指定后不允许修改。

```
clsDetail:
  ## 自动创建日志主题，需要同时指定日志集和主题的名称
  logsetName: test          ## CLS日志集的名称，若无该名称的日志集，会自动创建，若有，会在该日志集下
  创建日志主题
  topicName: test           ## CLS日志主题的名称，若无该名称的日志主题，会自动创建

  # 选择已有日志集日志主题，如果指定了日志集未指定日志主题，则会自动创建一个日志主题
  logsetId: xxxxxx-xx-xx-xx-xxxxxxx ## CLS日志集的ID，日志集需要在CLS中提前创建
  topicId: xxxxxx-xx-xx-xx-xxxxxxx ## 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: json_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，
  multiline_fullregex_log代表多行正则格式。默认为minimalist_log
  logFormat: xxx             ## 日志格式化方式
  period: 30                 ## 生命周期，单位天，可取值范围1~3600。取值为3640时代表永久保存
  partitionCount:            ## Integer 类型，日志主题分区个数。默认创建1个，最大支持创建10个分区。
  tags:                      ## 标签描述列表，通过指定该参数可以同时绑定标签到相应的日志主题。最大支持9
  个标签键值对，同一个资源只能绑定到同一个标签键下。
  - key: xxx                 ## 标签key
    value: xxx               ## 标签value
  autoSplit: false          ## boolean 类型，是否开启自动分裂，默认值为true
  maxSplitPartitions:
  storageType: hot           ## 日志主题的存储类型，可选值 hot（标准存储），cold（低频存储）；默认为
  hot。
  excludePaths:             ## 采集黑名单路径列表
  - excludeType: File        ## 类型，选填File或Path
    value: /xx/xx/xx/xx.log  ## type 对应的值
  indexes:                  ## 创建 topic 时可自定义索引方式和字段
  - indexName:               ## 需要配置键值或者元字段索引的字段，元字段Key无需额外添加__TAG__前缀，
  与上传日志时对应的字段Key一致即可，腾讯云控制台展示时将自动添加__TAG__前缀
    indexType:               ## 字段类型，目前支持的类型有：long、text、double
    tokenizer:               ## 字段的分词符，其中的每个字符代表一个分词符；仅支持英文符号及\n\t\r；
  long及double类型字段需为空；text类型字段推荐使用 @&?|#()="",;:<>[]{} / \n\t\r\ 作为分词符；
    sqlFlag:                 ## boolean 字段是否开启分析功能
    containZH:               ## boolean 是否包含中文
  region: ap-xxx            ## topic 所在地域，用于跨地域投递
  userDefineRule: xxxxxxx    ## 用户自定义采集规则，Json格式序列化的字符串
  extractRule: {}           ## 提取、过滤规则。如果设置了ExtractRule，则必须设置LogType
```

inputDetail 字段说明

```
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。支持多个命名空间，如果有多个命名空间使用", "分隔，如：
  default,namespace。如果不指定，代表所有命名空间。注意：与 excludeNamespace 不能同时指定
  excludeNamespace: nm1,nm2 ## 排除采集容器的kubernetes命名空间。支持多个命名空间，如果有多个命名空间使
  用", "分隔，如：nm1,nm2。如果不指定，代表所有命名空间。注意：与 namespace 不能同时指定
  nsLabelSelector: environment in (production),tier in (frontend) ## 根据命名空间label 筛选符合的
  namespace
  allContainers: false      ## 是否采集指定命名空间中的所有容器的标准输出。注意：allContainers=true 时不能同
  时指定 workload, includeLabels 和 excludeLabels
  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。
  excludeLabels:           ## 采集不包含指定label的Pod，与workload, namespace 和 excludeNamespace 不能同时指定
```

```

    key2: value2  ## 支持匹配同一个key下多个value值的pod,例填写environment = production,qa表示当key为
environment, value值为production或qa时,均会被排除,注意输入多个value值时请使用逗号隔开。如果同时指定了
includeLabels,则匹配与 includeLabels 交集的pod

includeLabels:  ## 采集包含指定label的Pod,与workload, namespace 和 excludeNamespace 不能同时指定
    key: value1  ## 收集规则收集的日志会带上metadata,并上报到消费端。支持匹配同一个key下多个value值的pod,例
填写environment = production,qa表示当key为environment, value值为production或qa时,均会被匹配,注意输入多个value
值时请使用逗号隔开。 如果同时指定了 excludeLabels,则匹配与 excludeLabels 交集的pod

metadataLabels:      ## 指定具体哪些pod label被当作元数据采集,如果不指定,则采集所有pod label为元数
据
- label1
customLabels:        ## 用户自定义metadata
    label: l1

workloads:
- container: xxx      ## 要采集的容器名,如果不指定,代表workload Pod中的所有容器
  kind: deployment    ## workload类型,支持deployment、daemonset、statefulset、job、cronjob
  name: sample-app    ## workload的名字
  namespace: prod     ## workload的命名空间

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
  excludeNamespace: nm1,nm2  ## 排除采集容器的kubernetes命名空间。支持多个命名空间,如果有多个命名空间使
用", "分隔,如: nm1,nm2。 如果不指定,代表所有命名空间。 注意:与 namespace 不能同时指定
  nsLabelSelector: environment in (production),tier in (frontend) ## 根据命名空间label 筛选符合的
namespace
  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
  logPath: /var/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
  customLabels:  ## 用户自定义metadata
    key: value
  excludeLabels:  ## 采集不包含指定label的Pod,与workload不能同时指定
    key2: value2  ## 支持匹配同一个key下多个value值的pod,例填写environment = production,qa表示当key为
environment, value值为production或qa时,均会被排除,注意输入多个value值时请使用逗号隔开。如果同时指定了
includeLabels,则匹配与 includeLabels 交集的pod

includeLabels:  ## 采集包含指定label的Pod,与workload不能同时指定
    key: value1  ## 收集规则收集的日志会带上metadata,并上报到消费端。支持匹配同一个key下多个value值的pod,例
填写environment = production,qa表示当key为environment, value值为production或qa时,均会被匹配,注意输入多个value
值时请使用逗号隔开。 如果同时指定了 excludeLabels,则匹配与 excludeLabels 交集的pod
  metadataLabels:      ## 指定具体哪些pod label被当作元数据采集,如果不指定,则采集所有pod label为元数据
- label1              ## pod label
  workload:
    container: xxx      ## 要采集的容器名,如果不指定,代表workload Pod中的所有容器
    name: sample-app    ## workload的名字

hostFile:          ## 节点文件路径
  filePattern: '*.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
  logPath: /tmp/logs   ## 日志文件夹,不支持通配符
  customLabels:        ## 用户自定义metadata
    label1: v1

```

extractRule 对象说明

名称	类型	必填项	描述
timeKey	String	否	时间字段的 key 名字, time_key 和 time_format 必须成对出现。

timeFormat	String	否	时间字段的格式，参考 C 语言的 strftime 函数对于时间的格式说明输出参数。
delimiter	String	否	分隔符类型日志的分隔符，只有 log_type 为 delimiter_log 时有效。
logRegex	String	否	整条日志匹配规则，只有 log_type 为 fullregex_log 时有效。
beginningRegex	String	否	行首匹配规则，只有 log_type 为 multiline_log 或 fullregex_log 时有效。
unMatchUpload	String	否	解析失败日志是否上传，true 表示上传，false 表示不上传。
unMatchedKey	String	否	失败日志的 key。
backtracking	String	否	增量采集模式下的回溯数据量，默认-1（全量采集），0 表示增量。
keys	Array of String	否	取的每个字段的 key 名字，为空的 key 代表丢弃这个字段，只有 log_type 为 delimiter_log 时有效，json_log 的日志使用 json 本身的 key。
filterKeys	Array of String	否	需要过滤日志的 key，与 FilterRegex 按下标进行对应。
filterRegex	Array of String	否	需要过滤日志的 key 对应的 regex，与 FilterKeys 按下标进行对应。
isGBK	String	否	是否为 Gbk 编码。0: 否，1: 是。 注意： 此字段可能返回 null，表示取不到有效值。
jsonStandard	String	否	是否为标准 json。0: 否，1: 是。 注意： 此字段可能返回 null，表示取不到有效值。

kafkaDetail 字段说明

```

kafkaDetail:
  brokers: x.x.x.x:p      ## 必填，broker地址，一般是域名:端口，多个地址以“,”分隔
  topic: test
  kafkaType: CKafka      ## kafka 类型，CKafka - ckafka, SelfBuildKafka - 自建kafka
  instanceId: xxxx       ## 当 kafkaType = CKafka，设置ckafka实例 id
  logType: minimalist_log ## kafka 日志解析类型，"minimalist_log" 或 "" 单行全文，"multiline_log" 多行全文，"json" json 格式
  timestampFormat: xxx    ## 时间戳的格式，默认是double
  timestampKey: xxx       ## 时间戳的key值，默认是"@timestamp"
  metadata:
    formatType: default   ## metadata 格式。 "default" 默认格式（与 eks kafka 采集器相同），"filebeat"
    filebeat 格式，"fluent-bit" fluent-bit 格式
    messageKey:           ## 支持指定一个Key，将日志投递到指定分区。默认不开启，日期随机投放；开启后带有同样Key的日志，
    将投递到相同的分区里。支持选择Pod字段作为Key，以Pod name为例，请选择Field>metadata.name
    value: Field          ## 必填，topicID
    valueFrom:
      fieldRef:
        fieldPath: metadata.name ## 当key为Field时可选
  metadata.name,metadata.namespace,spec.nodeName,spec.serviceAccountName

```

status 字段说明

status	说明
状态为空	初始状态
Synced	采集配置处理成功

Stale

采集配置处理失败

CRD 示例

配置容器标准输出 CRD 示例

所有容器

指定命名空间

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: "test"
spec:
  clsDetail:
    .....
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      allContainers: true
      namespace: default,kube-public
    type: container_stdout
```

排除命名空间

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: "test"
spec:
  clsDetail:
    .....
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      allContainers: true
      excludeNamespace: kube-system,kube-node-lease
    type: container_stdout
```

指定工作负载

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: "test"
spec:
  clsDetail:
    .....
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
```



```
containerStdout:
  allContainers: false
workloads:
- container: prod
  kind: deployment
  name: sample-app
  namespace: kube-system
type: container_stdout
```

指定 Pod Labels

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test
spec:
  clsDetail:
    .....
    topicId: xxxxxx-xx-xx-xx-xxxxxxx
  inputDetail:
    containerStdout:
      container: prod
      excludeLabels:
        key2: v2
      includeLabels:
        key1: v1
      namespace: default,kube-system
    type: container_stdout
```

配置容器文件路径 CRD 示例

指定工作负载

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test
spec:
  clsDetail:
    .....
    topicId: xxxx-xx-xx-xx-xxxx
  inputDetail:
    containerFile:
      container: prod
      filePattern: '*.log'
      logPath: /tmp/logs
      namespace: kube-system
    workload:
      kind: deployment
      name: sample-app
```

```
type: container_file
```

指定 Pod Labels

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test
spec:
  clsDetail:
    .....
    topicId: xxxx-xx-xx-xx-xxxx
  inputDetail:
    containerFile:
      container: prod
      filePattern: '*.log'
      includeLabels:
        key1: v1
      excludeLabels:
        key2: v2
      logPath: /tmp/logs
      namespace: default,kube-public
      type: container_file
```

配置节点文件路径 CRD 示例

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  creationTimestamp: "2022-03-13T12:48:49Z"
  generation: 4
  name: test
  resourceVersion: "11729531"
  selfLink: /apis/cls.cloud.tencent.com/v1/logconfigs/test
  uid: 233f4b72-cfef-4a43-abb8-e4d033185097
spec:
  clsDetail:
    .....
    topicId: xxxx-xx-xx-xx-xxxx
  inputDetail:
    hostFile:
      customLabels:
        testmetadata: v1
      filePattern: '*.log'
      logPath: /var/logs
      type: host_file
```

LogConfig json 格式说明

最近更新时间：2025-09-09 17:58:12

LogConfig CR 字段说明

```
{
  "kind": "cls.cloud.tencent.com/v1",
  "apiVersion": "LogConfig",
  "metadata": {
    "name": "configName" // 采集规则名
  },
  "spec": {
    "inputDetail": {...}, // 日志采集源定义
    "clsDetail": {...}, // 采集到CLS的配置
    "kafkaDetail": {...} // 采集到 Kafka 的配置
  }
}
```

spec 内容说明

采集规则 spec 部分主要定义了日志的采集源以及日志输出端的配置信息。

⚠ 注意：

clsDetail 和 kafkaDetail 在同一个采集规则中只能配置一种。

字段	是否必填	说明
inputDetail	是	定义采集源。
clsDetail	否	CLS 采集器配置，该配置信息将由日志组件 cls-provisioner 同步到 CLS 侧。
kafkaDetail	否	Kafka 采集器配置，定义日志采集格式等信息。

inputDetail 说明

```
{
  "type": "host_file",
  "containerStdout": {},
  "containerFile": {},
  "hostFile": {}
}
```

字段	是否必填	类型	说明
type	是	String	采集类型，取值如下： - container_file：容器内文件采集，该类型对应 <code>containerFile</code> 配置项。 - container_stdout：采集容器标准输出。 - host_file：采集节点日志，超级节点不支持该配置。
containerStdout	否	Object	容器标准输出采集配置。配置采集容器标准输出时，需指定采集类型为 container_stdout。
containerFile	否	Object	容器内路径采集。配置采集容器标准输出时，需指定采集类型为 container_file。
hostFile	否	Object	节点路径采集。配置采集容器标准输出时，需指定采集类型为 container_file。

containerStdout 对象结构说明

```
{
  "includeLabels": {
    "label1": "value1"
  },
  "excludeLabels": {
    "label": "value"
  },
  "metadataLabels": [
    "label1"
  ],
  "metadataContainer": [
    "namespace",
    "pod_name",
    "pod_ip",
    "pod_uid",
    "container_id",
    "container_name",
    "image_name",
    "cluster_id"
  ],
  "container": "container",
  "workloads": [
    {
      "container": "kubernetes-proxy",
      "kind": "deployment",
      "name": "kubernetes-proxy",
      "namespace": "default"
    },
    {
      "container": "testlog11",
      "kind": "deployment",
      "name": "testlog1",
      "namespace": "default"
    }
  ],
  "customLabels": {
    "xx": "xx"
  }
}
```

字段	是否必填	类型	说明
includeLabels	否	Map	采集包含指定 label:value 的 pod，不能与 workloads 同时配置。
excludeLabels	否	Map	排除包含指定 label:value 的 pod，不能与 workloads 同时配置。
metadataLabels	否	Array	定义将 pod 的哪些 label 作为元数据上报。不指定则全部上报。
metadataContainer	否	Array	定义将 pod、container 的哪些信息作为元数据上报。不指定则全部上报。
customLabels	否	Map	用户自定义元数据。
container	否	String	指定采集哪些容器的日志。不指定或配置 <code>*</code> ，则采集命中 pod 的所有容器的标准输出。不能与 workloads 同时配置。
workloads	否	Array	指定需要采集哪些 workload 的标准输出，同时指定采集的容器名称。

workload 对象说明

字段	是否必填	类型	说明
container	否	String	需要采集的容器名称。不指定，则采集所有容器标准输出。
kind	是	String	workload 类型。
name	是	String	workload 名称。
namespace	是	String	workload 所在 namespace。

containerFile 对象结构说明

```
{
  "logPath": "/var/logs",
  "filePattern": "*.log",
  "filePaths": [
    {
      "path": "/var/logs",
      "file": "*.log"
    }
  ],
  "customLabels": {
    "key": "value1"
  },
  "namespace": "default",
  "nsLabelSelector": "",
  "excludeNamespace": "xxx,xx",
  "includeLabels": {
    "label": "value"
  },
  "excludeLabels": {
    "exLabel": "exvalue"
  },
  "metadataLabels": [
    "xx"
  ],
  "metadataContainer": [
    "namespace",
    "pod_name",
    "pod_ip",
    "pod_uid",
    "container_id",
    "container_name",
    "image_name",
    "cluster_id"
  ],
  "container": "xxx",
  "workload": {
    "name": "xxx",
    "kind": "deployment"
  }
}
```

字段	是否必填	类型	说明
logPath	否	String	采集路径。不可与 filePaths 同时配置。

filePattern	否	String	预期采集文件匹配模式，不可与 filePaths 同时配置。通配符支持 * 和 ? 。
filePaths	是	Array	配置预期采集容器内的路径及文件匹配模式，可配置多个采集路径。
includeLabels	否	Map	采集包含指定 label:value 的 pod，不能与 workload 同时配置。
excludeLabels	否	Map	排除包含指定 label:value 的 pod，不能与 workload 同时配置。
metadataLabels	否	Array	定义将 pod 的哪些 label 作为元数据上报。不指定则全部上报。
metadataContainer	否	Array	定义将 pod、container 的哪些信息作为元数据上报。不指定则全部上报。
customLabels	否	Map	用户自定义元数据。
container	否	String	指定采集哪些容器的日志。
workload	否	Object	指定需要采集哪些 workload 的标准输出，同时指定采集的容器名称。

workload 字段说明

字段	是否必填	类型	说明
kind	是	String	workload 类型。
name	是	String	workload 名称。

filePath 对象字段说明

字段	是否必填	类型	说明
path	是	String	预期采集的日志所在路径。
file	是	String	预期采集的日志文件名称匹配模式，通配符支持 * 和 ? 。

hostFile 对象结构说明

```
{
  "logPath": "/var/logs",
  "filePattern": "*.log",
  "filePaths": [
    {
      "path": "/var/log",
      "file": "*.log"
    }
  ],
  "customLabels": {
    "key": "value"
  }
}
```

字段	是否必填	类型	说明
logPath	否	String	采集路径。不可与 filePaths 同时配置。
filePattern	否	String	预期采集文件匹配模式，不可与 filePaths 同时配置。通配符支持 * 和 ? 。
filePaths	否	Array	配置预期采集容器内的路径及文件匹配模式，可配置多个采集路径。
customLabels	否	Map	用户自定义元数据。

clsDetail 对象结构说明

```
{
  "topicId": "",
  "topicName": "",
  "logsetId": "",
  "logsetName": "",
  "logFormat": "",
  "logType": "",
  "hotPeriod": 7,
  "period": 30,
  "partitionCount": 1,
  "tags": [
    {
      "key": "key",
      "value": "value"
    }
  ],
  "autoSplit": false,
  "maxSplitPartitions": 50,
  "storageType": "hot",
  "extractRule": {...},
  "excludePaths": [
    {
      "excludeType": "File",
      "value": "/var/log/1.log"
    },
    {
      "excludeType": "Path",
      "value": "/var/log/1/"
    }
  ],
  "fullTextIndex": {...},
  "indexs": [
    {...}
  ],
  "indexStatus": "off",
  "region": "ap-xxx",
  "advancedConfig": {...}
}
```

字段名	是否必填	类型	说明
topicId	否	String	日志主题 ID。
topicName	否	String	日志主题名称。当 topicId 为空时，指定日志主题名称，可根据日志主题名称自动拉取日志主题 ID，或者自动创建日志主题。
logsetId	否	String	日志集 ID。
logsetName	否	String	日志集名称。当 logsetId 为空时，指定日志集名称，可根据日志集名称自动拉取日志集 ID，或者自动创建日志集。
region	否	String	日志主题所在 region。默认为集群所在 region。
logType	否	String	日志类型。可选值：json_log（json 格式）、delimiter_log（分隔符提取）、minimalist_log（单行全文格式）、multiline_log（多行全文格式）、fullregex_log（完全正则格式）。默认为 minimalist_log。
logFormat	否	String	日志格式化。

storageType	否	String	日志主题存储类型（仅在自动创建 topic 时生效）。可选值：hot（标准存储）、cold（低频存储）。默认 hot。
hotPeriod	否	Number	日志沉降开关（仅在自动创建 topic 时生效）。0：关闭日志沉降，非 0：开启日志沉降后标准存储的天数。HotPeriod 需要大于等于 7，且小于 Period。仅在 StorageType 为 hot 时生效。
period	否	Number	日志存储周期（单位天）（仅在自动创建 topic 时生效）。标准存储取值范围 1~3600，低频存储取值范围 7~3600 天。取值为 3640 时代表永久保存。默认 30 天。
partitionCount	否	Number	日志主题分区个数（仅在自动创建 topic 时生效）。默认创建 1 个，最大支持创建 10 个分区。
tags	否	Array	标签列表（仅在自动创建 topic 时生效）。
autoSplit	否	Bool	是否开启自动分裂，“false”或"true"（仅在自动创建 topic 时生效）。
maxSplitPartitions	否	Number	开启自动分裂后，每个日志主题能够容许的最大分区数，默认为 50 个（仅在自动创建 topic 时生效）。
extractRule	否	Object	采集规则相关配置。
excludePaths	否	Array	采集黑路径配置。
fullTextIndex	否	Object	全文索引配置（仅在自动创建 topic 时生效）。
indexs	否	Array	索引配置（仅在自动创建 topic 时生效）。
indexStatus	否	String	索引状态（仅在自动创建 topic 时生效）。
autoIndex	否	String	是否开启自动索引（仅在自动创建 topic 时生效）。
advancedConfig	否	Object	高级采集配置。

tag 对象结构说明

```
{
  "key": "key",
  "value": "value"
}
```

字段	类型	是否必填	说明
key	String	是	标签名称。
value	String	是	标签值。

extractRule 对象结构说明

```
{
  "timeKey": "xxx",
  "timeFormat": "xxx",
  "delimiter": "xxx",
  "logRegex": "xxx",
  "beginningRegex": "xxx",
  "keys": [
    "xx"
  ],
  "filterKeys": [
    "xx"
  ],
  "filterRegex": [
```



```

    "xx"
  ],
  "unMatchUpload": "true",
  "unMatchedKey": "parseFailed",
  "backtracking": "-1",
  "isGBK": "true",
  "jsonStandard": "true",
  "advancedFilters": [
    {
      "key": "level",
      "rule": 0,
      "value": "info"
    }
  ]
}

```

字段名	是否必填	类型	说明
timeKey	否	String	时间字段的 key 名字，timeKey 和 timeFormat 必须成对出现。
timeFormat	否	String	时间字段的格式，参考 C 语言的 strftime 函数对于时间的格式说明输出参数。
delimiter	否	String	分隔符类型日志的分隔符，只有 logType 为 delimiter_log 时有效。
logRegex	否	String	整条日志匹配规则，只有 logType 为 fullregex_log 时有效。
beginningRegex	否	String	行首匹配规则，只有 logType 为 multiline_log 或 fullregex_log 时有效。
keys	否	Array	取的每个字段的 key 名字，为空的 key 代表丢弃这个字段，只有 logType 为 delimiter_log 时有效，json_log 的日志使用 json 本身的 key。
filterKeys	否	Array	需要过滤日志的 key，与 filterRegex 按下标进行对应。
filterRegex	否	Array	需要过滤日志的 key 对应的 regex，与 filterKeys 按下标进行对应。
unMatchUpload	否	String	解析失败日志是否上传，true 表示上传，false 表示不上传。
unMatchedKey	否	String	失败日志的 key。
backtracking	否	String	增量采集模式下的回溯数据量，默认 -1（全量采集），0 表示增量。
isGBK	否	String	是否为 GBK 编码。可选值： <code>"false"</code> ：否； <code>"true"</code> ：是。注意：此字段可能返回 null，表示取不到有效值。
jsonStandard	否	String	是否为标准 JSON。可选值： <code>"false"</code> ：否； <code>"true"</code> ：是。注意：此字段可能返回 null，表示取不到有效值。
advancedFilters	否	Array	高级过滤规则，该字段仅适用于 v1.1.15 及以上的采集组件版本，低于 v1.1.15 请使用 filterKeys 与 filterRegex。

advancedFilters 对象结构说明

```

{
  "key": "level", // 需要过滤的key
  "rule": 0,      // 过滤规则 0(等于), 1(字段存在), 2(字段不存在), 3(不等于)
  "value": "info" // 需要过滤的值value。其中当rule为1或2时, value无需指定
}

```

excludePath 对象结构说明

```

{

```

```
"excludeType": "File",
"value": "/var/log/1.log"
}
```

字段	类型	是否必填	说明
excludeType	String	是	排除类型。取值 File、Path。
excludeType	String	是	排除的日志路径或者路径。

fullTextIndex 对象结构说明

```
{
  "caseSensitive": false,
  "containZH": false,
  "status": "on",
  "tokenizer": "@&()='\" , ; : < > [ ] { } \ / \n \t \r"
}
```

字段名	类型	是否必填	说明
caseSensitive	bool	否	是否大小写敏感。
containZH	bool	否	是否包含中文。
status	String	否	全文索引开关。如果不设置，则会默认打开全文索引；如果设置为 on 打开，不设置其他参数，则 caseSensitive 和 tokenizer 都会是默认值；如果设置为 off 关闭，则不打开全文索引。
tokenizer	String	否	全文索引的分词符。如果全文索引开关打开，则必须设置 tokenizer。可以设置为 "@&()='\" , ; : < > [] { } \ / \n \t \r"，为默认设置。

index 对象结构说明

```
{
  "indexName": "xxx",
  "indexType": "text",
  "tokenizer": "@&()='\" , ; : < > [ ] { } \ / \n \t \r",
  "sqlFlag": true,
  "containZH": false
}
```

字段名	类型	是否必填	说明
indexName	String	是	索引名称。
indexType	String	是	索引类型。可选值：long、text、double。
tokenizer	String	否	字段分词符。
sqlFlag	bool	否	字段是否开启分析功能。
containZH	bool	否	是否包含中文。

advancedConfig 对象结构说明

```
{
  "ClsAgentMaxDepth": 1,
  "ClsAgentFileTimeout": 60,
```

```
"ClsAgentParseFailMerge": false,
}
```

字段名	类型	是否必填	说明
ClsAgentMaxDepth	Number	否	当前配置采集时下钻的深度。
ClsAgentFileTimeout	Number	否	当前配置在采集文件时，当文件在指定时间未写入日志时，采集器释放文件句柄。
ClsAgentParseFailMerge	bool	否	采集器解析文件失败后是否合并日志上报。

kafkaDetail 对象结构说明

```
{
  "brokers": "127.0.0.1:9092",
  "topic": "test_log",
  "timestampKey": "@timestamp",
  "timestampFormat": "double",
  "extractRule": {
    "beginningRegex": "xxxx"
  },
  "logType": "minimalist_log",
  "metadata": {
    "formatType": "json"
  }
}
```

字段名	类型	是否必填	说明
brokers	String	是	Kafka brokers。
topic	String	是	Kafka topic。
timestampKey	String	否	时间戳的 key，默认 @timestamp。
timestampFormat	String	否	时间戳格式。可选值：double、iso8601。默认 double。
extractRule.beginningRegex	String	否	行首匹配规则。logType 为 multiline_log 时，该项需要配置。
logType	String	否	日志提取模式。默认单行全文。可选值：minimalist_log（单行全文）、multiline_log（多行全文）、json（json 格式）。
metadata.formatType	String	否	元数据格式。可选值：fluent-bit（fluent-bit 原生格式）、fluent-bit-string（字符串类型）。

日志采集源配置示例

容器标准输出

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",
"metadata": {
  "name": "test"
},
```

```
"spec": {
  "inputDetail": {
    "containerStdout": {
      "allContainers": true,
      "namespace": "default"
    },
    "type": "container_stdout"
  },
  ...
}
```

采集 production 命名空间中属于 ingress-gateway deployment 的 pod 中的所有容器的标准输出：

```
"apiVersion": "cls.cloud.tencent.com/v1",
"kind": "LogConfig",
"spec": {
  "inputDetail": {
    "containerStdout": {
      "allContainers": false,
      "workloads": [
        {
          "kind": "deployment",
          "name": "ingress-gateway",
          "namespace": "production"
        }
      ]
    },
    "type": "container_stdout"
  },
  ...
}
```

采集 production 命名空间中 pod 标签中包含 k8s-app=nginx 的 pod 中的所有容器的标准输出：

```
{
  "apiVersion": "cls.cloud.tencent.com/v1",
  "kind": "LogConfig",
  "spec": {
    "inputDetail": {
      "containerStdout": {
        "allContainers": false,
        "includeLabels": {
          "k8s-app": "nginx"
        },
        "namespace": "production"
      },
      "type": "container_stdout"
    },
    ...
  }
}
```

采集 production 命名空间中属于 ingress-gateway 工作负载的 pod 中，名称为 ingress 的容器的标准输出：

```
{
  "apiVersion": "cls.cloud.tencent.com/v1",
  "kind": "LogConfig",
```

```
"spec": {
  "inputDetail": {
    "containerStdout": {
      "allContainers": false,
      "includeLabels": {
        "k8s-app": "nginx"
      },
      "namespace": "production"
    },
    "type": "container_stdout"
  },
  ...
}
```

容器文件采集

采集 production 命名空间中属于 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",
"metadata": {
  "name": "test"
},
"spec": {
  "inputDetail": {
    "containerFile": {
      "container": "nginx",
      "filePaths": [
        {
          "file": "access.log",
          "path": "/data/nginx/log"
        }
      ],
      "namespace": "production",
      "workload": {
        "kind": "deployment",
        "name": "ingress-gateway"
      }
    },
    "type": "container_file"
  },
  ...
}
```

采集 production 命名空间中 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": {
      "containerFile": {
        "container": "nginx",
        "filePaths": [
          {
            "file": "access.log",
```

```
        "path": "/data/nginx/log"
      }
    ],
    "includeLabels": {
      "k8s-app": "ingress-gateway"
    },
    "namespace": "production"
  },
  "type": "container_file"
}
}
```

主机文件采集

采集主机 /data/ 路径下所有 .log 文件：

```
"apiVersion": "cls.cloud.tencent.com/v1",
"kind": "LogConfig",
"metadata": {
  "name": "test"
}
"spec": {
  "inputDetail": {
    "containerStdout": {
      "allContainers": false,
      "includeLabels": {
        "k8s-app": "nginx"
      },
      "namespace": "production"
    },
    "type": "container_stdout"
  },
  ...
}
```

完整示例

投递到 CLS 的日志采集规则，采集 default 命名空间下 deployment/testlog1 的标准输出：

```
{
  "apiVersion": "cls.cloud.tencent.com/v1",
  "kind": "LogConfig",
  "metadata": {
    "name": "123"
  },
  "spec": {
    "clsDetail": {
      "extractRule": {
        "backtracking": "0",
        "isGBK": "false",
        "jsonStandard": "false",
        "unMatchUpload": "false"
      },
      "indexs": [
        {
          "indexName": "namespace"
        }
      ]
    }
  }
}
```

```
{
  {
    "indexName": "pod_name"
  },
  {
    "indexName": "container_name"
  }
],
"logFormat": "default",
"logType": "minimalist_log",
"maxSplitPartitions": 0,
"region": "ap-chengdu",
"storageType": "",
"topicId": "c26b66bd-617e-4923-bea0-test"
},
"inputDetail": {
  "containerStdout": {
    "metadataContainer": [
      "namespace",
      "pod_name",
      "pod_ip",
      "pod_uid",
      "container_id",
      "container_name",
      "image_name",
      "cluster_id"
    ],
    "nsLabelSelector": "",
    "workloads": [
      {
        "kind": "deployment",
        "name": "testlog1",
        "namespace": "default"
      }
    ]
  },
  "type": "container_stdout"
}
}
```

投递到 Kafka 的采集规则，采集 default 命名空间下的所有 pod 标准输出：

```
{
  "apiVersion": "cls.cloud.tencent.com/v1",
  "kind": "LogConfig",
  "metadata": {
    "name": "321"
  },
  "spec": {
    "inputDetail": {
      "containerStdout": {
        "allContainers": true,
        "namespace": "default",
        "nsLabelSelector": ""
      },
      "type": "container_stdout"
    },
    "kafkaDetail": {
      "brokers": "127.0.0.1:9092",

```

```
    "extractRule": {},
    "instanceId": "",
    "kafkaType": "SelfBuildKafka",
    "logType": "minimalist_log",
    "messageKey": {
      "value": "",
      "valueFrom": {
        "fieldRef": {
          "fieldPath": ""
        }
      }
    },
  },
  "metadata": {},
  "timestampFormat": "double",
  "timestampKey": "",
  "topic": "test"
}
}
```


日志组件版本升级

最近更新时间：2025-09-08 18:09:42

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](#)。

操作步骤

- 登录 [容器服务控制台](#)，选择左侧导航栏中**运维中心** > **运维功能管理**。
- 在**运维功能管理**页面，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：



集群ID/名称	kubernetes版本	类型/状态	日志采集	集群审计	事件存储	Master日志	操作
cls-美国	1.16.3	标准集群 (Master自维护) (运行中)					日志采集 更多
cls-tcz	1.22.5	标准集群 (运行中)					日志采集 更多
cls-tke-19e	1.26.1	标准集群 (运行中)	已开启	已开启	已开启		日志采集 更多

- 选择您的集群，单击**组件可升级**，选择 tke-log-agent 组件右侧的升级。

审计管理

审计日志

最近更新时间：2025-02-17 10:22:33

简介

审计日志是基于 [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，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

审计日志功能提供了一种区别于 metrics 的集群观测维度。开启审计日志后，Kubernetes 可以记录每一次对集群的操作。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 审计日志策略

审计级别（level）

和一般日志不同，Kubernetes 审计日志的级别类似于 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 默认在收到请求时记录审计日志，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。

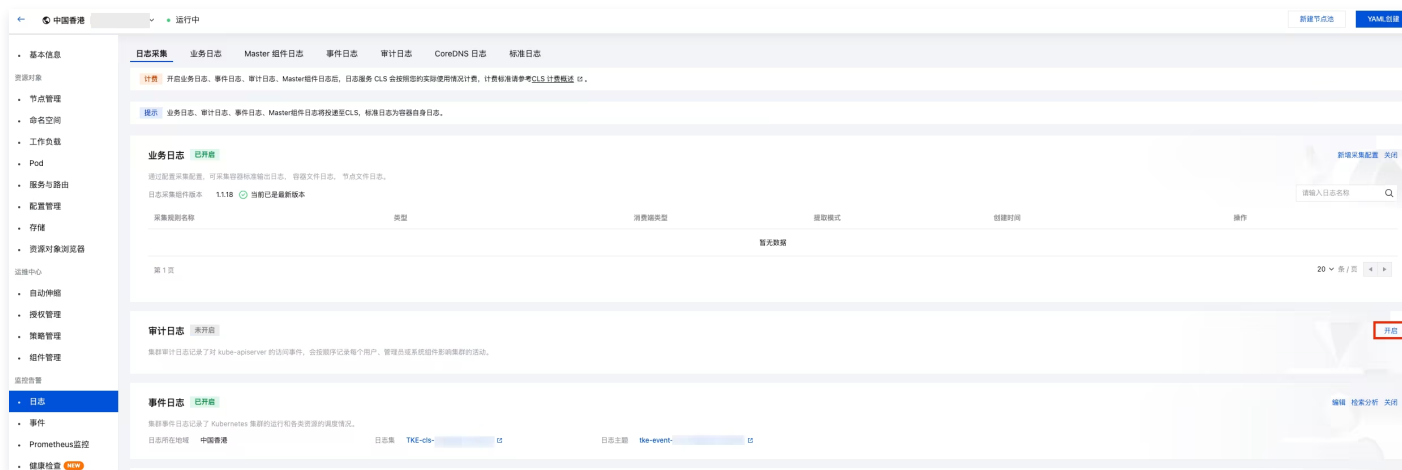
操作步骤

开启审计日志

注意：

- 开启审计日志功能需要重启 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。

- 登录 [容器服务控制台](#)，在左侧导航栏中选择**集群**。
- 在集群列表中，单击目标集群 ID，进入集群详情页。
- 选择左侧导航栏中的**日志**，在日志采集页签中单击**审计日志**右侧的**开启**。如下图所示：



- 在开启审计日志中，选择日志所在地域，并配置日志集和日志主题。如下图所示：



- 单击**确定**，即可开启审计日志。

审计仪表盘

最近更新时间：2025-09-03 17:11:12

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

审计日志中配置了五个大盘，分别是审计总览、节点操作概览、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日志页面，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，操作详情请参见 [开启审计日志](#)。
3. 选择集群 ID > 日志 > 审计日志。
4. 在全局检索页签查看集群审计详情。

审计总览

当您想观测整个集群 API Server 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。如下图所示：

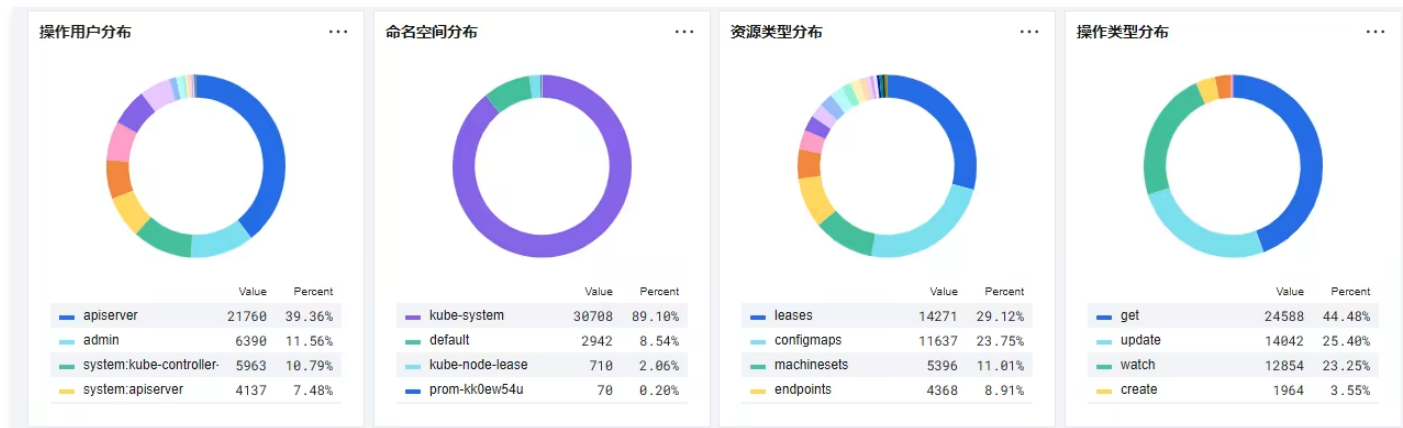


您还可在该页面中查看更多统计信息，如下所示：

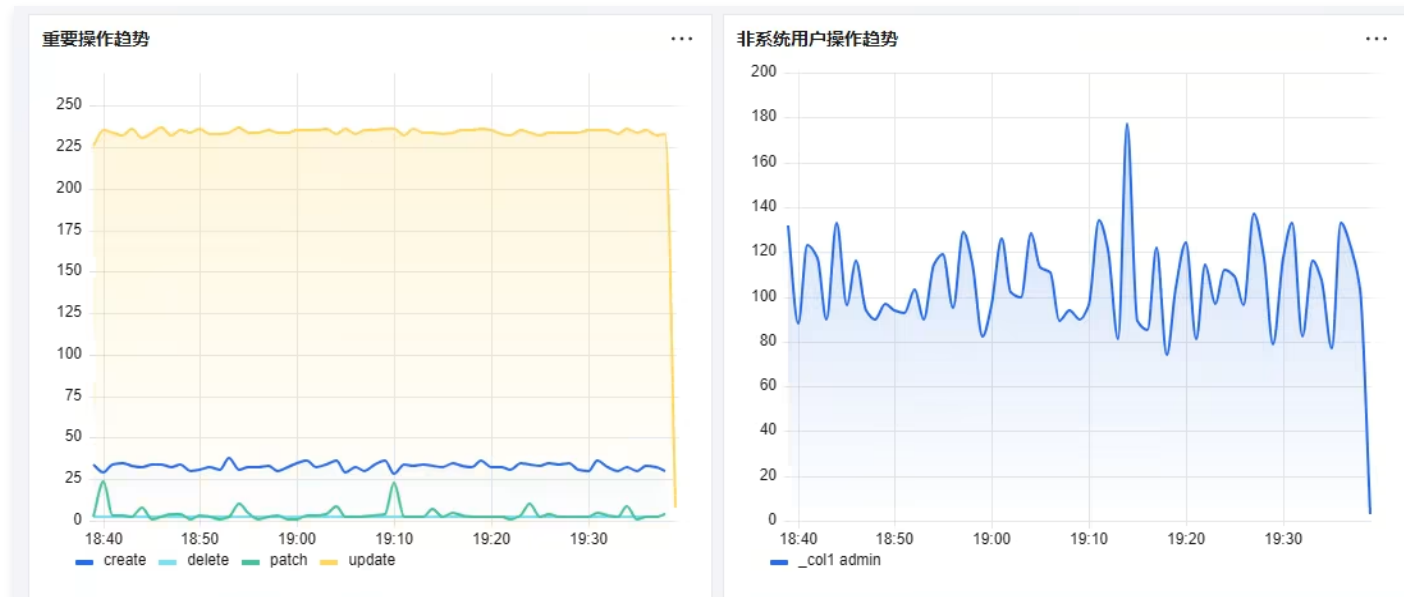
- 核心审计日志的统计数仪表盘：



- 分布情况仪表盘：

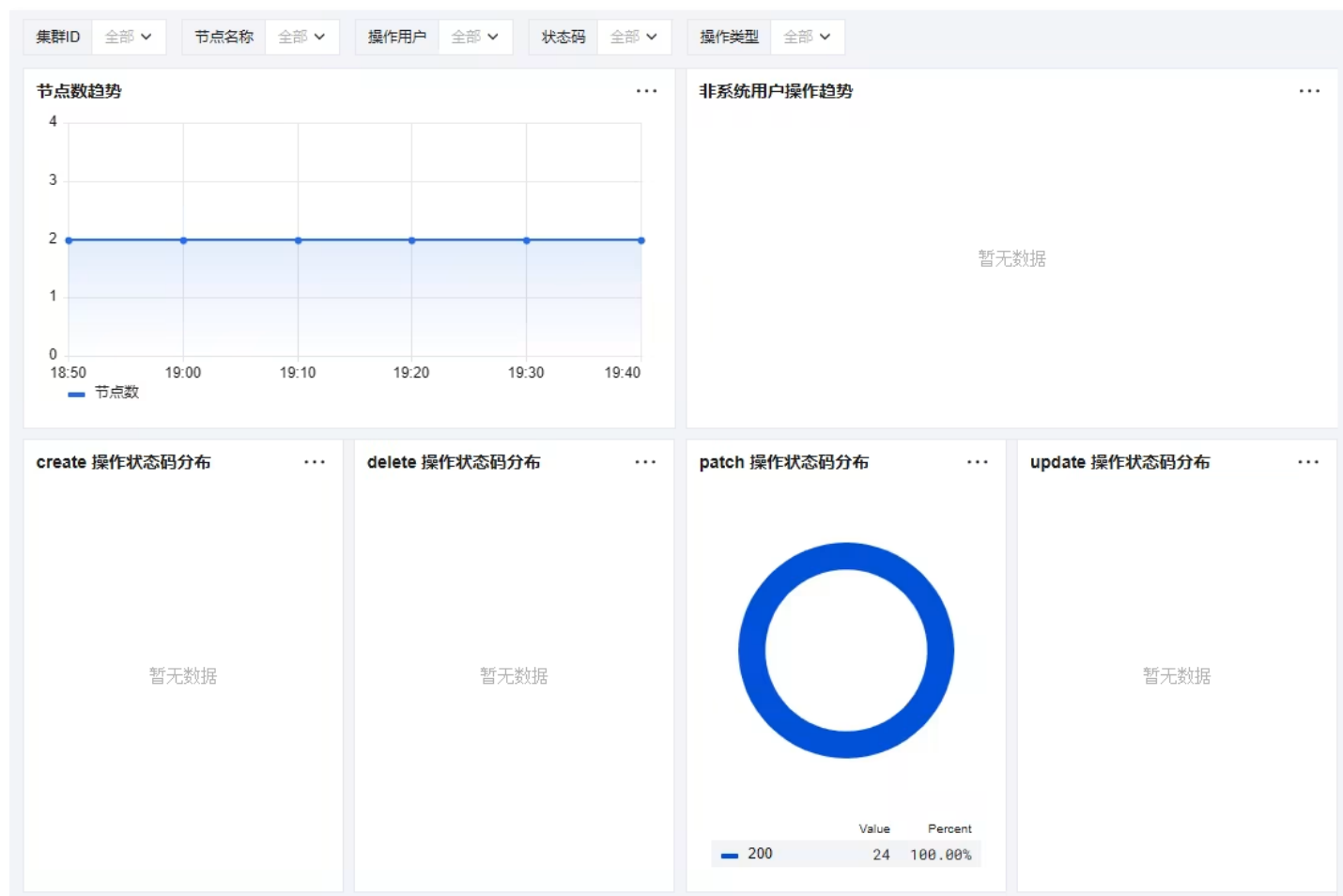


● 重要操作趋势仪表盘：



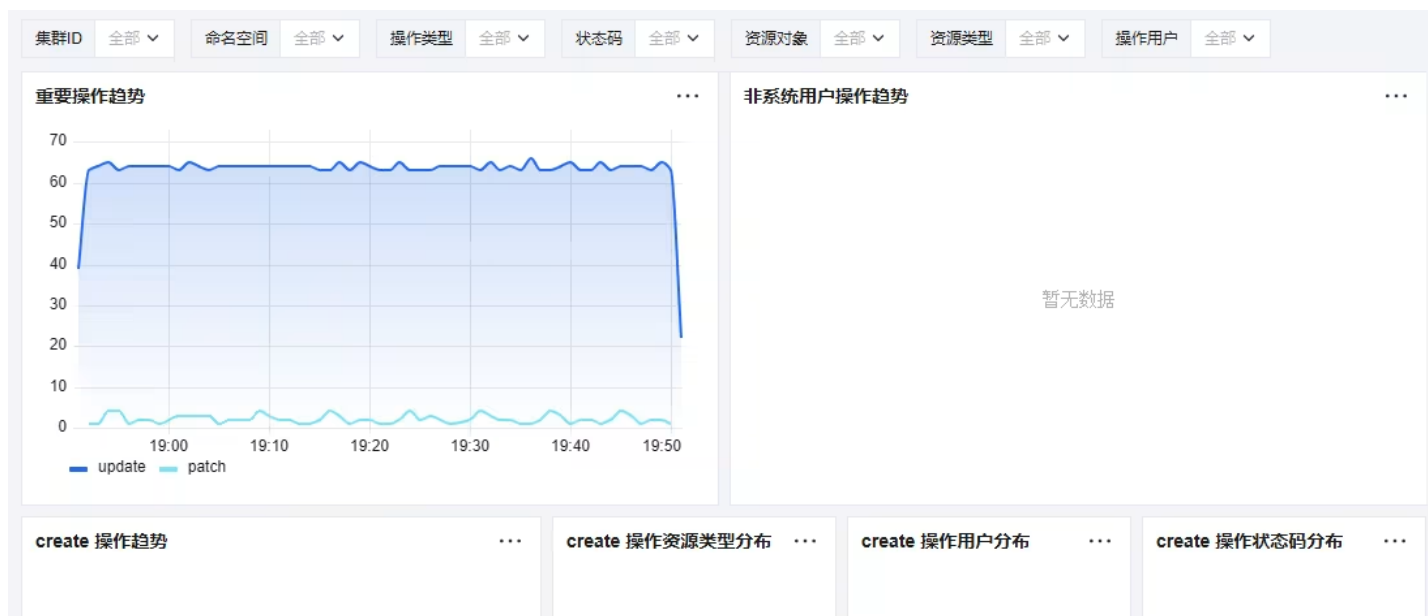
节点操作概览

当您需要排查节点相关问题时，可在节点操作概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：



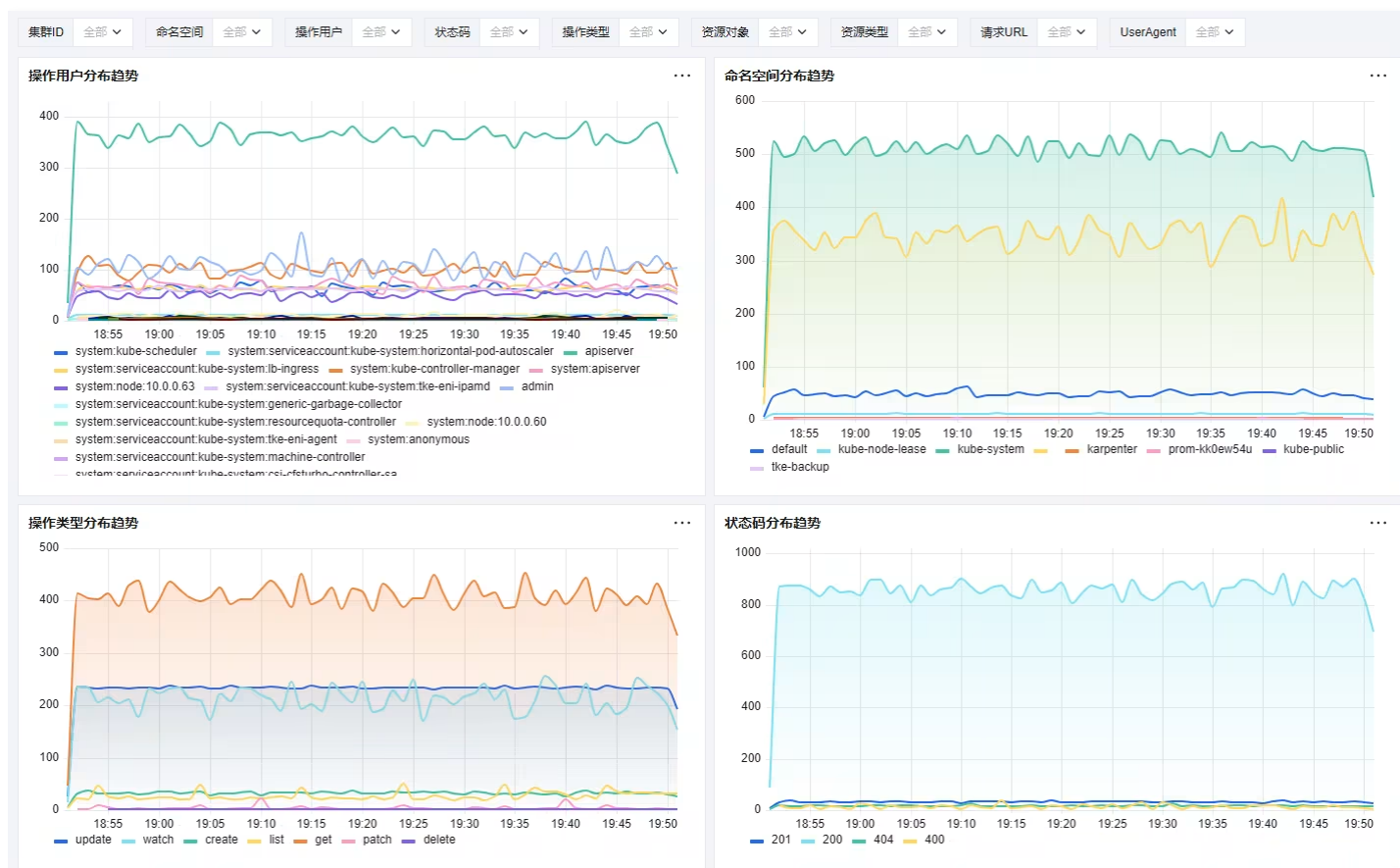
K8S 对象操作概览

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。维度包括操作用户、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

全局检索仪表盘内嵌了日志服务 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：



1. 单击需要配置告警的仪表盘右侧的**添加到监控告警** 。

2. 在 [日志服务控制台](#) 中，选择**监控告警**>**告警策略**，新建告警策略。详情可参见 [配置告警策略](#)。

事件管理

事件日志

最近更新时间：2025-02-21 10:23:23

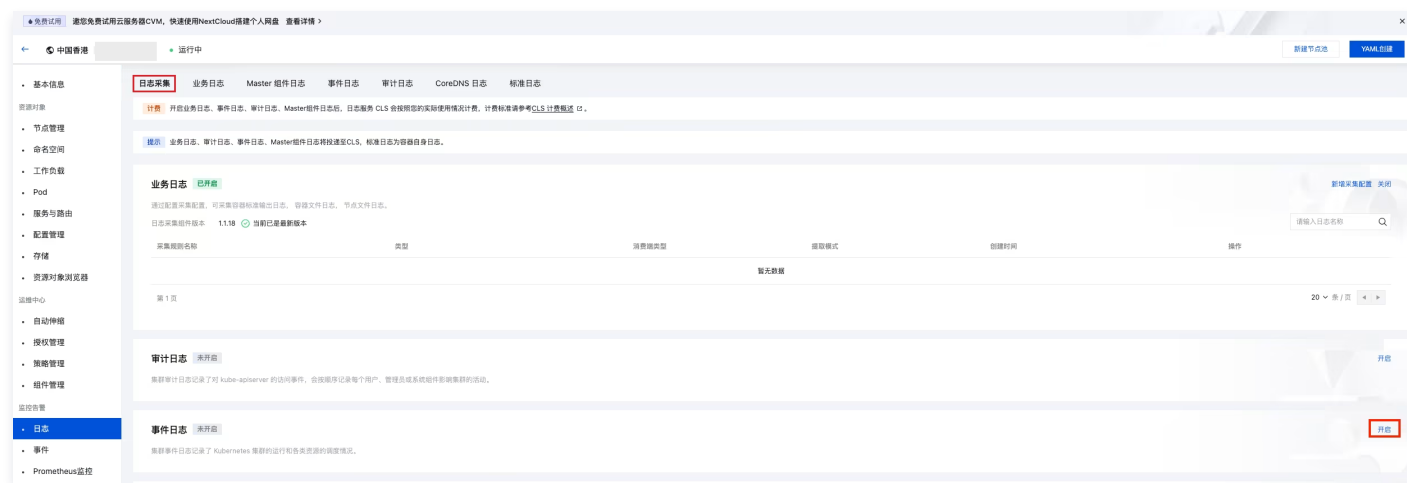
操作场景

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，集群事件会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

开启事件日志

1. 登录 [容器服务控制台](#)，在左侧导航栏中选择**集群**。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航栏中的**日志**，在日志采集页签中单击事件日志右侧的**开启**。如下图所示：



4. 在**开启事件日志**对话框中，选择日志所在地域，并配置日志集和日志主题。如下图所示：



- **日志所在地域**：CLS 支持跨地域日志投递，您可通过单击修改选择日志投递的目标地域。
- **日志集**：根据日志所在地域展示您已经创建的日志集，若现有日志集不合适，您可以在 [日志服务控制台](#) 新建日志集。操作详情见 [创建日志集](#)。

注意：
每个日志集下最多可以创建500个日志主题及指标主题。

- **日志主题**：选择日志集下对应的日志主题，支持“自动创建”和“选择已有”日志主题两种模式。

5. 单击**确定**，即可开启事件日志。

更新日志集或日志主题

1. 登录 [容器服务控制台](#)，在左侧导航栏中选择**集群**。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航栏中的**日志**，在日志采集页签中单击事件日志右侧的**编辑**。
4. 在编辑事件日志中，重新选择日志集和日志主题。如下图所示：

编辑事件日志

计费

开启事件存储后，日志服务CLS会按照您的实际使用情况计费，计费标准请参考[CLS计费概述](#)

日志所在地域

中国香港

修改

日志集

自动创建日志集

选择已有日志集

TKE-clr-

日志集不合适? [点击新建](#)

日志主题

自动创建日志主题

选择已有日志主题

tke-event-clr-

日志主题不合适? [点击新建](#)

一个日志主题目前仅支持配置一类日志，即日志、审计、事件不可采用同一个topic，会产生覆盖情况，请保证选用该日志主题没有被其他采集配置占用。

确定

取消

5. 单击**确定**即可更新日志集和日志主题。

关闭事件日志

1. 登录 [容器服务控制台](#)，在左侧导航栏中选择**集群**。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航栏中的**日志**，在日志采集页签中单击事件日志右侧的**关闭**。如下图所示：

运行中

新建节点池

VPA配置

基本信息

资源可查

节点管理

命名空间

工作负载

Pod

服务与路由

配置管理

存储

资源对象浏览器

运维中心

自动伸缩

授权管理

策略管理

组件管理

应用管理

日志

事件

Prometheus监控

日志采集

业务日志

Master 组件日志

事件日志

审计日志

CoreDNS 日志

标准日志

计费

开启业务日志、事件日志、审计日志、Master组件日志将按至CLS，计费标准请参考[CLS 计费概述](#)。

提示

业务日志、审计日志、事件日志、Master组件日志将按至CLS，标准日志为容器自身日志。

业务日志

已开启

新增采集配置

关闭

通过配置采集配置，可采集容器标准输出日志，容器文件日志、节点文件日志。

日志采集组件版本

1.1.18

当前已是最新版本

请输入日志名称

Q

采集规则名称	类型	消费端类型	提取模式	创建时间	操作
暂无数据					

第 1 页

20 / 条 / 页

审计日志

未开启

开启

集群审计日志记录了对 kube-apiserver 的访问事件，会按照身份记录每个用户、管理员或系统组件调用集群的活动。

事件日志

已开启

编辑

检索分析

关闭

集群事件日志记录了 Kubernetes 集群的运行和各类资源的调度情况。

日志所在地域

中国香港

日志集

TKE-clr-

日志主题

tke-event-clr-

确定

4. 在关闭事件日志页面，单击**确定**，即可关闭事件日志。如果不再需要事件日志，可勾选删除日志数据。如下图所示：

关闭cls-

集群事件日志

×

❗ 关闭集群事件日志后，将停止采集新增的事件日志。存量日志将持续收取存储费用直至过期。

当前事件日志主题信息

日志所在地域: 中国香港

日志集名称/ID: TKE-cl-

日志主题名称/ID: tke-event-cl-

☐ 删除日志数据

删除由容器服务创建的日志集和日志主题，删除后数据不可恢复，请谨慎操作。

确定

取消

查看事件

1. 登录 容器服务控制台，在左侧导航栏中选择集群。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航栏中的日志，在日志采集页签中单击事件日志右侧的检索分析。如下图所示：

腾讯云容器服务控制台

中国香港

运行中

新建节点池

YAML 编辑

基本信息

日志采集

业务日志

Master 组件日志

事件日志

审计日志

CoreDNS 日志

标准日志

事件日志

已开启

通过配置采集配置，可采集容器标准输出日志、容器文件日志、节点文件日志。

日志采集组件版本: 1.11.7

关联: 业务日志、审计日志、事件日志、Master 组件日志将投递至CLS，标准日志为容器自身日志。

无数据

第 1 页

20 / 条 / 页

事件日志

已开启

集群事件日志记录了 Kubernetes 集群的运行和各类资源的调度情况。

日志集: TKE-cl-dao7nm4-102564

日志主题: tke-event-cl-dao7nm4-102564

检索分析

4. 在事件日志页面，单击全局检索，按需自行配置显示字段并进行检索分析。

腾讯云容器服务控制台

中国香港

运行中

新建节点池

YAML 编辑

事件日志

已开启

通过配置采集配置，可采集容器标准输出日志、容器文件日志、节点文件日志。

日志采集组件版本: 1.11.7

关联: 业务日志、审计日志、事件日志、Master 组件日志将投递至CLS，标准日志为容器自身日志。

无数据

第 1 页

20 / 条 / 页

事件日志

已开启

集群事件日志记录了 Kubernetes 集群的运行和各类资源的调度情况。

日志集: TKE-cl-dao7nm4-102564

日志主题: tke-event-cl-dao7nm4-102564

检索分析

① 说明:

当您开启事件日志时，将默认为您的 Topic 开启索引。详情请参见 [日志检索分析](#)。

事件仪表盘

最近更新时间：2025-09-05 16:52:52

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。在集群开启事件日志功能后，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“事件日志功能”，详情请参见 [事件日志](#)。
3. 选择集群ID > 日志 > 事件日志。
4. 在全局检索页签查看集群事件详情。

事件总览

在事件日志 > 事件总览页面，可根据集群 ID、命名空间、级别、原因、资源类型、资源对象、事件源等维度过滤事件，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。如下图所示：

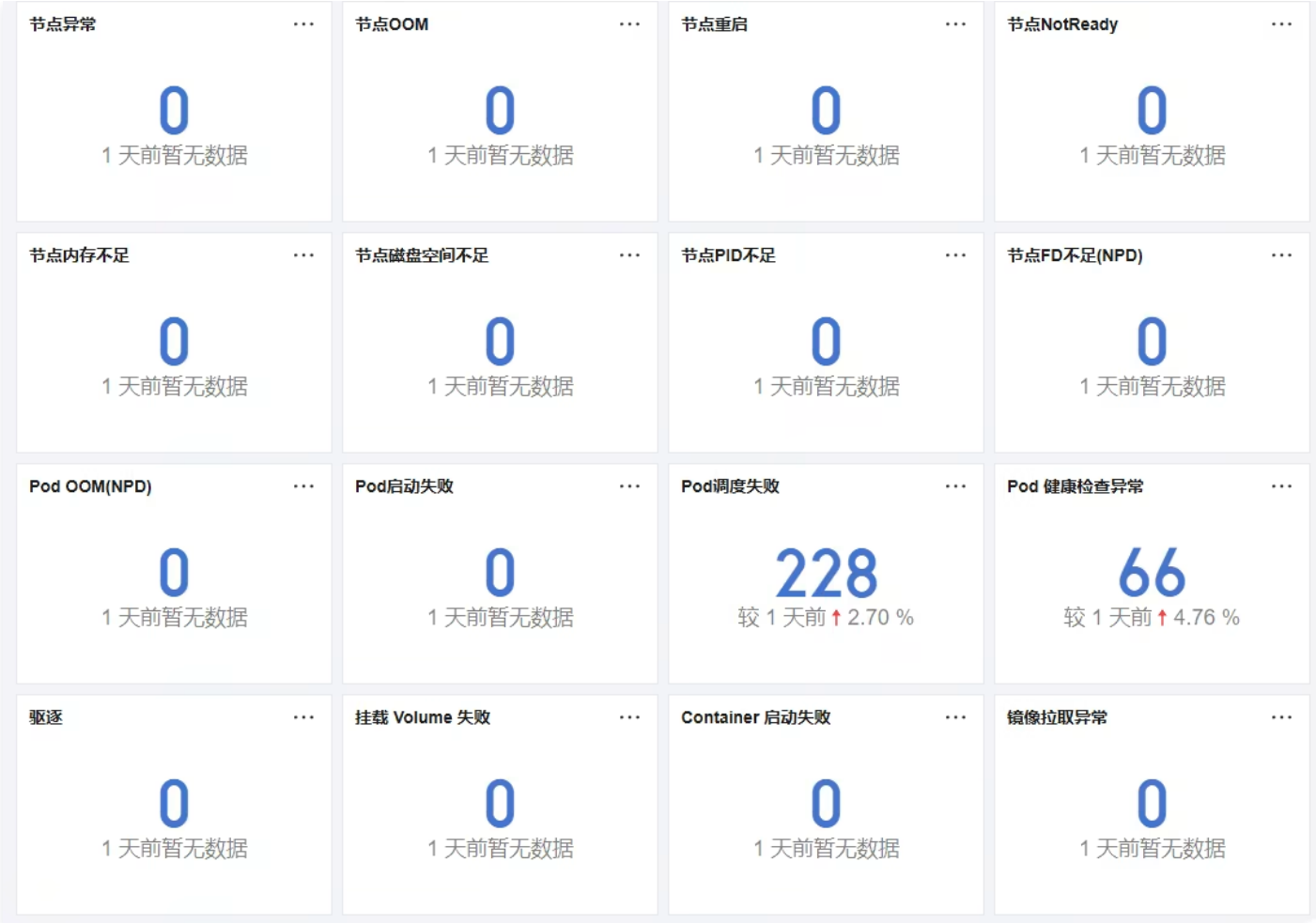


您还可在该页面中查看更多统计信息，如下所示：

- 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：



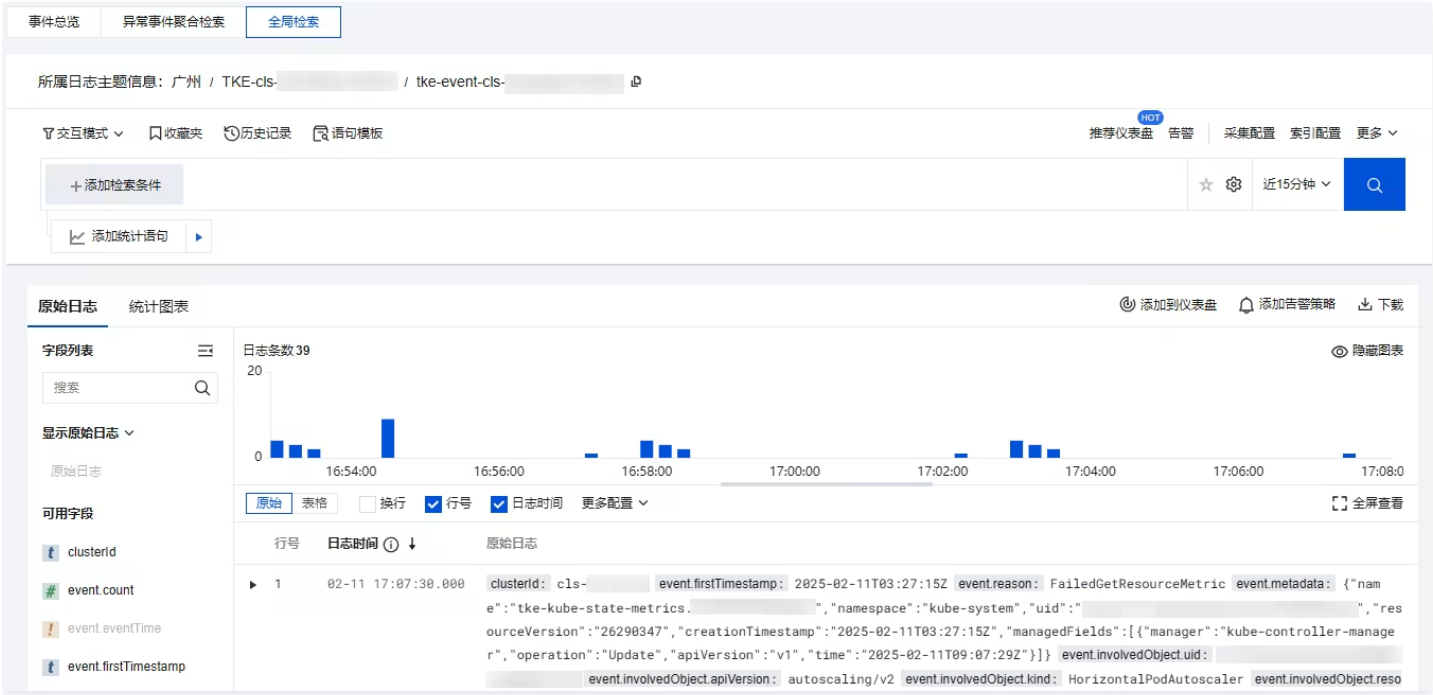
异常事件聚合检索

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

全局检索仪表盘内嵌了日志服务 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添加到监控告警。如下图所示：



2. 在 [日志服务控制台](#) 中，选择[监控告警](#) > [告警策略](#)，新建告警策略。详情请参见 [配置告警策略](#)。

监控管理

基础监控与告警

监控告警概述

最近更新时间：2023-05-10 17:09:56

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 [腾讯云可观测平台](#) 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 [Prometheus 监控](#) 服务。Prometheus 监控致力于提供轻量、稳定、高可用的服务。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。具体操作请参见 [腾讯云 Prometheus 一键关联监控容器服务](#)。

查看监控数据

最近更新時間：2025-04-17 17:36:02

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。

前提条件

已登录容器服务控制台，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：

名称ID	监控	状态	集群类型	kubernetes版本	节点数	创建时间	资源量	腾讯云标签	操作
管理控制台		运行中	标准集群	1.30.0	0台		CPU - 内存 -	-	新建节点 查看集群凭证 更多
管理控制台		运行中	标准集群	1.30.0	6台 (全部正常)		CPU - 内存 -	-	新建节点 查看集群凭证 更多
管理控制台		运行中	标准集群	1.22.5	2台 (全部正常)		CPU 4.55/10% 10核 内存 4.11/97% 20GB	-	新建节点 查看集群凭证 更多

查看节点指标

您可以通过以下操作查看节点的监控信息。

- 选择集群 ID，进入该集群的基本信息页。
- 在 **节点管理** 页面，选择 **节点** 页签，单击 **监控** 即可查看节点的监控信息。如下图所示：

节点名称	节点类型	状态	可用区	节点版本	运行时间	配置	Labels	IP地址	已分配资源	所属节点池	计费模式	操作
node-1	标准节点	运行中	北京三区	v1.26.1-k8s-1	containerd/1.6.9-k8s-4	SA2.MEDIUM4 2核, 4GB, 100GB 系统盘: 100GB	beta.kubernetes.io/arch:amd64 beta.kubernetes.io/instance-type:SA2.MEDIUM4 beta.kubernetes.io/os:linux	-	CPU 1.16 / 1.90 核 内存 1.40 / 2.49 Gi EKS-IP: 12.127	标准节点池	按量计费	详情 更多

查看节点内 Pod 指标

- 选择集群 ID，进入该集群的基本信息页。
- 在 **节点管理** 页面，选择 **节点**，进入节点详情页。
- 在 **Pod 管理** 页签中单击 **监控**，即可查看到该节点内 Pod 的监控指标曲线图。如下图所示：



查看工作负载指标

1. 选择集群 ID，进入该集群的基本信息页。
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

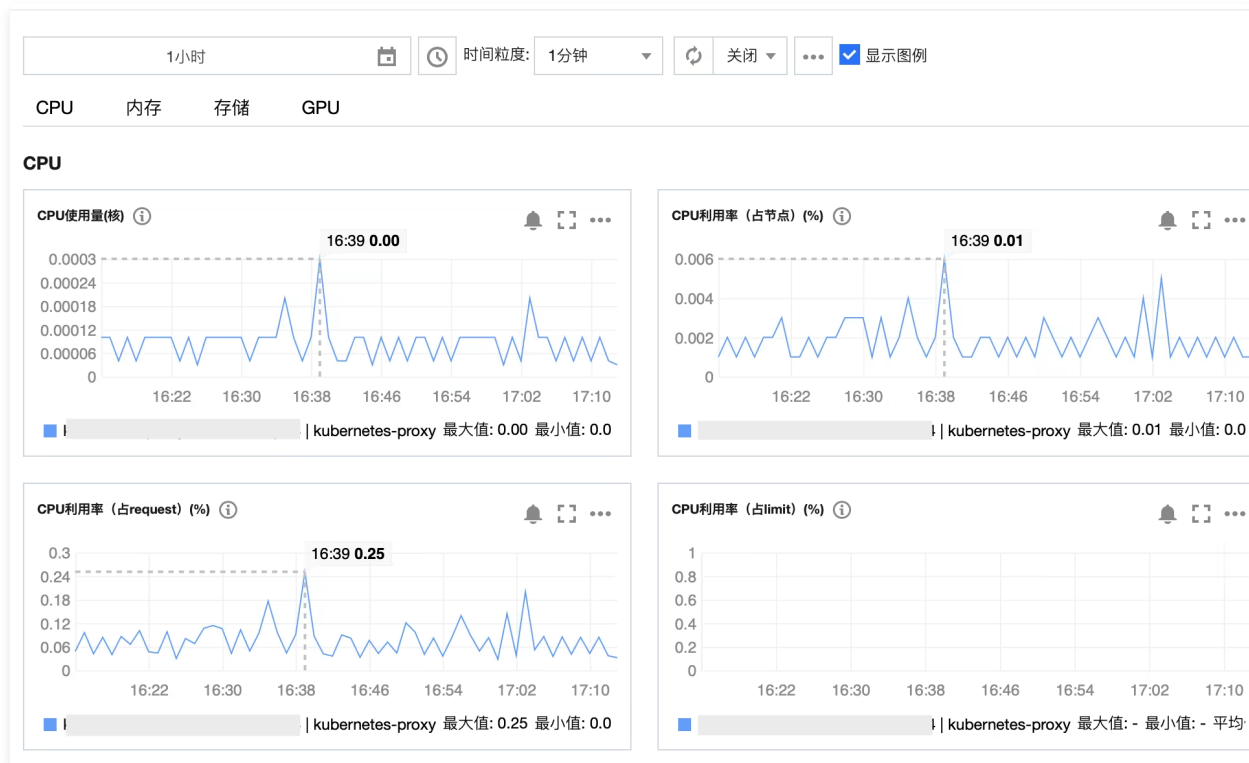
1. 选择需要查看的集群 ID，进入该集群的基本信息页。
2. 选择工作负载 > 任意类型工作负载。例如，选择 **Deployment**，进入 Deployment 管理页面。
3. 选择工作负载名称，在该工作负载的 **Pod 管理** 页签中单击**监控**，即可查看该工作负载内所有 Pod 的监控指标对比图。如下图所示：



查看 Pod 内 Container 指标

1. 选择 集群 ID，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. 选择工作负载名称，在该工作负载的 **Pod 管理** 页签中，单击实例名称前的▶，即可查看该 Pod 的 Container 信息。

4. 单击 ，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：2024-12-12 09:57:22

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
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 [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 [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
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 下设置了 Request 容器的 CPU 使用量之和与设置的 Request 值之和的比值（没设置 Request 的容器不参与计算）
CPU 利用率（占 Limit）	%	Pod 下设置了 Limit 容器的 CPU 使用量之和与设置的 Limit 之和的比值（没设置 Limit 的容器不参与计算）
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 下设置了 Request 容器的内存使用量之和与设置的 Request 值之和的比值（没设置 Request 的容器不参与计算）
内存利用率（占 Request，不包含 Cache）	%	Pod 下设置了 Request 容器的内存使用量（不含缓存）之和与设置的 Request 值之和的比值（没设置 Request 的容器不参与计算）
内存利用率（占 Request，working_set）	%	Pod 下设置了 Request 容器的工作集内存使用量之和与设置的 Request 值之和的比值（没设置 Request 的容器不参与计算）
内存利用率（占 Limit）	%	Pod 下设置了 Limit 容器的内存使用量之和与设置的 Limit 值之和的比值（没设置 Limit 的容器不参与计算）
内存利用率（占 Limit，不包含 Cache）	%	Pod 下设置了 Limit 容器的内存使用量（不含缓存）之和与设置的 Limit 值之和的比值（没设置 Request 的容器不参与计算）
内存利用率（占 Limit，working_set）	%	Pod 下设置了 Limit 容器的工作集内存使用量之和与设置的 Limit 值之和的比值（没设置 Limit 的容器不参与计算）
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
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
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

控制面组件监控

查看 TKE 集群控制面组件监控

最近更新时间：2025-02-27 17:55:12

使用场景

腾讯云容器服务 TKE 是一种托管的 Kubernetes 服务，TKE 自动管理 Kubernetes 控制平面节点的可用性和可扩展性，这些节点负责调度容器、管理应用程序可用性、存储集群数据和其他关键任务。但某些工作负载行为或配置可能会增加控制平面的负载，从而导致性能下降。在这些情况下，访问关键控制面指标和监控图表使集群管理员能够快速检测和排除集群上运行的工作负载的问题。例如，资源不足的工作节点可能会妨碍调度程序调度新 Pod 的能力。为了及时检测这些新出现的调度问题，集群管理员需要访问调度程序指标以查看待处理的 Pod 并及时收到通知。此外，控制面指标数量庞大，即使是经验丰富的管理员也面临选择最佳监控指标和创建有效监控图表的挑战。

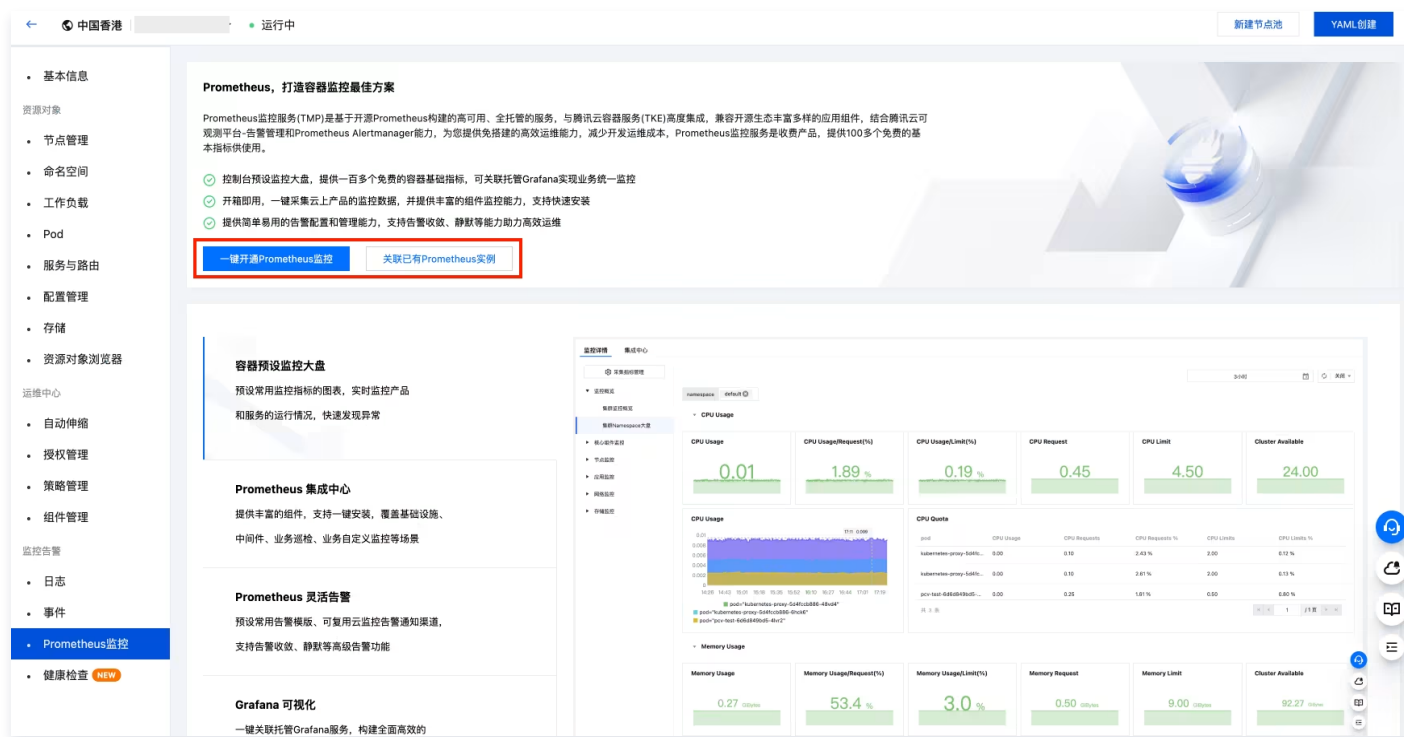
TKE 提供了集群控制面组件的监控能力，使集群管理员能够查看 Kubernetes 集群控制面的性能，快速检测、排除故障并修复问题。TKE 集群通过 Prometheus 监控提供监控图表。

前提条件

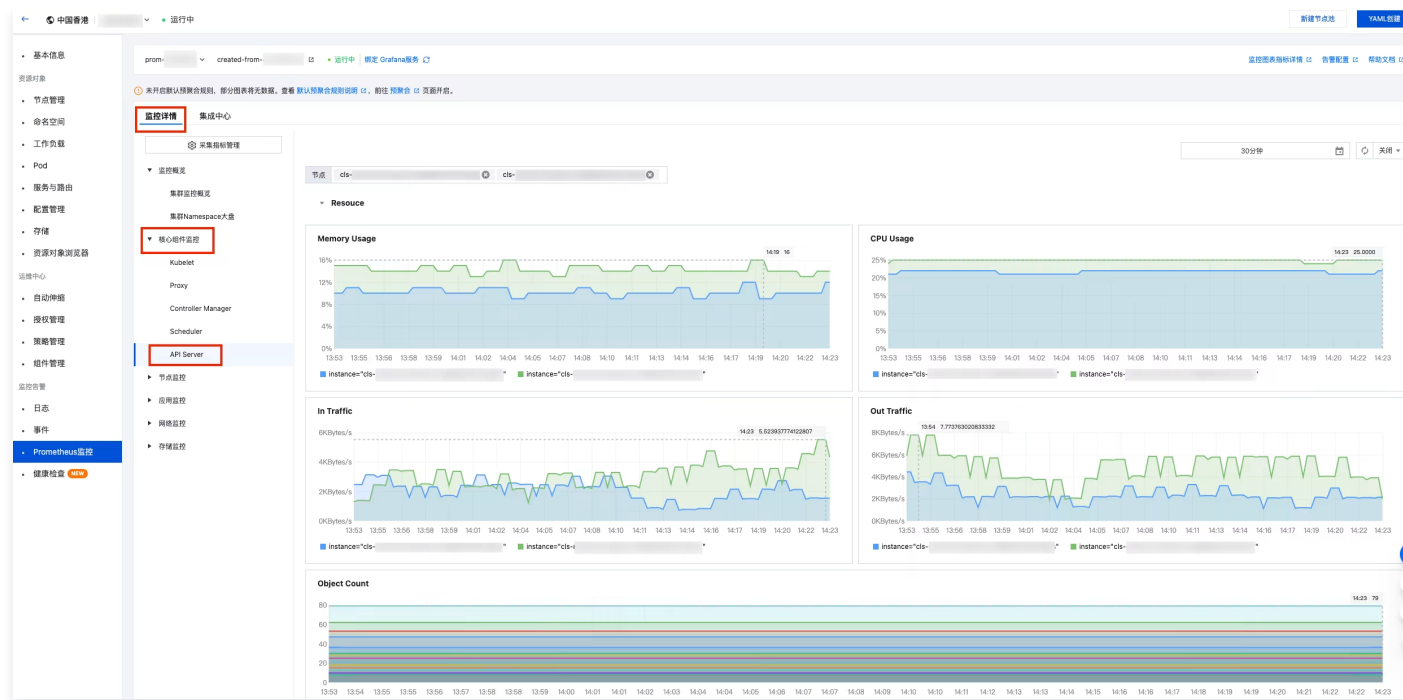
- 已将监控组件 [clustermonitor](#) 升级到 1.3.2 版本以上。
- 已开通 Prometheus 监控服务并关联实例。操作详情请参见 [腾讯云 Prometheus 一键关联监控容器服务](#)。

操作步骤

- 登录 [容器服务控制台](#)，选择左侧导航栏中的**集群**。
- 在**集群管理**页面，单击目标集群ID，进入集群详情页。
- 选择左侧导航中的 **Prometheus 监控**。
- 可通过**一键开通 Prometheus 监控**、**关联已有 Prometheus 实例**开启 Prometheus 监控。



- 在 Prometheus 监控页面中，单击监控详情，单击**核心组件监控**。



在该页面，您可以查看 API Server、Kube Scheduler、Controller Manager 监控大盘。

相关文档

- kube-apiserver 组件：指标列表、对应监控大盘的使用说明，请参见 [kube-apiserver 组件指标说明](#)。
- kube-scheduler 组件：指标列表、对应监控大盘的使用说明，请参见 [kube-scheduler 组件指标说明](#)。
- kube-controller-manager 组件：指标列表、对应监控大盘的使用说明，请参见 [kube-controller-manager 组件指标说明](#)。
- 如何基于用户自建的 Prometheus，采集 TKE 集群的控制面组件监控 API Server、Scheduler、KCM 指标配置说明及推荐的告警配置，请参见 [用户自建 Prometheus 采集控制面监控和告警](#)。

kube-apiserver 组件指标说明

最近更新时间：2025-04-23 15:05:02

前提条件

请参见 [查看 TKE 集群控制面组件监控](#)。

指标列表

功能分类	指标	类型	说明
请求概览	apiserver_request_duration_seconds_bucket	Histogram	该指标用于统计 APIServer 客户端对 API Server 不同请求的时延分布。 请求的维度包括： • Verb：请求的类型，例如 GET、POST、PUT、DELETE 等。 • Group：API 组，即相关 API 接口的集合，用于扩展 Kubernetes API。 • Version：API 版本，例如v1、v1beta1等。 • Resource：请求针对的资源类型，例如 Pod、Service、Lease 等。 • Subresource：资源的子资源，例如 Pod 详细信息、Pod 日志等。 • Scope：请求的范围，例如命名空间维度的资源（Namespace-scope）或集群维度的资源（Cluster-scope）。 • Component：发起请求的组件的名称，例如 kube-controller-manager、kube-scheduler、cloud-controller-manager 等。 • Client：发起请求的客户端，可能是内部组件或外部服务。 API ServerHistogram 的 Bucket 阈值为{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.25, 1.5, 1.75, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 40, 50, 60}。单位：秒。
请求概览	apiserver_request_total	Counter	对 API Server 不同请求的计数。请求的维度包括 Verb、Group、Version、Resource、Scope、Component、HTTP contentType、HTTP code（响应的HTTP状态码）和 Client。
资源请求	apiserver_request_count	Counter	API Server 接收到的请求总数。
资源分析	apiserver_storage_objects	Gauge	资源对象数量。
请求概览	apiserver_watch_events_total	Counter	资源对象总量。按照资源类型分类，例如 nodes，pods 等。
请求概览	apiserver_current_inflight_requests	Gauge	API Server 当前处理的请求数量。请求包括两种： • ReadOnly：这类请求不会改变集群的状态，通常为读取资源的操作，例如获取 Pods 列表、查询节点状态等。 • Mutating：这类请求会改变集群的状态，通常为创建、更新或删除资源的操作，例如新建 Pod、更新 Service 配置等。
资源分析	pod_core_usage	Gauge	CPU 使用率。单位：%。
资源分析	pod_mem_usage	Gauge	内存使用率。单位：%。
资源分析	container_network_receive_bytes_total	Counter	网络入流量。单位：Bit。 说明：原指标 k8s_pod_network_receive_bytes_bw 需收费，因此进行指标名称映射为免费指标。
资源分析	container_network_transmit_packets_total	Counter	网络出流量。单位：Bit。

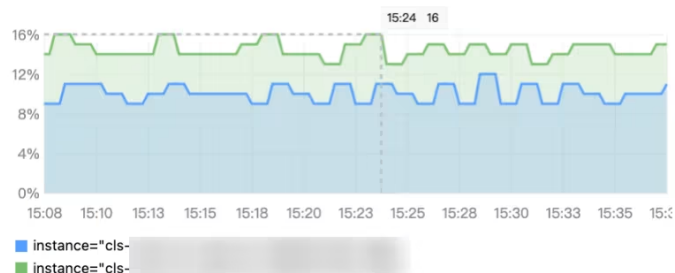
说明：原指标 k8s_pod_network_transmit_bytes_bw 需收费，因此进行指标名称映射为免费指标。

资源分析

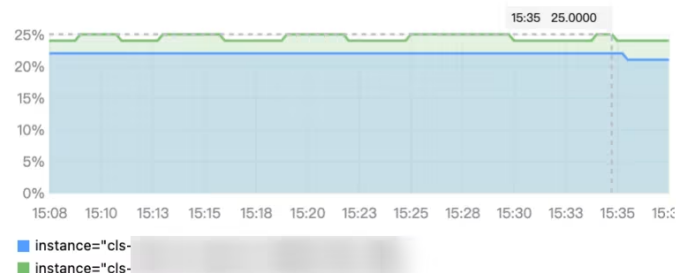
监控指标展示

Resource

Memory Usage



CPU Usage



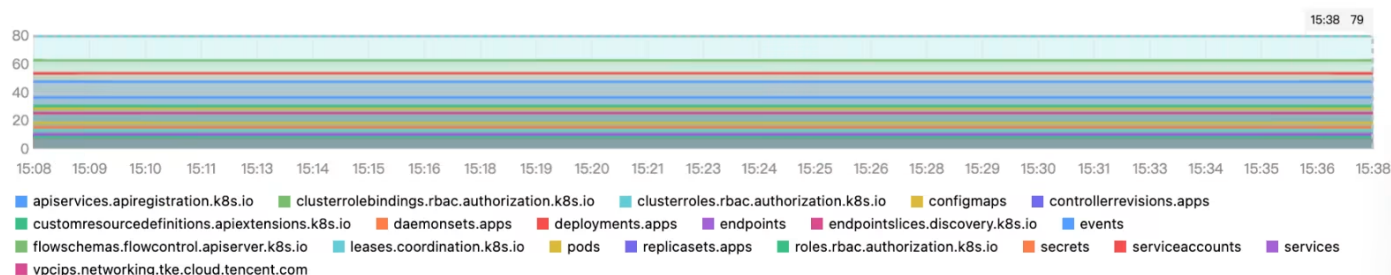
In Traffic



Out Traffic



Object Count



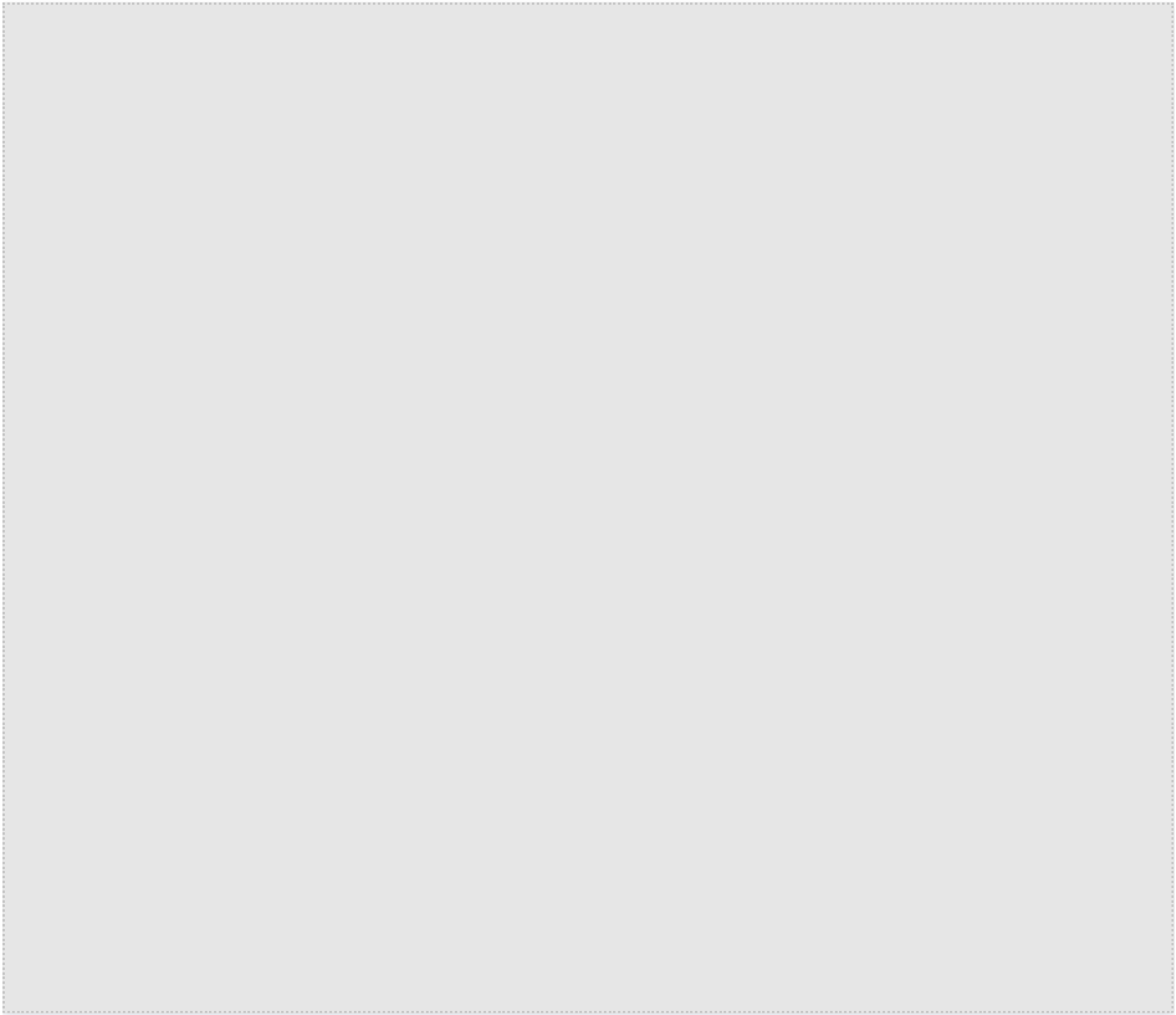
功能说明

指标	指标名称	使用指标	指标说明
Memory Usage	内存使用率	pod_mem_usage	API Server 的内存使用率。单位：%。
Cpu Usage	CPU 使用率	pod_core_usage	API Server 的 CPU 使用率。单位：%。
In Traffic	入流量	container_network_receive_bytes_total	API Server 的网络入流量。单位：Bit/s。
Out Traffic	出流量	container_network_transmit_packets_total	API Server 的网络出流量。单位：Bit/s。
Object Count	etcd 资源对象数	apiserver_storage_objects etcd_object_counts	资源对象数量。 当 TKE 为1.22及以上版本时，指标名字为 apiserver_storage_objects。

		当 TKE 为1.22及以下版本时，指标名字为 etcd_object_counts。 说明： 由于兼容性问题，1.22版本中 apiserver_storage_objects 名称和 etcd_object_counts 名称均存在。
--	--	--

请求概览

监控指标展示



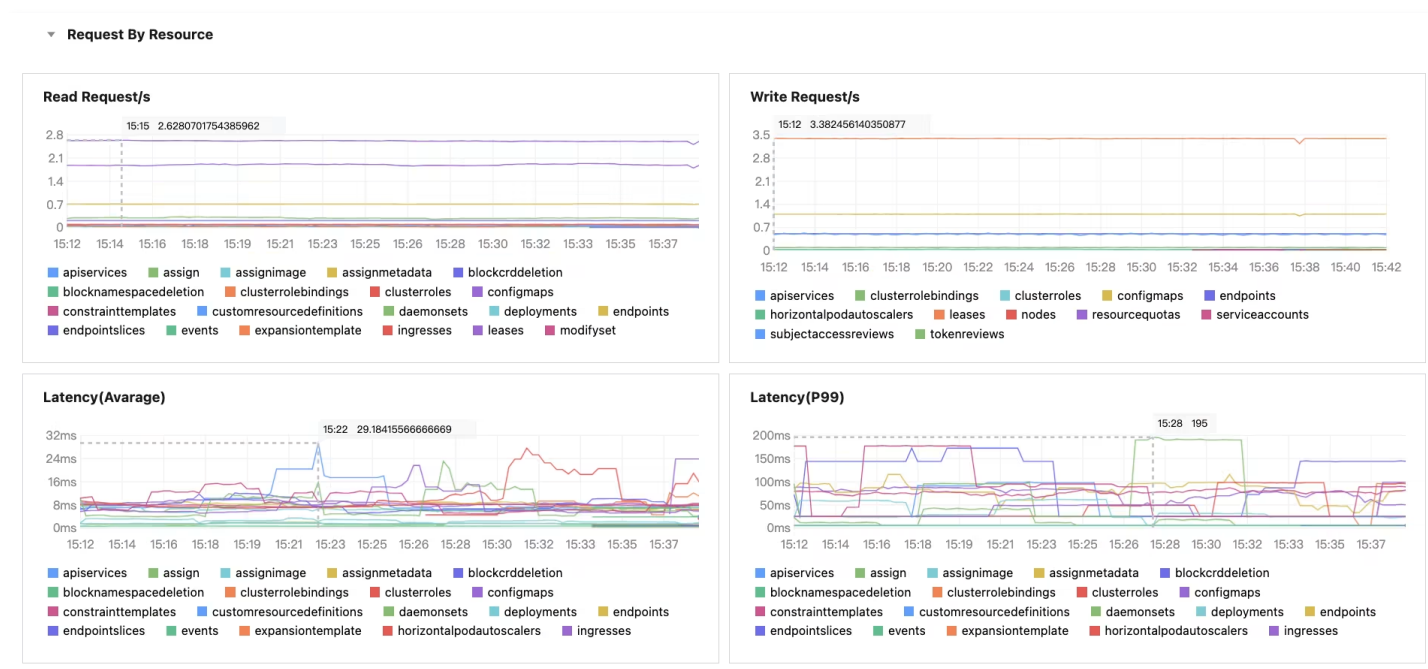
功能说明

指标	指标名称	使用指标	指标说明
Request/s	apiserver 每秒处理请求总量	apiserver_request_count apiserver_request_total	apiserver 每秒处理请求总量。单位：req/s。

Failed Request/s	apiserver 每秒返回失败请求数量	apiserver_request_count apiserver_request_total	apiserver 每秒返回失败请求数量。单位：req/s。
Write Request/s	apiserver 每秒写请求数量	apiserver_request_count apiserver_request_total	apiserver 每秒写请求数量。单位：req/s。
Read Request/s	apiserver 每秒读请求数量	apiserver_request_count apiserver_request_total	apiserver 每秒读请求数量。单位：req/s。
Latency(Average)	apiserver 访问平均延时	apiserver_request_latencies_summary_sum apiserver_request_latencies_summary_count	apiserver访问平均延时。单位：毫秒。
Latency(P99)	apiserver 请求时延 P99	apiserver_request_duration_seconds_bucket	统计APIServer客户端对API Server不同请求的时延分布。单位：毫秒。
Current Inflight Request	apiserver Inflight Request	apiserver_current_inflight_requests	API Server当前处理的请求数量。
Self Request/s	apiserver selfrequest QPS	apiserver_selfrequest_total	apiserver selfrequest QPS。
Response Body Size(P99)	apiserver 回包大小 P99	apiserver_response_sizes_bucket	apiserver回包大小P99。单位：Bytes。
Watch Events/s	apiserver watch 事件推送 QPS	apiserver_watch_events_total	资源对象总量。按照资源类型分类，例如 nodes，pods 等。单位Count/s。
Too Many Objects Events/s	list_too_many_objects_events_total	list_too_many_objects_events_total	list_too_many_objects 事件数量。单位Count/s。
Too Old Objects Events/s	watch_too_old_objects_events_total	watch_too_old_objects_events_total	watch_too_old_objects 事件数量。单位Count/s。

资源请求

监控指标展示



功能说明

指标	指标名称	使用指标	指标说明
Read Request/s	apiserver 每秒读请求数量	apiserver_request_count apiserver_request_total	apiserver 每秒读请求数量（按资源分类）。单位：req/s。
Write Request/s	apiserver 每秒写请求数量	apiserver_request_count apiserver_request_total	apiserver 每秒写请求数量（按资源分类）。单位：req/s。
Latency(Average)	apiserver 访问平均延时	apiserver_request_duration_seconds_sum apiserver_request_duration_seconds_count	apiserver 访问平均延时（按资源分类）。单位：毫秒。
Latency(P99)	apiserver 请求时延 P99	apiserver_request_duration_seconds_bucket	apiserver 请求时延P99（按资源分类）。单位：毫秒。

kube-scheduler 组件指标说明

最近更新时间：2025-03-26 11:18:42

前提条件

请参见 [查看 TKE 集群控制面组件监控](#)。

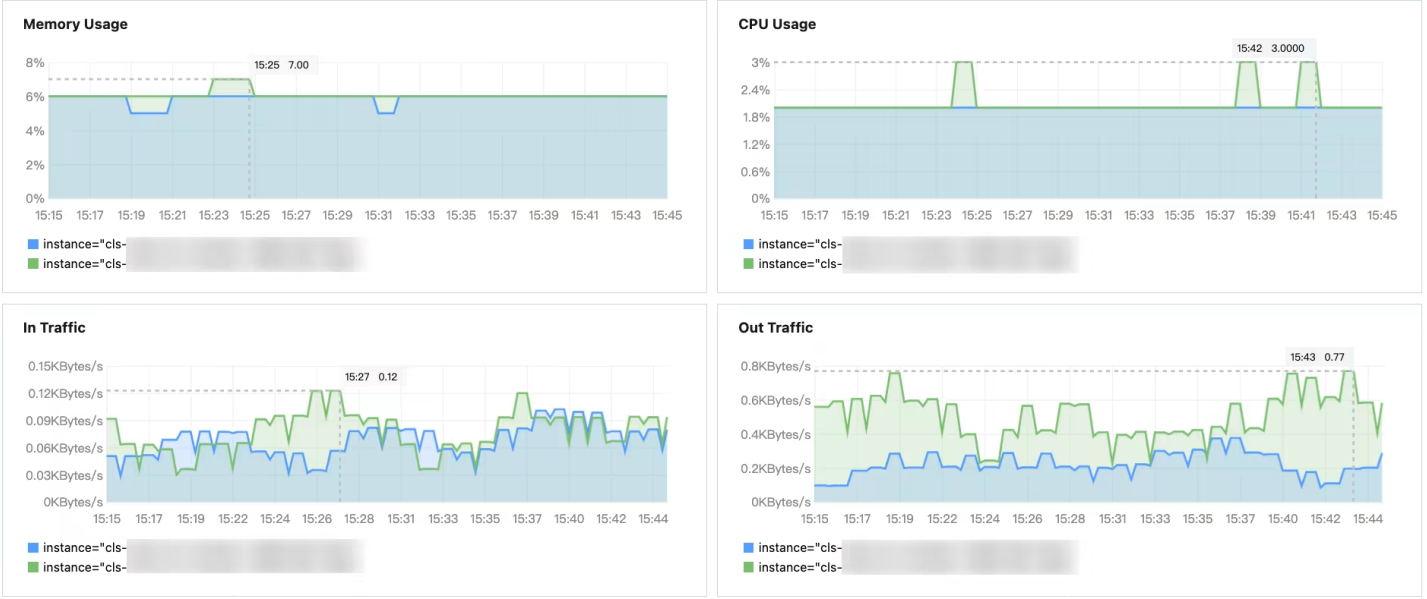
指标列表

功能分类	指标	类型	说明
请求分析	scheduler_pending_pods	Gauge	Pending Pod 的数量。队列种类如下： - unschedulable：表示不可调度的 Pod 数量。 - backoff：表示 backoffQ 的 Pod 数量。 - active：表示 activeQ 的 Pod 数量。
请求分析	scheduler_pod_scheduling_attempts_bucket	Histogram	调度器在尝试将 Pod 调度到节点时的调度尝试次数。bucket 阈值为{1、2、4、8、16}。
资源分析	pod_core_usage	Gauge	CPU 使用率。单位：%。
资源分析	pod_mem_usage	Gauge	内存使用率。单位：%。
请求分析	rest_client_requests_total	Counter	从状态值（Status Code）、方法（Method）和主机（Host）维度分析 HTTP 请求数。
请求分析	rest_client_request_duration_seconds_bucket	Histogram	从方法（Verb）和 URL 维度分析 HTTP 请求时延。bucket 阈值 {0.001,0.002,0.004,0.008,0.016,0.032,0.064,0.128,0.256,0.512}。单位：秒。
资源分析	container_network_receive_bytes_total	Counter	网络入流量。单位：Bit。 说明：原指标 k8s_pod_network_receive_bytes_bw 需收费，因此进行指标名称映射为免费指标。
资源分析	container_network_transmit_packets_total	Counter	网络出流量。单位：Bit。 说明：原指标 k8s_pod_network_transmit_bytes_bw 需收费，因此进行指标名称映射为免费指标。

资源分析

监控指标展示

▼ Resource



功能说明

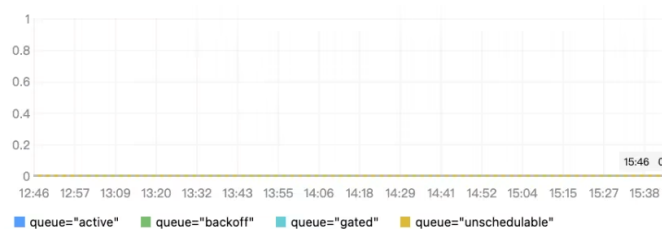
指标	指标名称	使用指标	指标说明
Memory Usage	内存使用率	pod_mem_usage	Scheduler 的内存使用率。单位：%。
Cpu Usage	CPU使用率	pod_core_usage	Scheduler 的 CPU 使用率。单位：%。
In Traffic	入流量	container_network_receive_bytes_total	Scheduler 的网络入流量。单位：Bit/s。
Out Traffic	出流量	container_network_transmit_packets_to tal	Scheduler 的网络出流量。单位：Bit/s。

请求分析

监控指标展示

Request

Pending Pods



Number of attempts to successfully schedule(P90)



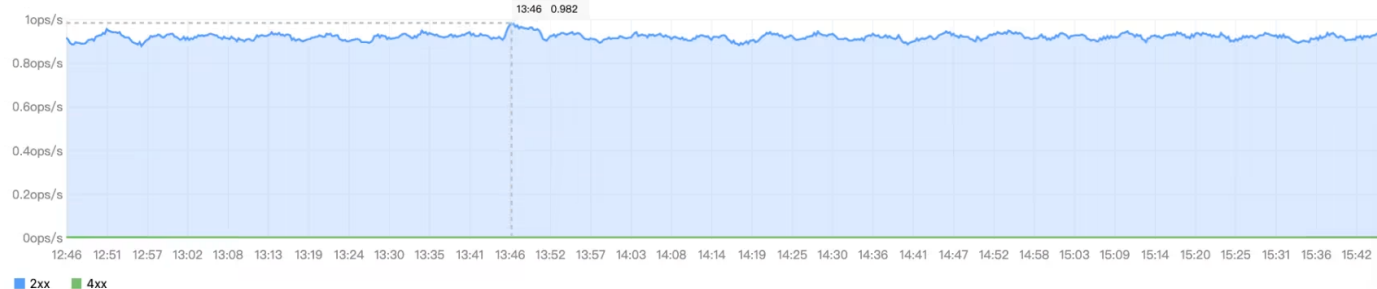
Get Request Latency 99th Quantile



Post Request Latency 99th Quantile



Kube API Request Rate



功能说明

指标	指标名称	使用指标	指标说明
Pending Pods	Scheduler Pending Pods	scheduler_pending_pods	Pending Pod 的数量。
Number of attempts to successfully schedule(P90)	Scheduler 尝试成功调度 Pod 次数	scheduler_pod_scheduling_attempts_bucket	调度器在尝试将 Pod 调度到节点时的调度尝试次数。
Get Request Latency 99th Quantile	Get 请求P99延时	rest_client_request_duration_seconds_bucket	Get 请求P99延时。单位：毫秒。
Post Request Latency 99th Quantile	Post 请求P99延时	rest_client_request_duration_seconds_bucket	Post 请求P99延时。单位：毫秒。
Kube API Request Rate	Kube API 请求 QPS	rest_client_requests_total	从状态值（Status Code）、方法（Method）和主机（Host）维度分析 HTTP 请求数。单位：ops/s

kube-controller-manager 组件指标说明

最近更新时间：2025-03-27 11:23:23

前提条件

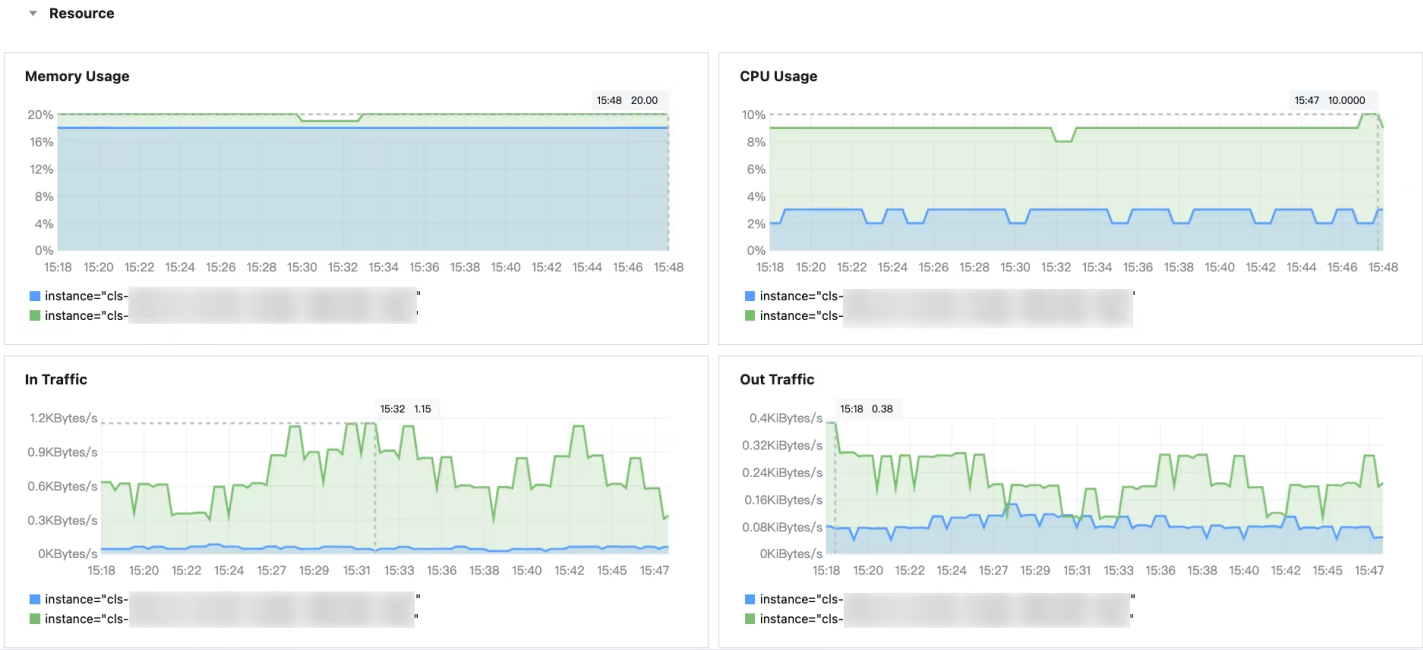
请参见 [查看 TKE 集群控制面组件监控](#)。

指标列表

功能分类	指标	类型	说明
请求分析	workqueue_adds_total	Counter	Workqueue 处理的新增事件（Adds）数量。
请求分析	workqueue_depth	Gauge	Workqueue 当前队列深度。如果队列深度长时间保持在较高水平，表明 Controller 不能及时处理队列中的任务，导致任务堆积。
请求分析	workqueue_queue_duration_seconds_bucket	Histogram	任务在 Workqueue 中存在的时长。Bucket 阈值为{10 ⁻⁸ , 10 ⁻⁷ , 10 ⁻⁶ , 10 ⁻⁵ , 10 ⁻⁴ , 10 ⁻³ , 10 ⁻² , 10 ⁻¹ , 1, 10}。单位：秒。
资源分析	pod_core_usage	Gauge	CPU 使用率。单位：%。
资源分析	pod_mem_usage	Gauge	内存使用率。单位：%。
请求分析	rest_client_requests_total	Counter	从状态值（Status Code）、方法（Method）和主机（Host）维度分析 HTTP 请求次数。
请求分析	rest_client_request_duration_seconds_bucket	Histogram	从方法（Verb）和 URL 维度分析 HTTP 请求时延。
资源分析	container_network_receive_bytes_total	Counter	网络入流量。单位：Bit。 说明：原指标 k8s_pod_network_receive_bytes_bw 需收费，因此进行指标名称映射为免费指标。
资源分析	container_network_transmit_packets_total	Counter	网络出流量。单位：Bit。 说明：原指标 k8s_pod_network_transmit_bytes_bw 需收费，因此进行指标名称映射为免费指标。

资源分析

监控指标展示

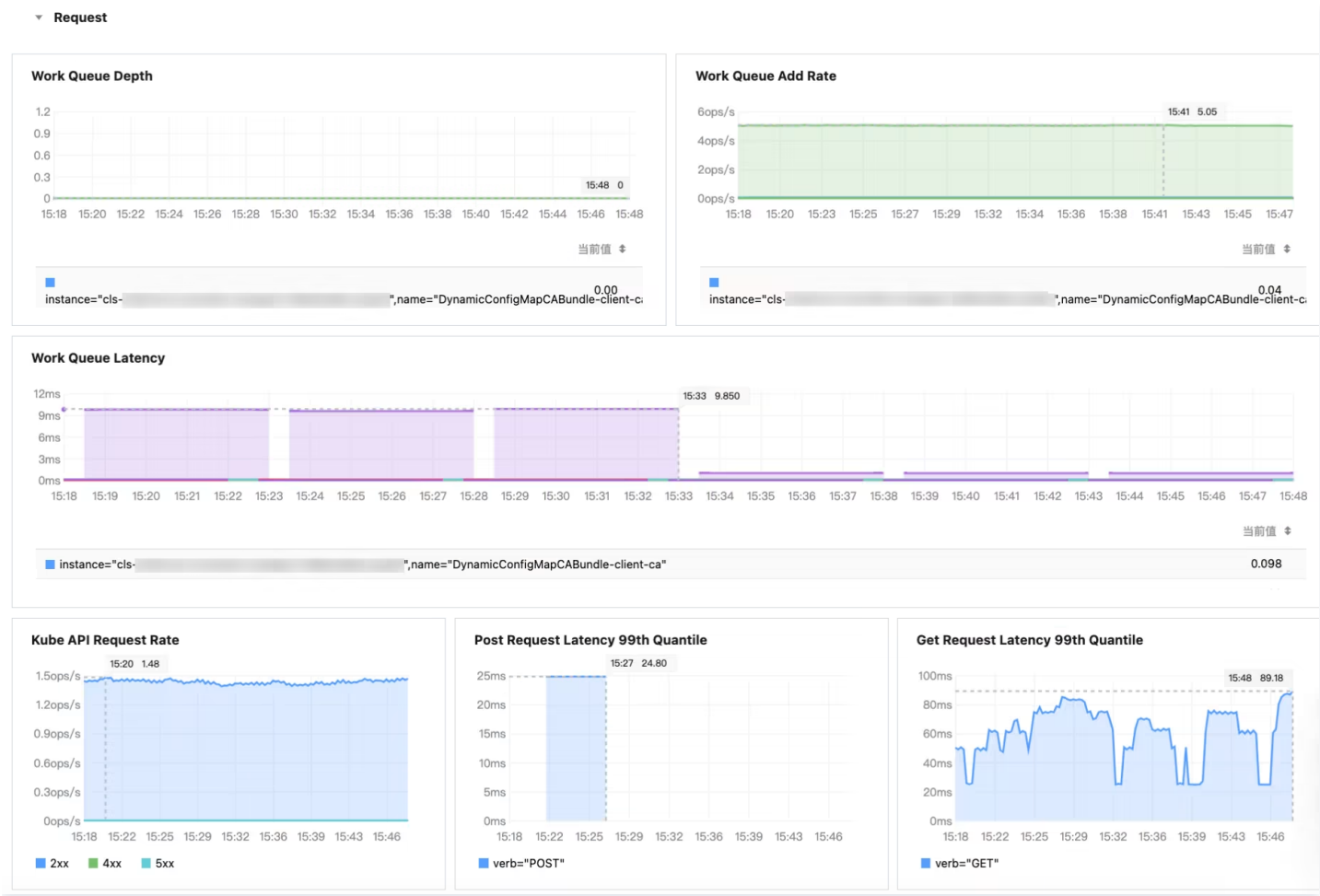


功能说明

指标	指标名称	使用指标	指标说明
Memory Usage	内存使用率	pod_mem_usage	Controller Manager 的内存使用率。单位：%。
Cpu Usage	CPU 使用率	pod_core_usage	Controller Manager 的 CPU 使用率。单位：%。
In Traffic	入流量	container_network_receive_bytes_total	Controller Manager 的网络入流量。单位：Bit/s。
Out Traffic	出流量	container_network_transmit_packets_total	Controller Manager 的网络出流量。单位：Bit/s。

请求分析

监控指标展示



功能说明

指标	指标名称	使用指标	指标说明
Work Queue Depth	Work Queue深度	workqueue_depth	Workqueue 深度在单位时间内的变化。
Work Queue Add Rate	Work Queue 增长率	workqueue_adds_total	Workqueue 在单位时间内新增事件（Adds）的数量。单位：ops/s。
Work Queue Latency	Work Queue 延迟	workqueue_queue_duration_seconds_bucket	事件在 Work Queue 中存在的时长。单位：毫秒。
Kube API Request Rate	Kube API 请求 QPS	rest_client_requests_total	从状态值（Status Code）、方法（Method）和主机（Host）维度分析 HTTP 请求数。单位：ops/s。
Post Request Latency 99th Quantile	Post 请求P99延时	rest_client_request_duration_seconds_bucket	Post 请求P99延时。单位：毫秒。
Get Request Latency 99th Quantile	Get 请求P99延时	rest_client_request_duration_seconds_bucket	Get 请求P99延时。单位：毫秒。

用户自建 Prometheus 采集控制面监控

最近更新时间：2025-09-04 10:51:42

本文介绍如何基于用户自建的 Prometheus，采集 TKE 托管集群的控制平面组件监控 APIServer、Scheduler、KCM 指标配置。

前提条件

- 自建的 Prometheus 能够访问 TKE 集群的 APIServer。详情请参见 [连接集群](#)。
- 自建的 Prometheus 可以部署在 TKE 集群内，也可以部署在 TKE 集群外。

Prometheus 采集配置

使用自建的 Prometheus 采集 TKE 集群控制面核心组件指标时，首先需要在 Prometheus 的配置文件 prometheus.yaml 中配置指标采集 Job。配置文件格式如下：

```
global:
  scrape_interval:      15s # By default, scrape targets every 15 seconds.

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: kube-apiserver
    .....

  - job_name: kube-controller-manager
    .....

  - job_name: kube-scheduler
    .....
```

其中，每个核心组件对应一个 Job 配置，具体配置可参见对应核心组件的指标列表。社区 Prometheus 配置 Prometheus.yaml 方法，请参见 [Configuration](#)。

TKE 集群内监控采集

内监控是将 Prometheus 部署在待监控的 TKE 集群内的监控形式。

kube-apiserver

关于 kube-apiserver 组件的更多信息，请参见 [kube-apiserver 组件指标说明](#)。

```
scrape_configs:
- job_name: kube-apiserver
  honor_timestamps: true
  params:
    component:
      - apiserver
  scrape_interval: 15s
  metrics_path: /master/metrics
  scheme: http
  follow_redirects: true
  enable_http2: true
  relabel_configs:
  - source_labels: [job]
    separator: ;
    regex: (.*)
```



```

target_label: __tmp_prometheus_job_name
replacement: $1
action: replace
- source_labels: [__meta_kubernetes_service_label_app_kubernetes_io_managed_by,
  __meta_kubernetes_service_labelpresent_app_kubernetes_io_managed_by]
  separator: ;
  regex: (Helm);true
  replacement: $1
  action: keep
- source_labels: [__meta_kubernetes_service_label_label_qcloud_app,
  __meta_kubernetes_service_labelpresent_label_qcloud_app]
  separator: ;
  regex: (cluster-monitor);true
  replacement: $1
  action: keep
- source_labels: [__meta_kubernetes_service_label_label_qcloud_service,
  __meta_kubernetes_service_labelpresent_label_qcloud_service]
  separator: ;
  regex: (master-metrics-service);true
  replacement: $1
  action: keep
- source_labels: [__meta_kubernetes_endpoint_port_name]
  separator: ;
  regex: http-metrics
  replacement: $1
  action: keep
- source_labels: [__meta_kubernetes_endpoint_address_target_kind,
  __meta_kubernetes_endpoint_address_target_name]
  separator: ;
  regex: Node;(.*)
  target_label: node
  replacement: ${1}
  action: replace
- source_labels: [__meta_kubernetes_endpoint_address_target_kind,
  __meta_kubernetes_endpoint_address_target_name]
  separator: ;
  regex: Pod;(.*)
  target_label: pod
  replacement: ${1}
  action: replace
- source_labels: [__meta_kubernetes_namespace]
  separator: ;
  regex: (.*)
  target_label: namespace
  replacement: $1
  action: replace
- source_labels: [__meta_kubernetes_service_name]
  separator: ;
  regex: (.*)
  target_label: service
  replacement: $1
  action: replace
- separator: ;
  regex: (.*)
  target_label: endpoint
  replacement: http-metrics
  action: replace
metric_relabel_configs:
- source_labels: [pod]

```

```
separator: ;
regex: (.*)
target_label: instance
replacement: $1
action: replace
kubernetes_sd_configs:
- role: endpoints
  kubeconfig_file: ""
  follow_redirects: true
  enable_http2: true
  namespaces:
    own_namespace: false
    names:
    - kube-system
```

kube-scheduler

关于 kube-scheduler 组件的更多信息，请参见 [kube-scheduler 组件指标说明](#)。

```
scrape_configs:
- job_name: kube-scheduler
  honor_labels: true
  honor_timestamps: true
  params:
    component:
    - scheduler
  scrape_interval: 30s
  scrape_timeout: 10s
  metrics_path: /master/metrics
  scheme: http
  follow_redirects: true
  enable_http2: true
  relabel_configs:
  - source_labels: [job]
    separator: ;
    regex: (.*)
    target_label: __tmp_prometheus_job_name
    replacement: $1
    action: replace
  - source_labels: [__meta_kubernetes_service_label_app_kubernetes_io_managed_by,
    __meta_kubernetes_service_labelpresent_app_kubernetes_io_managed_by]
    separator: ;
    regex: (Helm);true
    replacement: $1
    action: keep
  - source_labels: [__meta_kubernetes_service_label_label_qcloud_app,
    __meta_kubernetes_service_labelpresent_label_qcloud_app]
    separator: ;
    regex: (cluster-monitor);true
    replacement: $1
    action: keep
  - source_labels: [__meta_kubernetes_service_label_label_qcloud_service,
    __meta_kubernetes_service_labelpresent_label_qcloud_service]
    separator: ;
    regex: (master-metrics-service);true
    replacement: $1
    action: keep
  - source_labels: [__meta_kubernetes_endpoint_port_name]
    separator: ;
```

```
    regex: http-metrics
    replacement: $1
    action: keep
  - source_labels: [__meta_kubernetes_endpoint_address_target_kind,
    __meta_kubernetes_endpoint_address_target_name]
    separator: ;
    regex: Node;(.*)
    target_label: node
    replacement: ${1}
    action: replace
  - source_labels: [__meta_kubernetes_endpoint_address_target_kind,
    __meta_kubernetes_endpoint_address_target_name]
    separator: ;
    regex: Pod;(.*)
    target_label: pod
    replacement: ${1}
    action: replace
  - source_labels: [__meta_kubernetes_namespace]
    separator: ;
    regex: (.*)
    target_label: namespace
    replacement: $1
    action: replace
  - source_labels: [__meta_kubernetes_service_name]
    separator: ;
    regex: (.*)
    target_label: service
    replacement: $1
    action: replace
  - separator: ;
    regex: (.*)
    target_label: endpoint
    replacement: http-metrics
    action: replace
metric_relabel_configs:
  - source_labels: [pod]
    separator: ;
    regex: (.*)
    target_label: instance
    replacement: $1
    action: replace
kubernetes_sd_configs:
  - role: endpoints
    kubeconfig_file: ""
    follow_redirects: true
    enable_http2: true
    namespaces:
      own_namespace: false
      names:
        - kube-system
```

kube-controller-manager

关于 kube-controller-manager 组件的更多信息，请参见 [kube-controller-manager 组件指标说明](#)。

```
scrape_configs:
  - job_name: kube-controller-manager
    honor_labels: true
    honor_timestamps: true
```

```
params:
  component:
    - controller-manager
scrape_interval: 30s
scrape_timeout: 10s
metrics_path: /master/metrics
scheme: http
follow_redirects: true
enable_http2: true
relabel_configs:
- source_labels: [job]
  separator: ;
  regex: (.*)
  target_label: __tmp_prometheus_job_name
  replacement: $1
  action: replace
- source_labels: [__meta_kubernetes_service_label_app_kubernetes_io_managed_by,
  __meta_kubernetes_service_labelpresent_app_kubernetes_io_managed_by]
  separator: ;
  regex: (Helm);true
  replacement: $1
  action: keep
- source_labels: [__meta_kubernetes_service_label_label_qcloud_app,
  __meta_kubernetes_service_labelpresent_label_qcloud_app]
  separator: ;
  regex: (cluster-monitor);true
  replacement: $1
  action: keep
- source_labels: [__meta_kubernetes_service_label_label_qcloud_service,
  __meta_kubernetes_service_labelpresent_label_qcloud_service]
  separator: ;
  regex: (master-metrics-service);true
  replacement: $1
  action: keep
- source_labels: [__meta_kubernetes_endpoint_port_name]
  separator: ;
  regex: http-metrics
  replacement: $1
  action: keep
- source_labels: [__meta_kubernetes_endpoint_address_target_kind,
  __meta_kubernetes_endpoint_address_target_name]
  separator: ;
  regex: Node;(.*)
  target_label: node
  replacement: ${1}
  action: replace
- source_labels: [__meta_kubernetes_endpoint_address_target_kind,
  __meta_kubernetes_endpoint_address_target_name]
  separator: ;
  regex: Pod;(.*)
  target_label: pod
  replacement: ${1}
  action: replace
- source_labels: [__meta_kubernetes_namespace]
  separator: ;
  regex: (.*)
  target_label: namespace
  replacement: $1
  action: replace
```

```
- source_labels: [__meta_kubernetes_service_name]
  separator: ;
  regex: (.*)
  target_label: service
  replacement: $1
  action: replace
- separator: ;
  regex: (.*)
  target_label: endpoint
  replacement: http-metrics
  action: replace
metric_relabel_configs:
- source_labels: [pod]
  separator: ;
  regex: (.*)
  target_label: instance
  replacement: $1
  action: replace
kubernetes_sd_configs:
- role: endpoints
  kubeconfig_file: ""
  follow_redirects: true
  enable_http2: true
  namespaces:
    own_namespace: false
    names:
    - kube-system
```

通过 ServiceMonitor 进行采集

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: master-metrics-exporter
  namespace: kube-system
spec:
  endpoints:
  - honorLabels: true
    interval: 15s
    metricRelabelings:
    - action: replace
      sourceLabels:
      - pod
      targetLabel: instance
    params:
      component:
      - apiserver
    path: /master/metrics
    port: http-metrics
    relabelings:
    - action: replace
      replacement: kube-apiserver
      targetLabel: job
  - honorLabels: true
    interval: 15s
    metricRelabelings:
    - action: replace
      sourceLabels:
```

```
- pod
  targetLabel: instance
params:
  component:
    - controller-manager
path: /master/metrics
port: http-metrics
relabelings:
- action: replace
  replacement: kube-controller-manager
  targetLabel: job
- honorLabels: true
  interval: 15s
metricRelabelings:
- action: replace
  sourceLabels:
    - pod
  targetLabel: instance
params:
  component:
    - scheduler
path: /master/metrics
port: http-metrics
relabelings:
- action: replace
  replacement: kube-scheduler
  targetLabel: job
namespaceSelector: {}
selector:
  matchLabels:
    app.kubernetes.io/managed-by: Helm
    label_qcloud_app: cluster-monitor
    label_qcloud_service: master-metrics-service
```

TKE 集群外监控采集

本文以同 VPC 下集群外采集为示例。

前提条件

被采集集群已开启公网或内网访问。

开启跨集群指标采集

1. 登录 [容器服务控制台](#)，选择左侧导航栏中的**集群**。
2. 在**集群管理**页面，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航中的**组件管理**。
4. 单击 clustermonitor 组件的**更新配置**。
5. 在更新组件配置页面，**勾选**开启控制面组件监控开启集群外采集，选择容器子网。

基本信息

所在地域 华南地区(广州)
集群ID cls-
资源名称 clustermonitor (ClusterAddon)

控制面组件监控开启集群外采集

开启后，控制面组件指标可实现集群外采集。[查看详情](#)

容器子网

subnet-



选择容器子网后不支持更新，若需变更容器子网，请关闭控制面组件监控开启集群外采集能力后重新开启配置。

说明：

选择容器子网后不支持更新，若需变更容器子网，请关闭控制面组件监控开启集群外采集能力后重新开启配置。

6. 单击完成。

采集配置

具体操作请参见 [Configuration](#) 和 [Monitoring kubernetes with prometheus from outside of k8s cluster](#)，与集群内配置相比，主要修改 `kubernetes_sd_configs` 配置，配置方式如下：

```
scrape_configs:
- job_name: out-tke-apiserver
  kubernetes_sd_configs:
  - role: endpoints
    kubeconfig_file: ""
    follow_redirects: true
    enable_http2: true
    namespaces:
      names:
      - kube-system
    namespaces:
      names:
      - kube-system
    api_server: 'https://<KUBERNETES_URL>'
    tls_config:
      ca_file: /etc/prometheus/kubernetes-ca.crt
      bearer_token: '<SERVICE_ACCOUNT_BEARER_TOKEN>'
    scrape_interval: 30s
```

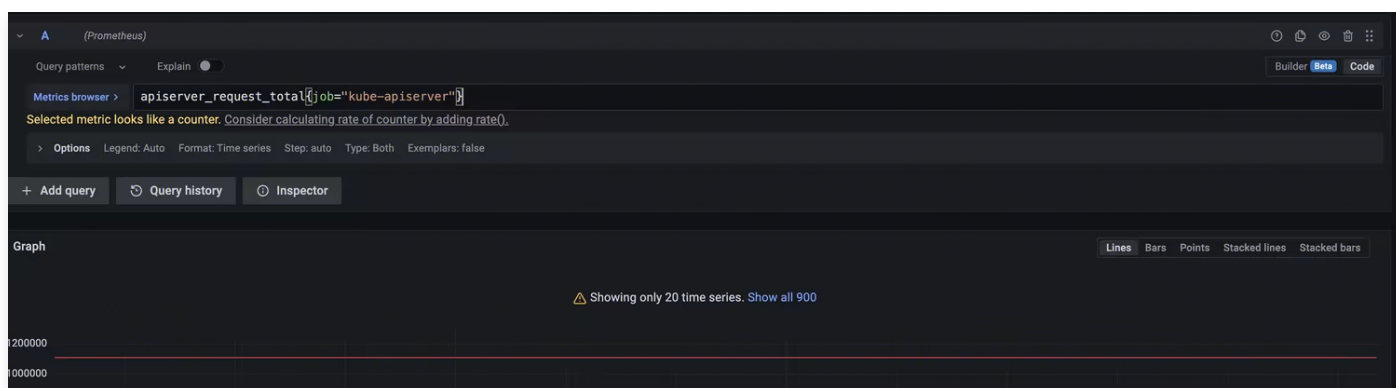
验证

1. 登录自建的 Prometheus 控制台，切换到 Graph 页面。
2. 输入 `up`，查看是否全部控制面组件都可以显示。



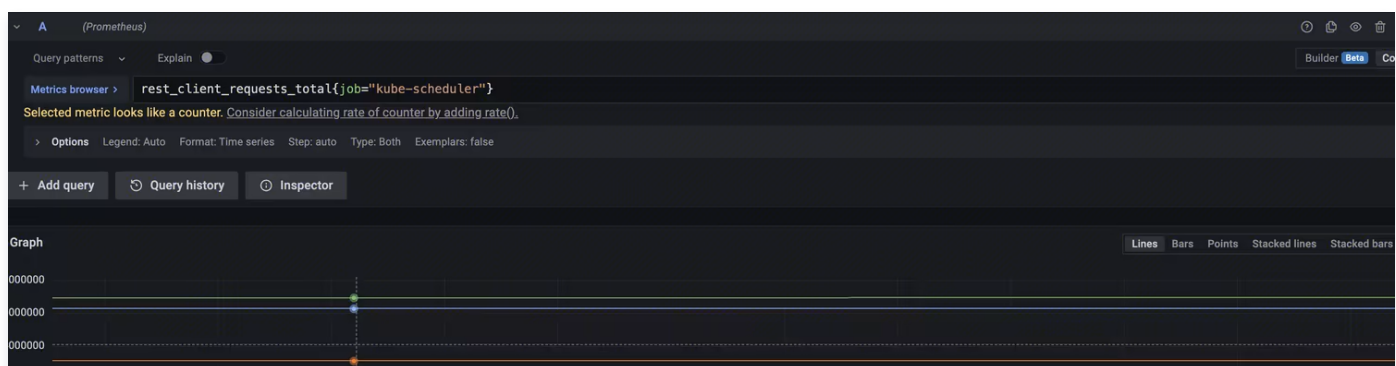
3. 输入以下指令，验证 API Server 是否正常：

```
apiserver_request_total{job="apiserver"}
```



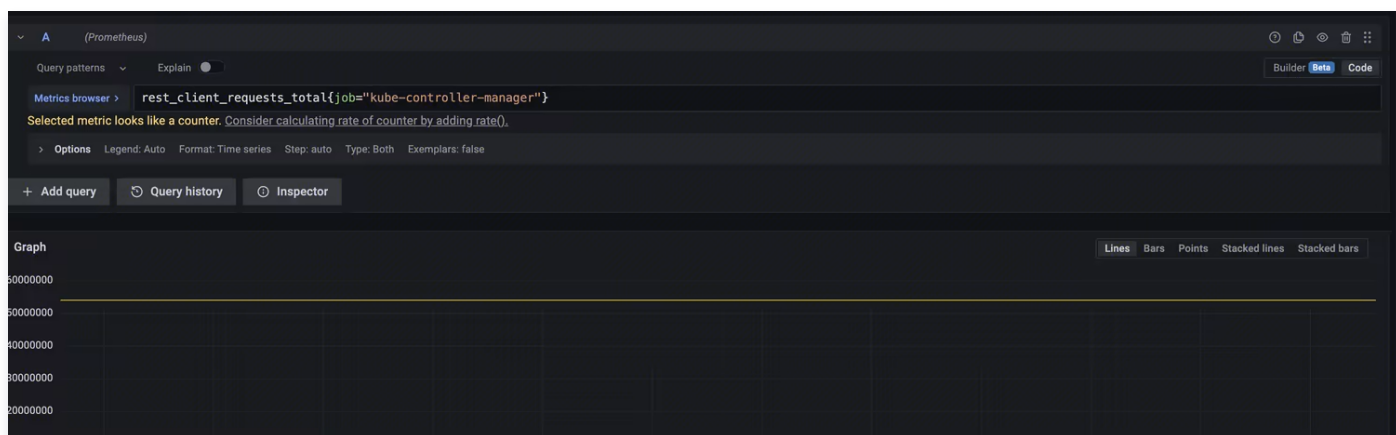
4. 输入以下指令，验证 Scheduler 是否正常：

```
rest_client_requests_total{job="scheduler"}
```



5. 输入以下指令，验证 Controller Manager 是否正常：

```
rest_client_requests_total{job="controller-manager"}
```



应用性能管理

最近更新时间：2025-05-21 14:34:42

背景

随着企业越来越多地采用容器化和微服务架构，应用的复杂性显著增加，性能管理变得尤为重要。在 Kubernetes 中，应用性能管理（APM）涉及监控、分析和优化应用的资源使用、响应时间和可用性。有效的性能管理可以提高应用响应速度、优化资源利用率、减少停机时间、提升用户体验。

前提条件

登录 [腾讯云可观测平台](#) 控制台，创建业务系统管理。详情请参见 [新建业务系统](#)。

接入方案选型

APM 支持 OpenTelemetry、Skywalking 等多种上报协议。OpenTelemetry 社区活跃度高，技术更迭迅速，广泛兼容主流编程语言、组件与框架，请优先选择 OpenTelemetry 方案，关于 OpenTelemetry 的更多信息请参考 [OpenTelemetry 官方网站](#)。

对于部署在 Kubernetes 的应用，请优先选择 Operator 模式自动接入，节省安装探针和修改配置方面的工作量，关于 Operator 模式的更多信息请参考 [安装 tencent-opentelemetry-operator](#)。

接入方式

不同编程语言的推荐接入方案请参见 [接入方式](#)。

Operator 方案

tencent-opentelemetry-operator 是腾讯云基于社区 opentelemetry-operator 构建的探针接入方案。由于实现了 Kubernetes 环境下探针自动注入机制，免去了探针部署和配置的工作量，让应用接入变得更加简单。目前 tencent-opentelemetry-operator 支持 Java、Python、node.js、.Net 应用的快速接入，应用程序部署到腾讯云容器服务 TKE 以后，只需要在工作负载的 YAML 文件中添加2行 annotation，就能完成整个接入流程，无需在开发态涉及任何代码的修改。

快速接入应用性能监控（APM）

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**页面，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航中的**工作负载**。
4. 单击进入需要监控的工作负载，选择**应用性能**，单击**立即接入**，并选择应用类型快速接入应用性能监控。

腾讯云HA提供GPU和Otlama环境，支持本地部署DeepSeek-R1模型，限时低至0.8折 [查看详情](#) >

集群-（北京）

Pod管理 详情 修订历史 **应用性能** 事件 日志 YAML

通过接入应用性能监控（APM）获得工作负载的应用性能管理能力

应用性能监控（Application Performance Management, APM）是专业的应用性能管理平台，为您提供分布式应用性能分析和故障自修复能力，全方位保障系统的可用性和稳定性。关于 APM 的产品计费，请参考[计费概述](#)。

对于Java、Python、Node.js 和 .Net应用，参考Operator 方案。

Operator 方案会自动注入探针，实现 TKE 工作负载的自动接入，并自动建立工作负载与 APM 应用之间的关联。

使用其他方式接入的工作负载，参考[自定义应用接入流程](#)。

通过创建或更新手动接入 TKE 相关信息，以建立工作负载与 APM 应用之间的关联。

立即接入

应用性能总览

基于响应时间、吞吐量、错误率、Apdex指标等维度，快速了解系统应用性能以及应用健康状态。

应用性能详情

基于性能监控、链路分析、错误分析等视图，深入分析单个应用的性能指标，并进行针对性的优化。

全局拓扑

可视化展示应用之间的调用关系和依赖关系，全面理解系统的整体架构，提升故障效率。

链路追踪

通过跨应用的多维度调用链搜索与分析，快速定位性能瓶颈，跟踪链路中每一个环节的调用耗时以及执行状态。

性能剖析

以低性能瓶颈、高延迟的调用链CPU内存飙升、GC频繁等现象为起因，帮助解决应用性能问题的疑难杂症。

接入Java应用

北京 mesh [apm-vm...] [新建业务系统](#) [接入遇到问题？扫码加技术交流群](#)

接入协议类型

OpenTelemetry 增强探针

Skywalking

上报方式

TKE环境自动接入

内网上报

外网上报

接入流程

对于部署在容器服务 TKE 上的 Java 应用，APM 提供自动接入方案，可以在应用部署到 TKE 之后实现探针自动注入，方便应用快速接入。

步骤 1：安装 Operator

请先通过 [一键安装 Operator](#) 在 TKE 集群安装 Operator。

步骤 2：在工作负载添加 annotation

1. 登录 [容器服务](#) 控制台。

2. 点击**集群**，进入对应的 TKE 集群。

3. 在**工作负载**中找到需要接入 APM 的应用，点击**更多**，点击**编辑yaml**。

4. 应用所在的 Pod annotation 中如下内容，点击**完成**，即可以完成接入。

D

cloud.tencent.com/project-java: "true"
cloud.tencent.com/otel-service-name: my-app # 应用名，多个应用相同应用名会注入的进程，在APP中会表现为相同应用下的多个实例

请注意，此内容需要添加到 spec.template.metadata.annotations 中，作用于 Pod 的 annotation，而不是工作负载的 annotation，可以参考如下代码片段：

D
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
 k8s-app: my-app
 name: my-app
 namespace: default
spec:
 selector:
 matchLabels:
 k8s-app: my-app
 template:
 metadata:
 labels:
 k8s-app: my-app
 annotations:
 cloud.tencent.com/project-java: "true" # 添加到这里
 cloud.tencent.com/otel-service-name: my-app
 spec:
 containers:
 - image: my-app:0.1
 name: my-app

接入验证

版权所有：腾讯云计算（北京）有限责任公司

第85 共123页

- 5. 单击**一键安装 Operator**，在 TKE 集群安装 Operator。
- 6. 在**一键安装 Operator** 页面，通过下拉框选择上报地域、默认业务系统、TKE 集群等信息，确认后单击**安装**。

一键安装 Operator

×

上报地域* ⓘ

北京

默认业务系统* ⓘ

TKE所在地域* ⓘ

北京

TKE集群* ⓘ

请选择

安装需要1~2分钟时间，您可以在TKE控制台的应用中心查看安装状态
安装成功后，可以参考接入文档为Pod添加annotation实现自动接入APM

安装

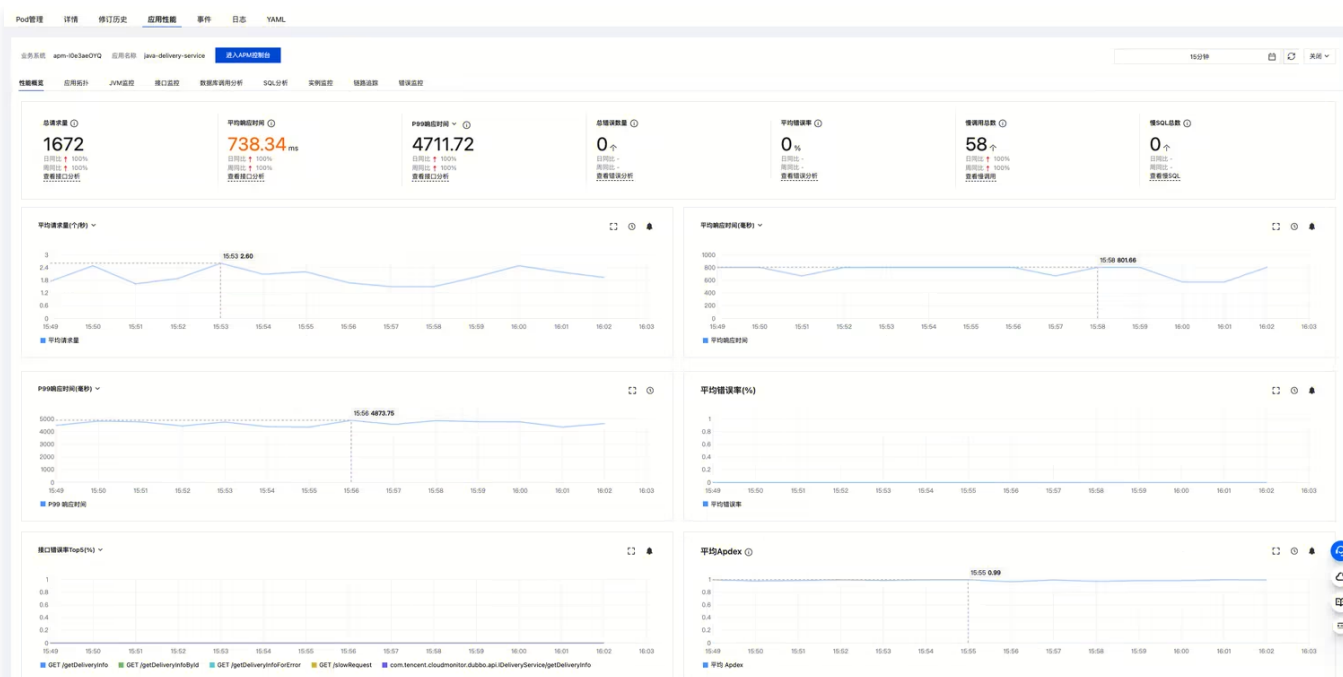
取消

说明：
上报地域为 APM 业务系统所在的地域，每个 TKE 集群只能指定唯一的上报地域，建议将上报地域与 TKE 集群所在的地域保持一致。

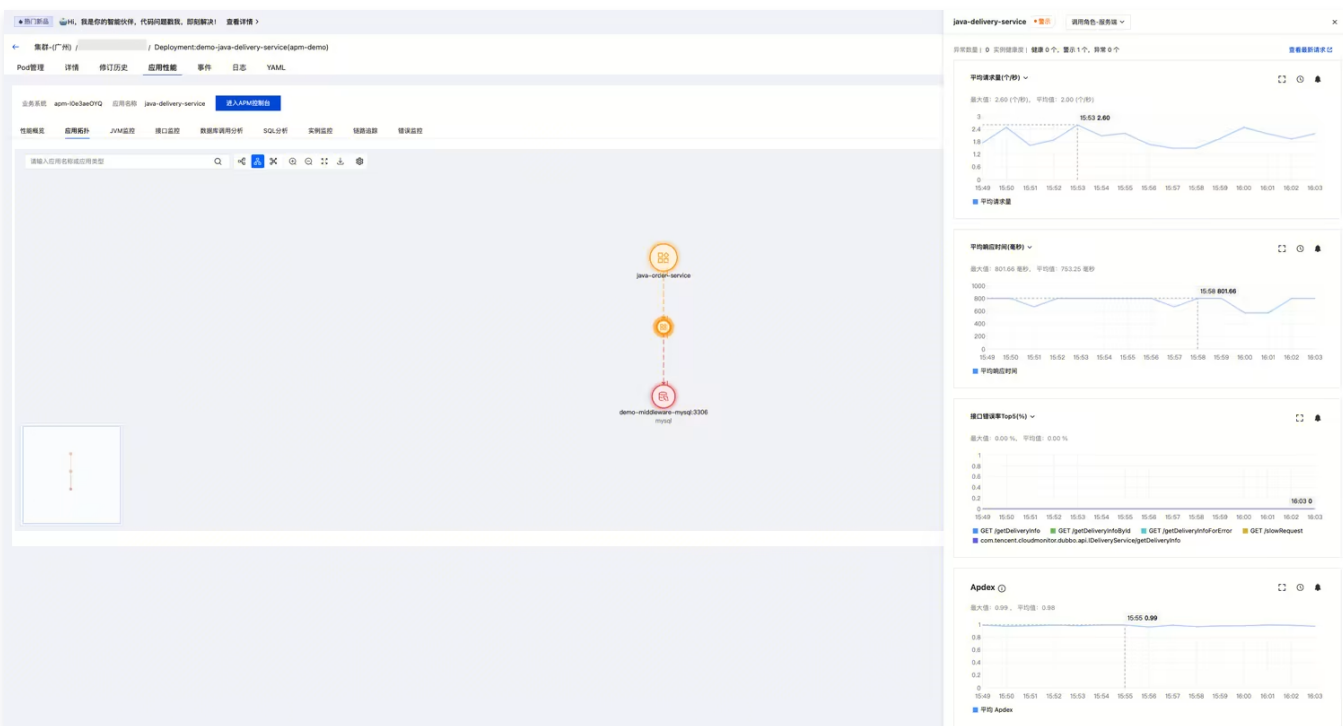
- 7. 根据应用类型参考以下文档进行后续的操作步骤。

应用类型	OpenTelemetry 方案
Go	通过 OpenTelemetry-Go 接入 Go 应用（推荐）
Java	K8s 环境自动接入 Java 应用（推荐）
Python	K8s 环境自动接入 Python 应用（推荐）
PHP	通过 OpenTelemetry-PHP 接入 PHP 应用（推荐）
Node.js	K8s 环境自动接入 Node.js 应用（推荐） 通过 OpenTelemetry-JS 方案接入 Node.js 应用（推荐）
.Net	K8s 环境自动接入 .NET 应用（推荐）
其他语言	接入其他语言编写的应用

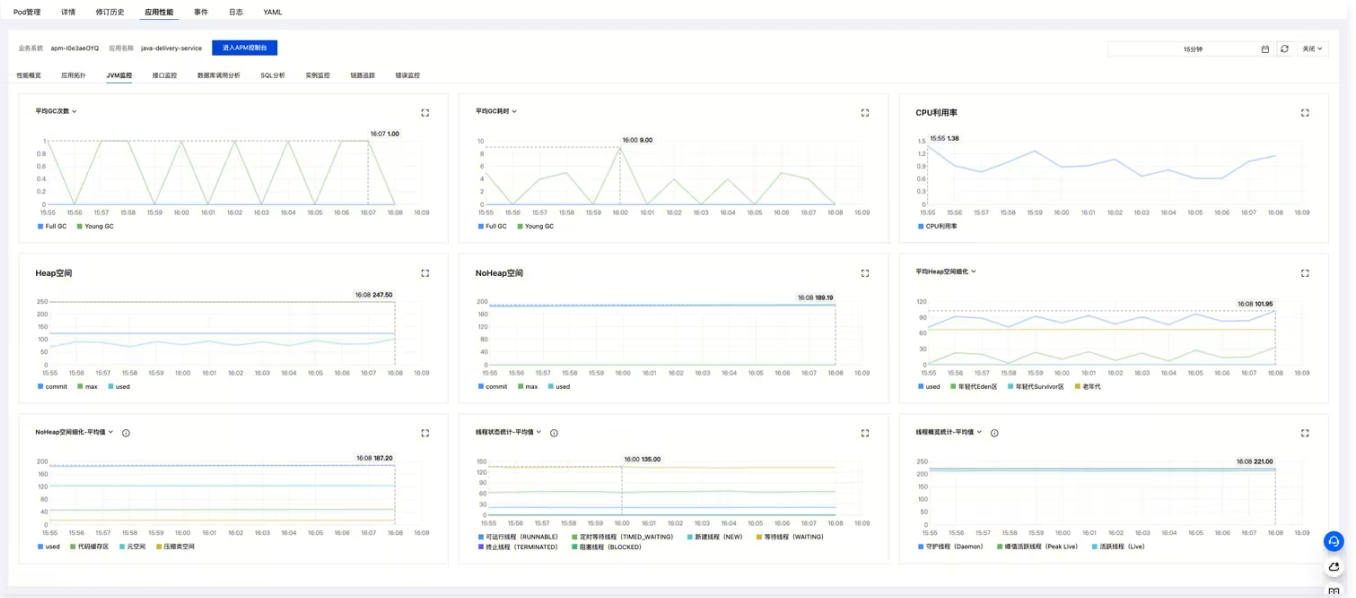
- 8. 以 Java 应用为例，可查看丰富的应用指标：
 - 性能概览



应用拓扑



JVM 监控



接口监控

Pod管理 详情 修订历史 应用性能 事件 日志 YAML

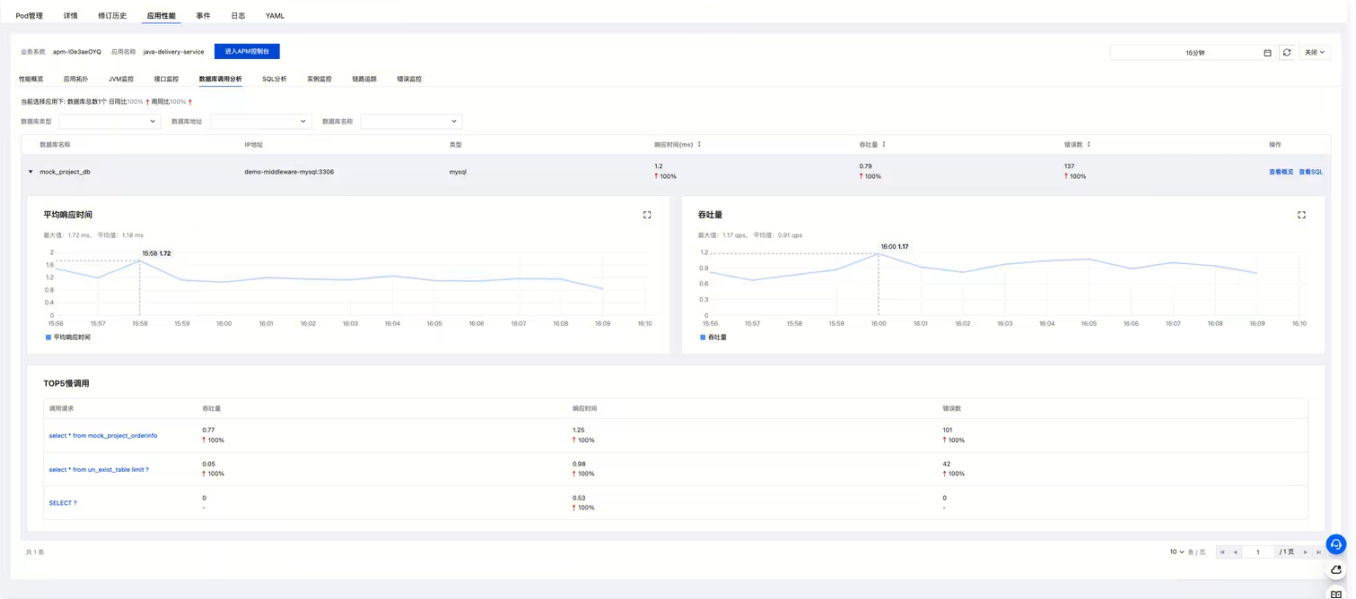
业务系统 apm-0x3ae07q 应用名称 java-delivery-service 进入APM控制台

性能概览 应用拓扑 JVM监控 接口监控 数据库调用分析 SQL分析 实例监控 链路追踪 错误监控

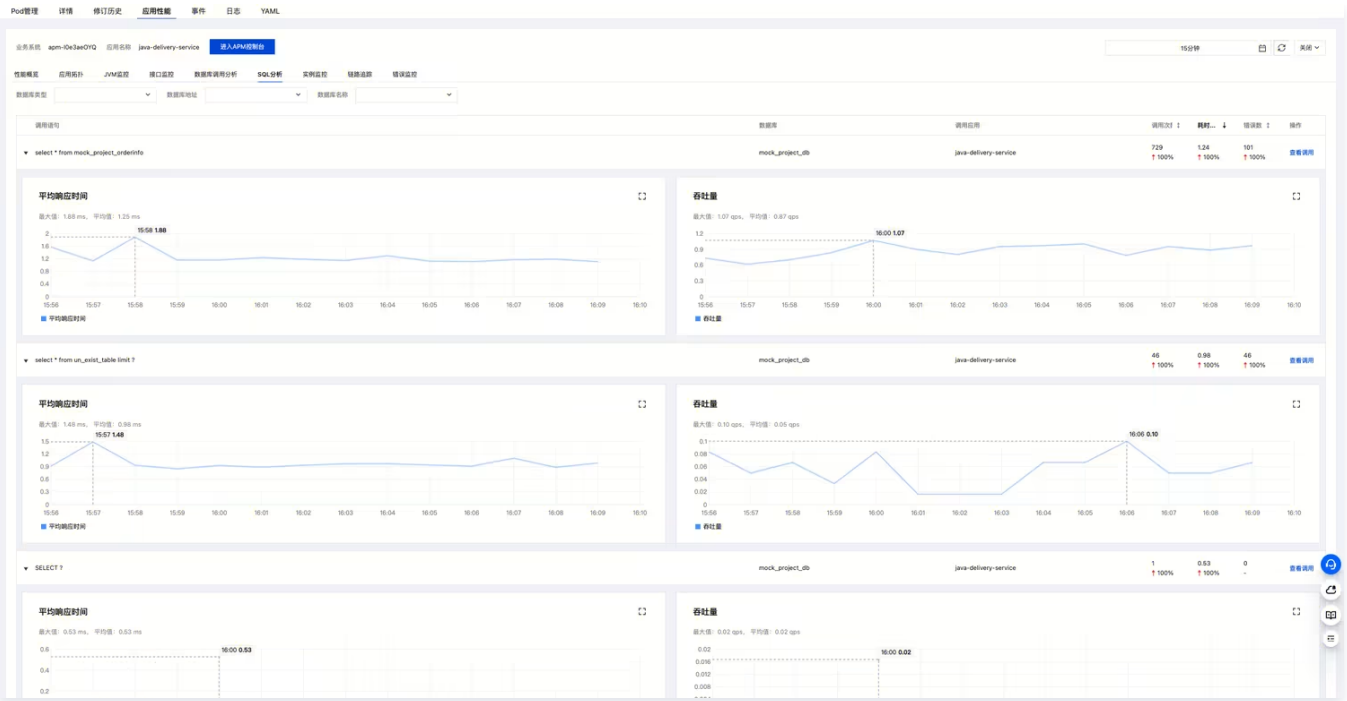
选择应用: 业务系统 | 应用名称 | 接口类型 | 应用 | 查询

接口	请求次数	吞吐量	平均耗时	P90	P95	P99	最大耗时	最小耗时	错误率/错误数
GET /liveRequest @ io.opentelemetry.tomcat-7.0	67	0.07 qps ↑ 100%	4000.78 ms ↑ 100%	4500 ms ↑ 100%	4850 ms ↑ 100%	4990 ms ↑ 100%	4001.09 ms ↑ 100%	4000.82 ms ↑ 100%	0% / 0
GET /getDeliveryInfo @ io.opentelemetry.tomcat-7.0	49	0.05 qps ↑ 100%	3.59 ms ↑ 100%	3.73 ms ↑ 100%	5.92 ms ↑ 100%	9.18 ms ↑ 100%	9.13 ms ↑ 100%	2.96 ms ↑ 100%	0% / 0
GET /getDeliveryInfoById @ io.opentelemetry.tomcat-7.0	726	0.81 qps ↑ 100%	1.97 ms ↑ 100%	3.04 ms ↑ 100%	4.88 ms ↑ 100%	8.87 ms ↑ 100%	18.88 ms ↑ 100%	0.96 ms ↑ 100%	0% / 0
GET /getDeliveryInfo @ io.opentelemetry.tomcat-7.0	424	0.47 qps ↑ 100%	0.87 ms ↑ 100%	0.53 ms ↑ 100%	1 ms ↑ 100%	4.19 ms ↑ 100%	1.93 ms ↑ 100%	0.38 ms ↑ 100%	0% / 0
com.tencent.cloudmonitor.dubbo.api.DeliveryService.getDeliveryInfo @ io.opentelemetry.apache-dubbo-2.7	424	0.47 qps ↑ 100%	0.06 ms ↑ 100%	0.5 ms ↑ 100%	0.95 ms ↑ 100%	0.99 ms ↑ 100%	0.21 ms ↑ 100%	0.22 ms ↑ 100%	0% / 0

数据库调用分析



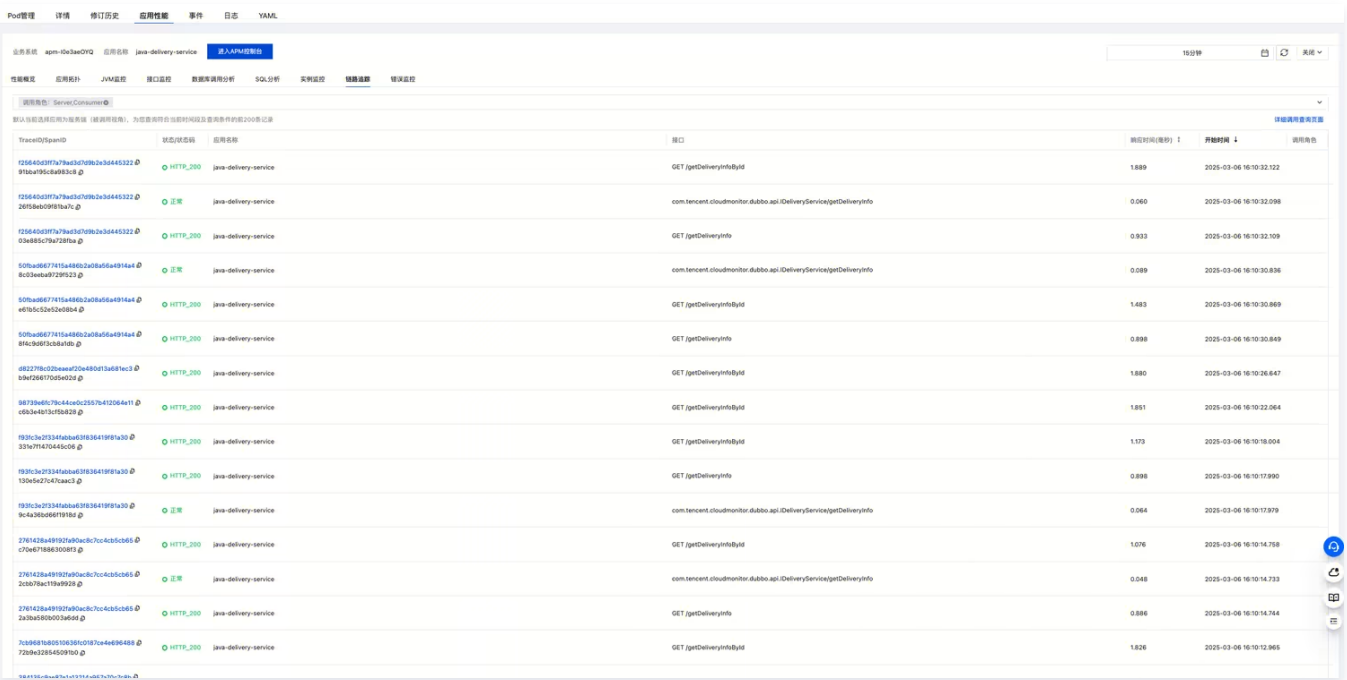
SQL 分析



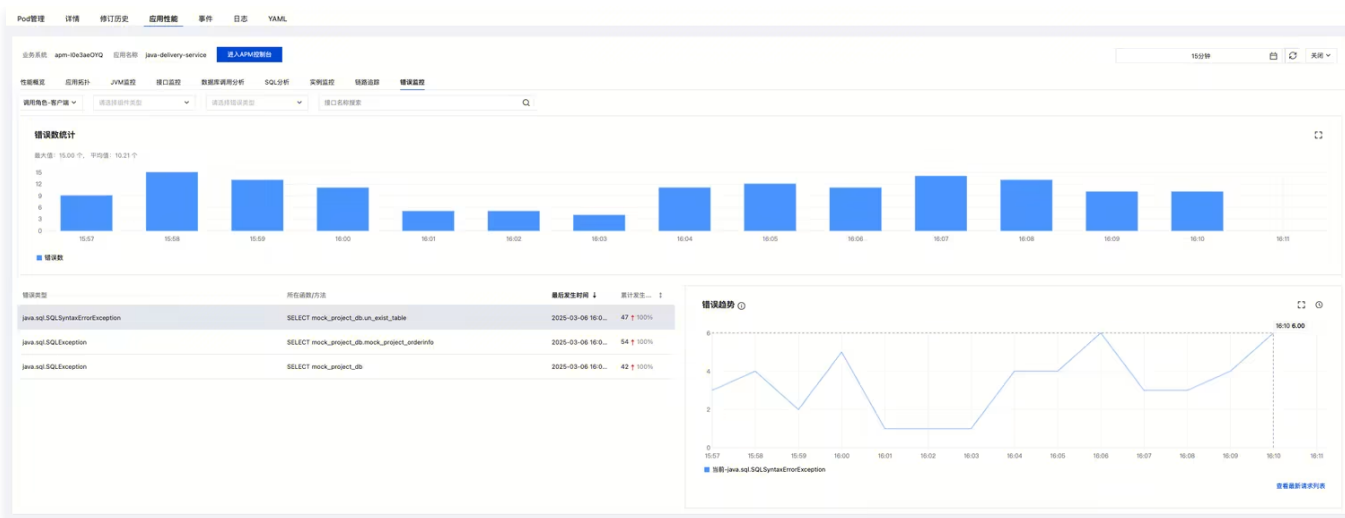
实例监控



链路追踪

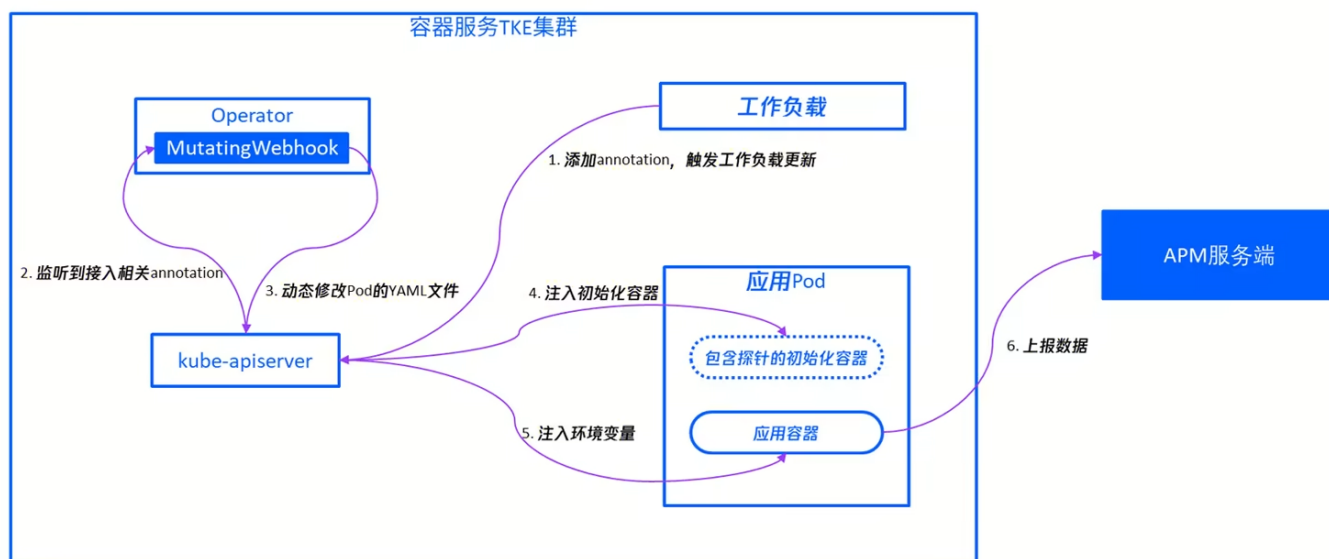


错误监控



接入原理

Operator 方案利用了 Kubernetes 提供的 Dynamic Admission Control 机制实现探针的注入，可以参考以下流程了解 Operator 方案的接入原理：



1. 用户对工作负载添加 annotation，API Server 将基于工作负载的发布策略启动工作负载的更新。
2. 在 Operator 的相关组件中，包含一个具备 Dynamic Admission Control 能力的工作负载，通过 Mutating Webhook 的形式在容器服务集群中运行，这个组件可以从 API Server 监听工作负载的变化。
3. Operator 监听到应用工作负载添加了接入相关的 annotation 之后，会通过 API Server 对即将创建的应用 Pod 进行动态修改，修改的内容通过 Pod 的 YAML 描述表现出来。
4. Operator 会在应用 Pod 中注入一个初始化容器，在容器本地路径中包含了探针文件。
5. Operator 还会在应用容器中注入一些环境变量，这些环境变量可以对应用进程进行配置，实现挂载探针的动作。以 Java 应用为例，Operator 会对应用容器添加 JAVA_TOOL_OPTIONS 环境变量，其中包括 -javaagent 参数，指向由初始化容器引入的探针文件本地路径，使 Java 进程启动的时候执行挂载探针操作。
6. 探针通过字节码增强机制自动采集监控数据，并向 APM 服务端上报。

接入优势

- **简化 Operator 安装流程**：安装部署流程，不但上架了 TKE 应用市场，还能在 APM 控制台实现 Operator 一键安装。
- **通过封装减少人力**：对接入过程中需要用到的一系列配置项进行了封装，不再需要手工填写，包括 APM 接入点的地址，业务系统 token 等。
- **优化证书管理机制**：解决了 Operator 安装过程中有可能遇到的各类兼容性问题。
- **降级处理**：当 APM 平台的任何一个组件出现故障的时候，整个接入机制都以不影响业务正常运行为前提，在必要的时候对探针注入和监控数据上报逻辑行降级处理，确保业务的稳定运行。

自定义埋点

在探针接入方案的基础上，Operator 模式通过探针注入进一步简化了应用接入 APM 的流程。tencent-opentelemetry-operator 注入的探针来自 OpenTelemetry 生态，对主流的开发框架与类库实现了运行态自动埋点。对大多数用户而言，应用在接入成功后即可完成监控数据上报，实现分布式链路追踪，不需要修改任何代码。但在某些特殊的场景中，如果探针的自动埋点机制不能满足用户需求，用户可以引入 OpenTelemetry API，在自动埋点的基础上，通过修改应用代码增加自定义埋点。例如下述场景中，可能需要引入自定义埋点：

- 应用引入的开发框架与类库没有在开发者中被广泛使用，或者版本很低。
- 应用引入的开发框架与类库由用户自己封装，或者来自于闭源商业化定制开发。
- 对于某些核心的自定义方法，需要在链路中追加方法级埋点。

对于 tencent-opentelemetry-operator 注入的探针，可以参考 [OpenTelemetry 方案支持的组件和框架](#) 了解自动埋点的范围。自定义埋点代码的具体编写方式，可以参考如下 OpenTelemetry 的官方文档：

应用	参考链接
Java	https://opentelemetry.io/docs/languages/java/instrumentation/
Python	https://opentelemetry.io/docs/languages/python/instrumentation/
Node.js	https://opentelemetry.io/docs/languages/js/instrumentation/

以 Java 语言为例，如果我们需要对某个自定义方法在链路中追加方法级埋点，可以参考以下步骤实现。

1. 引入 OpenTelemetry API 依赖：

```
<dependencies>
  <!-- 其他依赖 -->
  <dependency>
    <groupId>io.opentelemetry</groupId>
    <artifactId>opentelemetry-api</artifactId>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>io.opentelemetry</groupId>
      <artifactId>opentelemetry-bom</artifactId>
      <version>1.9.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

2. 通过如下代码，在链路中增加 doTask() 的方法级埋点。

```
import io.opentelemetry.api.trace.Span;
import io.opentelemetry.api.trace.StatusCode;
import io.opentelemetry.api.trace.Tracer;
import io.opentelemetry.context.Scope;

// Trace 对象可以在业务方法中获取，或者通过参数传入业务方法
public void doTask(Tracer tracer) {
  // 创建一个 Span
  Span span = tracer.spanBuilder("doTask").startSpan();
  // 在 Span 中添加一些 Attributes
  span.setAttribute("RequestId", "5fc92ff1-8ca8-45f4-8013-24b4b5257666");
  // 将此 Span 设置为当前的Span
  try (Scope scope = span.makeCurrent()) {
    doSubTask1();
  }
```

```
doSubTask2();  
} catch (Throwable t) {  
    // 处理异常，异常信息将记录到 Span 的对应事件中  
    span.recordException(t);  
    span.setStatus(StatusCode.ERROR);  
    throw t;  
} finally {  
    // 结束 Span  
    span.end();  
}  
}
```

3. 前往 [腾讯云可观测平台控制台](#)，在应用性能监控 > 调用查询中，找到相关的调用链，单击 Span ID 后，链路详情页面中，即可查到通过自定义埋点新增的 Span。

告警管理

最近更新时间：2025-07-15 15:26:32

使用场景

通过配置监控，能够获取集群的实时状态信息；通过配置告警，能够及时发现问题并触发告警，减少故障排查时间并及时处理故障。您可以确保 Kubernetes 集群的高效、稳定运行，同时降低运维成本，提升业务价值。

TKE 集群支持以下三类告警方式：

- **基础监控指标告警**：基于集群及基础资源的监控指标进行告警。
- **日志告警**：基于日志采集与分析进行告警，包括自定义日志、事件日志、审计日志等。
- **默认事件与审计告警**：内置常见异常场景的默认告警规则，支持一键开启，属于日志告警的特殊形式。该功能结合腾讯云容器服务大规模多场景运维经验，覆盖集群和容器运行过程中常见的异常场景，能够及时发出预警，帮助用户快速发现并定位问题，维护集群和业务的稳定运行。具有以下特点：
 - 预设定性：默认告警规则是预先设定好的，用户无需手动创建，方便快捷。
 - 通用性：默认告警规则通常适用于大多数场景，可以满足大部分用户的可观测需求。
 - 灵活性：用户可以根据自己的实际需求，自定义告警规则。
 - 易于管理：默认告警规则可以帮助用户快速搭建起一套基本的监控体系，降低管理成本。

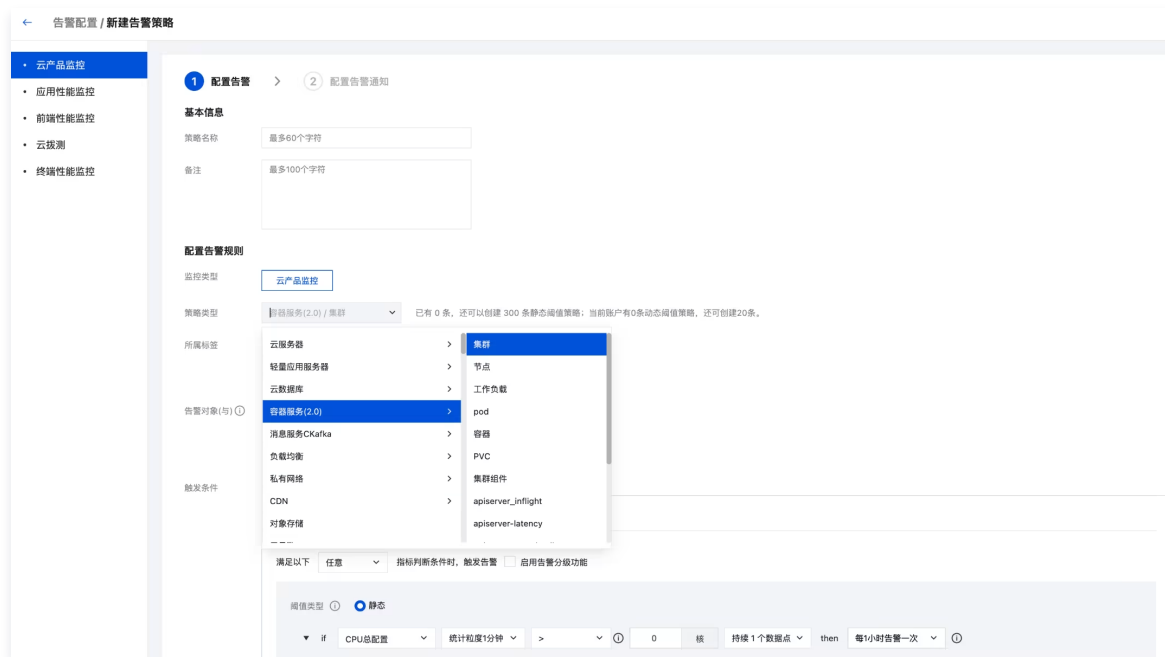
前提条件

- **基础监控指标告警**：集群相关基础资源的指标告警，此类告警的数据链路同步自云监控基础资源监控。更多信息，请参见 [基础资源监控](#)。
- **日志告警、默认事件与审计告警**：集群异常事件告警、异常审计告警。需同时开启日志服务功能，请参见 [日志采集](#)、[事件日志](#)、[审计日志](#)。

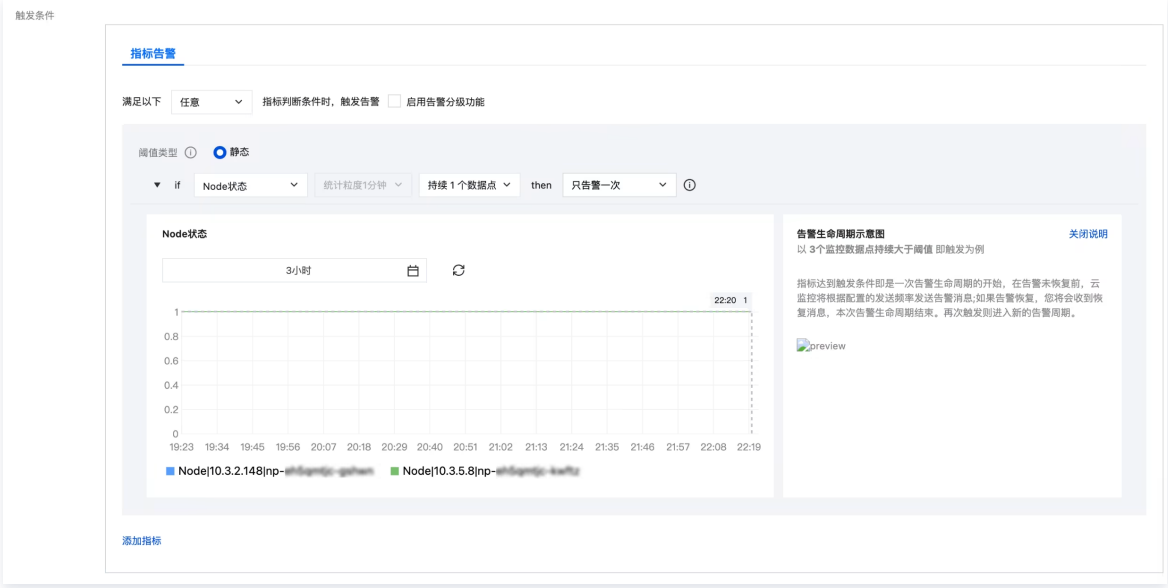
操作指引

基础监控指标告警配置

1. 登录 [容器服务控制台](#)，选择左侧导航栏中的**运维中心 > 告警设置**。
2. 在**新建告警策略**页面，策略类型选择**容器服务(2.0)**，选择所需设置的指标维度。



3. 设置触发条件，可增加多个指标，请参见 [监控及告警指标列表](#)。



4. 配置告警通知。



5. 单击完成，可在告警策略中查看。



日志告警配置

1. 登录 [容器服务控制台](#)，选择左侧导航栏中的**集群**。
2. 在**集群管理**页面，单击目标集群 ID，进入集群详情页。
3. 选择左侧导航中的**日志**，配置日志采集规则并投递到 CLS，详情参见 [采集容器日志到 CLS](#)、[事件日志](#)、[审计日志](#)。
4. 登录 [日志服务控制台](#)，选择左侧导航栏中的**监控告警** > **告警策略**。
5. 单击**新建**，即可新建日志告警策略，详情参见 [配置告警策略](#)。



默认事件与审计告警一键开启

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. 选择左侧导航中的**告警**，按需选择事件告警、审计告警，单击**一键开启**。



4. 前往开启事件日志、审计日志。
5. 基于默认告警模板做告警策略选择。

批量新建告警策略

×

实例选择

地域

清远

云产品类型

容器服务TKE

容器集群(标准)

使用模版新建告警策略依赖TKE集群开启事件日志功能

事件日志

已开启

告警模版选择

已选择规则(23)

清空已选

选择推荐模版

告警模版	告警等级	类型	状态
☐ 节点磁盘容量不合法	警告	TKE事件日志	未配置
☒ 清理磁盘空间失败	紧急	TKE事件日志	未配置
☒ 节点磁盘多次不足	警告	TKE事件日志	未配置
☒ 节点磁盘不足	提醒	TKE事件日志	未配置
☒ 节点状态事件告警			
☒ 集群网络事件告警			
☒ Service 事件告警			

关联通知渠道组

通知渠道组

关联通知渠道组

新建通知渠道组

其他配置

标签

标签键

标签值

×

+ 添加

键值粘贴板

- 关联通知渠道组，通知渠道推荐邮件、短信、微信、电话。详情参见 [管理通知渠道组](#)。
- 单击确定，即可一键开启默认事件告警策略、审计告警策略。
- 可在告警策略中查看创建的告警策略。

告警策略

告警历史

批量新建告警策略

策略名称

应用状态

策略对象

应用状态

通知渠道

提醒

操作

节点磁盘容量不合法

策略名称: 节点磁盘容量不合法

策略对象: 集群名称: 集群名称

应用状态: 已开启

通知渠道: 通知渠道

提醒: 1

操作: 编辑 复制 删除 策略管理

清理磁盘空间失败

策略名称: 清理磁盘空间失败

策略对象: 集群名称: 集群名称

应用状态: 已开启

通知渠道: 通知渠道

提醒: 1

操作: 编辑 复制 删除 策略管理

节点磁盘多次不足

策略名称: 节点磁盘多次不足

策略对象: 集群名称: 集群名称

应用状态: 已开启

通知渠道: 通知渠道

提醒: 1

操作: 编辑 复制 删除 策略管理

节点磁盘不足

策略名称: 节点磁盘不足

策略对象: 集群名称: 集群名称

应用状态: 已开启

通知渠道: 通知渠道

提醒: 1

操作: 编辑 复制 删除 策略管理

节点状态事件告警

策略名称: 节点状态事件告警

策略对象: 集群名称: 集群名称

应用状态: 已开启

通知渠道: 通知渠道

提醒: 1

操作: 编辑 复制 删除 策略管理

集群网络事件告警

策略名称: 集群网络事件告警

策略对象: 集群名称: 集群名称

应用状态: 已开启

通知渠道: 通知渠道

提醒: 1

操作: 编辑 复制 删除 策略管理

Service 事件告警

策略名称: Service 事件告警

策略对象: 集群名称: 集群名称

应用状态: 已开启

通知渠道: 通知渠道

提醒: 1

操作: 编辑 复制 删除 策略管理

9. 可在告警历史中查看告警历史，单击查看详情。

事件告警 统计告警

告警策略

告警历史

告警策略 全部

日志主题 全部

报警主题 全部

告警状态 全部

告警级别 全部

报警时间范围 本月

发生时间	告警策略	容器集群	监控对象	触发条件及分幅	通知渠道	持续时间	告警级别	告警状态	操作
2025-06-19 21:13:43	集群CIS- CIS-节点CPU使用率	集群CIS-节点CPU使用率	节点CPU使用率	\$1.log_count > 1	短信 notice	6分钟	警告	已恢复	查看详情
2025-06-19 21:05:42	集群CIS- 弹性网卡IP地址冲突	集群CIS-弹性网卡IP地址冲突	弹性网卡IP地址冲突	\$1.log_count > 1	短信 notice	7分钟	警告	已恢复	查看详情
2025-06-19 20:51:01	集群CIS- 设置辅助网卡	集群CIS-设置辅助网卡	设置辅助网卡	\$1.log_count > 0	短信 notice	5分钟	紧急	已恢复	查看详情
2025-06-19 20:48:55	集群CIS- 节点网络内	集群CIS-节点网络内	节点网络内	\$1.log_count > 0	短信 notice	5分钟	紧急	已恢复	查看详情
2025-06-19 20:24:47	集群CIS- 同步辅助网卡	集群CIS-同步辅助网卡	同步辅助网卡	\$1.log_count > 1	短信 notice	8分钟	警告	已恢复	查看详情
2025-06-19 20:03:53	集群CIS- 同步指定子网	集群CIS-同步指定子网	同步指定子网	\$1.log_count > 1	短信 notice	5分钟	紧急	已恢复	查看详情

默认告警模板

节点资源事件告警策略集			
模板名称	模板说明	模板类型	事件名称
节点磁盘容量不合法	节点磁盘容量不合法	事件类	InvalidDiskCapacity
清理磁盘空间失败	清理磁盘空间失败	事件类	FreeDiskSpaceFailed
节点磁盘不足	节点磁盘不足	事件类	NodeHasDiskPressure
节点磁盘多次不足	节点磁盘不足	事件类	NodeHasDiskPressure

节点状态事件告警策略集			
模板名称	模板说明	模板类型	事件名称
节点内存不足强杀进程	节点是否存在 OOM 事件	事件类	SystemOOM
节点状态异常	节点状态异常	事件类	NodeNotReady

集群网络事件告警策略集			
模板名称	模板说明	模板类型	事件名称
设置辅助网卡网络失败	设置辅助网卡网络失败，可能影响节点数据面网络通信	事件类	FailedSetupENINetwork
新建节点全局路由失败	新建节点全局路由失败	事件类	FailedToCreateRoute
节点网络内核参数设置失败	节点网络内核参数设置失败，可能影响数据面网络通信	事件类	FailedSetupKernel
固定IP模式下尝试添加共享网卡失败	固定 IP 模式下尝试添加共享网卡失败	事件类	FailedAllocENI
ip_forward 参数被修改	ip_forward 参数被修改，可能影响数据面网络通信	事件类	IPForwardChanged
弹性网卡 IP 分配失败	尝试分配弹性网卡 IP 失败	事件类	FailedAddRouteIPs
节点 PodCIDR 分配失败	ipamd 同步 node 并尝试分配 podCIDR 失败	事件类	SyncNodeFailed
同步指定子网失败	同步指定子网失败	事件类	FailedSyncNominatedSubnets
同步辅助网卡安全组配置失败	同步辅助网卡安全组配置失败	事件类	FailedSyncSecurityGroups

rp_filter 参数被修改	rp_filter 参数被修改，可能影响数据面网络通信	事件类	RpFilterChanged
全局路由冲突	全局路由冲突，可能影响节点数据面网络通信	事件类	GlobalRoutesConflict
非固定 IP 模式下尝试添加共享网卡失败	非固定 IP 模式下尝试添加共享网卡失败	事件类	FailedAttachRENI

Service 事件告警策略集

模板名称	模板说明	模板类型	事件名称
CLB 监听器配额达到上限	CLB 监听器配额达到上限	事件类	EnsureServiceFailed
CLB 后端 RS 配额达到上限	CLB 后端 RS 配额达到上限	事件类	EnsureServiceFailed
监听器使用证书已过期	监听器使用证书已过期	事件类	EnsureServiceFailed
后端 RS 单点风险	后端 RS 单点风险	事件类	EnsureServiceWarning

Ingress 事件告警策略集

模板名称	模板说明	模板类型	事件名称
CLB 监听器配额达到上限	CLB 监听器配额达到上限	事件类	EnsureServiceFailed
CLB 后端 RS 配额达到上限	CLB 后端 RS 配额达到上限	事件类	EnsureServiceFailed
监听器使用证书已过期	监听器使用证书已过期	事件类	EnsureServiceFailed
后端 RS 单点风险	后端 RS 单点风险	事件类	EnsureServiceWarning

健康检查

最近更新时间：2025-09-04 10:51:42

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
	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**运维中心 > 健康检查**。
2. 在**健康检查**页，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：

☐	ID/名称	检查进度	检查结果	上次检查时间	自动检查	操作
☐	cls- [模糊]	已完成	警告16项 查看结果	2025-08-26 20:1...	未开启	立即检查 自动检查

在**自动检查**设置页面，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：

自动检查设置

×

开启状态

☐

请设置集群cls-

健康自动检查周期

检查周期

☒ 每天 ☐ 每周

时刻

0点

▼

确定

取消

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：

☐	ID/名称	检查进度	检查结果	上次检查时间	自动检查	操作
☐	cls- [模糊]	初始化环境，安装agent... 1%	检查中...	2025-08-26 20:1...	未开启	立即检查 自动检查

4. 检查完成后，可单击**查看结果**以查看检查报告。如下图所示：

☐	ID/名称	检查进度	检查结果	上次检查时间	自动检查	操作
☐	cls- cilium-gateway-test	已完成	建议1项 查看结果	2020-08-03 16:57:01	未开启	立即检查 自动检查
☐	cls- go-kubelet-probe	已完成	警告6项 查看结果	2020-08-03 16:57:07	未开启	立即检查 自动检查

5. 在检查报告页面，选择**资源状态**和**运行情况**分别查看资源状态检查结果和运行情况检查结果：

- 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：

[←](#)
检查报告
2020-08-03 16:57:01 ▼

集群 cls-

检查时间 2020-08-03 16:57:01

检查结果 建议 1 项

资源状态

运行情况

运行情况检查结果

集群参数

Node配置

master配置合理性

正常 (0 / 0)

检查内容 ⓘ

node高可用

异常 (1 / 2)

检查内容 ⓘ

工作负载配置

Request Limit设置

正常

检查内容 ⓘ

版权所有：腾讯云计算（北京）有限责任公司

第101 共123页

风险推送

最近更新时间：2025-09-04 10:51:42

容器服务 TKE 现在可以通过与 [腾讯云事件总线 EB](#) 对接，支持自定义推送规则的配置。TKE 检测到的风险可以及时通过短信、站内信等方式推送给您，您可以及时处理相关风险。

相关组件配置

1. 登录 [容器服务控制台](#)，在左侧导航栏中选择**集群**。
2. 在集群列表中，单击目标集群 ID，进入集群详情页。
3. 在**组件管理**页面，检查 kubejarvis 版本是否大于或等于1.0.8。

说明：
如果版本低于1.0.8，请进行组件升级。

4. 运行命令 `kubectl edit kjc default`，并在 `exporters` 下添加以下配置：

```
exporters:
- config: null
  name: my-eb-test
  requirements:
  - diagnostic: hosting-apiserver-etcd-object-counts-check
    levels:
    - warn
    - risk
    - serious
  type: eb
```

推送规则配置

1. 如果您是第一次使用事件总线功能，请先参考文档 [开通事件总线](#)。
2. 登录 [腾讯云可观测平台](#)。
3. 在**事件总线 > 事件规则**页面，地域选择广州，事件集选择 default，单击**新建**。
4. 配置需要推送的事件。以集群资源数量超过推荐值/最大配额为例，配置如下：
 - 规则名称：按需求配置。
 - 事件匹配：
 - 事件模式：选择**表单模式**。
 - 云服务类型：容器服务。
 - 事件类型：以“集群资源数量超过推荐值”和“集群当前资源数量超过了最大配额”为例。

事件匹配 以 JSON 的方式编写事件匹配规则，并可参照事件示例进行测试，[规则编写指引](#)

事件模式：☒ 表单模式 ☐ 自定义事件

云服务类型：

事件类型：

☐ 节点内存将要耗尽

☐ 集群当前 APIServer 严重过载

☐ 节点Inode不足

☒ 集群当前资源数量超过了推荐

☒ 集群当前资源数量超过了最大

☐ 节点异常

事件匹配规则预览

事件匹配规则，对投递到 EventBridge 事件总线的事件进行过滤

```
1 {
2   "source": "tke.cloud.tencent",
3   "type": [
4     "tke:ErrorEvent:ApiserverEtcdObjectCountsWarn",
5     "tke:ErrorEvent:ApiserverEtcdObjectCountsSerious"
6   ]
7 }
```

JSON 语法合法

5. 单击**下一步**，配置通知相关参数：

事件目标

触发方式 *

消息推送①

消息模板 *

监控告警模板

通用通知模板

告警内容 *

中文

英文

通知方式 *

渠道推送

渠道推送

接收对象 *

用户

通知时段 *

09:30:00 ~ 23:30:00

接收渠道 *①

邮件

短信

电话

站内信

添加

立即启用事件规则

- **触发方式**：选择“消息推送”。
- **消息模板**：选择“通用通知模板”。
- **告警内容**：按需选择。
- **通知方式**：选择“渠道推送”。
- **接收对象**：您希望接收推送的账户。
- **接收渠道**：按需选择。

⚠ 注意：

- 使用限制：短信消息限制500字，电话限制350字。如果您的实例名等字段过长，可能导致消息因长度超限而发送失败，建议您同时配置多个渠道。
- 跨境进行接口回调可能存在因网络不稳定导致回调失败的情况，请您谨慎选择。

6. 单击**完成**。用户收到通知后，通知内容中会显示超限的资源名称、当前数量、推荐值，以及最大配额数量。如下图所示：

云服务 容器服务 事件通知

尊敬的腾讯云用户，您好！

您的腾讯云账号（账号 ID：332*****，昵称：[REDACTED] 云服务产品 容器服务 事件 tke:ErrorEvent:ApiserverEtcdObjectCountsWarn 已经产生，请及时关注并处理。

事件类型：tke:ErrorEvent:ApiserverEtcdObjectCountsWarn

产品类型：容器服务

资源 ID：uuid: [REDACTED] | Name: hosting-apiserver-etc
d-object-counts | resourceName: pods | currentValue: 161 | softNum: 150 | hardNum: 60
0 | instance: cls-[REDACTED]

资源地域：ap-[REDACTED]

事件产生时间：2023-06-30 17:40:14 (UTC+08:00)

查看更多详情，请登录腾讯云「事件总线」产品控制台查看与管理。

[进入事件总线控制台](#)

成本洞察和优化

成本洞察

最近更新时间：2024-08-06 17:22:51

功能介绍

传统的 Kubernetes 集群通常只关注资源层面的数据，例如节点的 Allocatable 标示可分配的资源，Metrics-Server 可以监控节点和 Pod 的利用率数据。TKE Insight 推出的成本可视化面板可以帮助用户更好地理解、监控和控制容器环境中的资源使用和成本情况。主要目标包括：

- **透明化**：用户可以直观地了解应用和服务在容器环境中的资源使用情况，包括不同集群、命名空间以及工作负载在 CPU 和内存方面的成本消耗。这有助于用户了解整体成本结构，提高决策效率。
- **成本控制与优化**：通过成本洞察，用户可以发现成本较高的应用或服务，并针对这些问题进行优化。例如，使用 [Request 智能推荐](#)，[可抢占式 Job](#)，[原生节点专用调度器](#)，[QoSAgent](#)，以减少不必要的开支。
- **预算管理**：用户可以根据实际成本情况制定更加合理的预算，并监控预算执行情况。当资源使用接近或超过预算时，用户可以及时调整策略，避免产生额外支出。

说明：
该功能仅适用于 TKE 标准集群。

操作步骤

1. 登录 [容器服务控制台](#)。
2. 选择左侧导航中的 **TKE Insight > 成本洞察**。

使用说明

- 成本洞察的成本数值是一个计算出的参考值，仅供参考，准确费用请登录腾讯云 [费用中心](#) 查看。
- 超级节点无法区分按量计费具体是普通按量计费还是竞价 Pod。因此目前不支持使用预留券抵消对应规格 Pod 的成本。原因是预留券需要在计算成本时调用接口查询预留券信息，放在监控组件中去调用会对接口造成较大压力。
- 如果上个月没有成本数据，则当月成本相较于上月成本的百分比将不会展示数据。
- 在计算成本时，Pod 的 GPU 使用量不进行计算，而是把 GPU 的价格分摊在 CPU 和内存上。即 GPU 节点上的 CPU、内存单价更高。
- 对于原生节点和普通节点的刊例价指标，在上报时，由于无法获取节点的计费模式，因此统一按照按量计费模式的单价去上报数据。
- 对于按量计费的超级节点上的 Pod，当 annotation 中指定了多个 GPU 类型时，使用第一个 GPU 类型去计算价格。

成本洞察功能说明

计费方式

计费项包含了普通节点（CVM）、原生节点和超级节点。

普通节点计费方式

根据节点机型和计费方式（按量计费或包年包月），找出节点的刊例价。机型对应的刊例价是通过调用 CVM 接口获取的，接口文档请参见 [获取可用区机型配置信息](#)。

说明：
机型从 node 的 annotation node.kubernetes.io/instance-type 中获取。

原生节点计费方式

根据节点机型和计费方式（按量计费或包年包月），找出节点的刊例价。

说明：
机型从 node 的 annotation node.kubernetes.io/instance-type 中获取。

超级节点计费方式

包年包月

根据机器规格（CPU 规格、内存规格），计算出节点的刊例价。

计算方式： $\text{CPU 规格} \times \text{CPU 单价} + \text{内存规格} \times \text{内存单价}$ 。包年包月超级节点的 CPU 单价和内存单价是固定的。

说明：

节点规格从 node 的 annotation `eks.tke.cloud.tencent.com/node-resource` 中获取。

按量计费

按量计费超级节点刊例价 = 对应超级节点上各个 Pod 所在子机刊例价总和。

Pod 所在子机刊例价计算方式如下：

- 解析 Pod 所在子机的 CPU 类型和 GPU 类型。如果 CPU 类型指定了多个，则以第一个类型为依据进行价格计算。如果 GPU 类型指定了多个，则以第一个类型为依据进行价格计算。
- 如果 GPU 类型不为空，则查找对应 GPU 类型的 CPU 单价、内存单价、GPU 单价和 license 单价（如果 license 不收费，则单价为 0 元）。如果 GPU 类型为空，则查找对应 CPU 类型的 CPU 单价和内存单价。
- 解析 Pod 所在子机的规格（即 CPU 规格、内存规格、GPU 规格），详情请参见 [资源规格计算方法](#) 和 [资源规格](#)。
- 计算 Pod 的刊例价： $\text{GPU 规格} \times \text{GPU 单价} + \text{CPU 规格} \times \text{CPU 单价} + \text{内存规格} \times \text{内存单价} + \text{license 单价}$

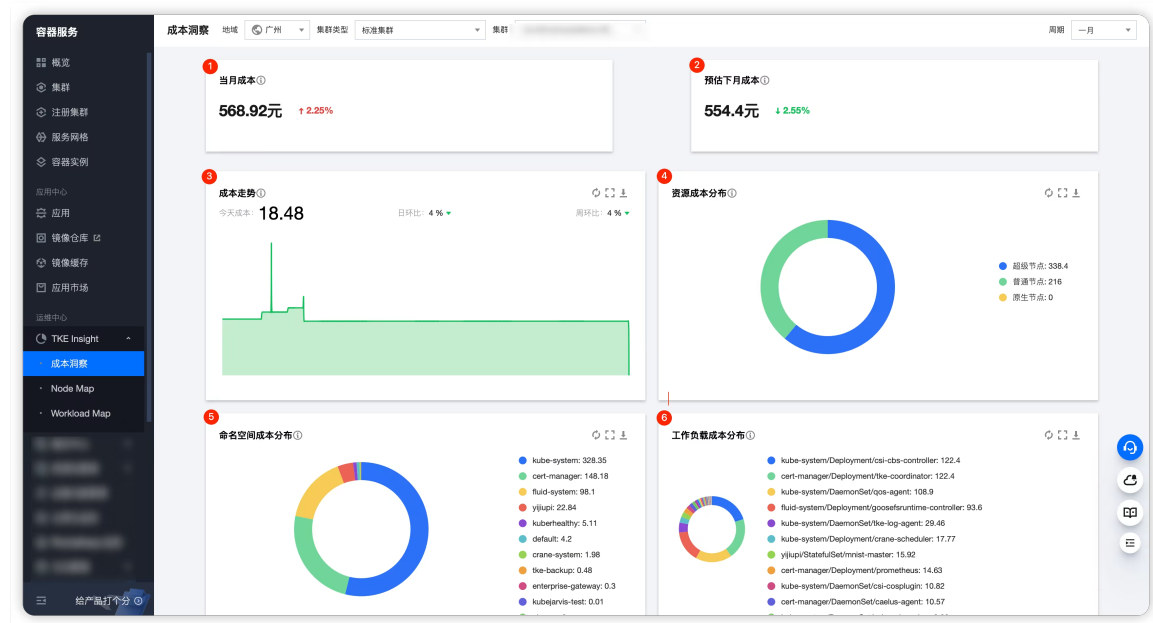
说明：

- CPU 类型从 Pod 的 annotation `eks.tke.cloud.tencent.com/cpu-type` 中获取。
- GPU 类型从 Pod 的 annotation `eks.tke.cloud.tencent.com/gpu-type` 中获取。

节点刊例价上报到腾讯云可观测平台

- 针对每一个节点，会将节点的刊例价信息上报到 [腾讯云可观测平台 TCOP](#)。当前支持 5min 和 1h 粒度，5min 粒度表示节点 5min 的价格，1h 粒度表示节点 1h 的价格。
- 针对按量计费超级节点上的 Pod，会上报各个 Pod 的刊例价格信息。当前支持 5min 和 1h 粒度，5min 粒度表示 Pod 所在子机 5min 的价格，1h 粒度表示 Pod 所在子机 1h 的价格。

视图功能说明



1. 当月成本

从腾讯云可观测平台 TCOP 拉取最近一个月普通节点、原生节点、超级节点的价格数据，进行相加，以得到一个总的价格。

2. 预估下月成本

从腾讯云可观测平台 TCOP 以小时粒度拉取最近1h普通节点、原生节点、超级节点价格信息。假设最近1h普通节点、原生节点和超级节点价格之和为 x 元，则预估的下月成本为 $x \times 24 \times 30$ 元。

3. 成本走势

- 今天成本：从监控中拉取最近一天普通节点、原生节点和超级节点的价格数据，进行相加。
- 当月成本走势：从云监控以小时粒度拉取最近一个月的价格数据，前端以小时粒度展示出最近一个月的成本趋势。

❗ 说明：

每个小时的数据点表示对应时间点普通节点、原生节点和超级节点刊例价总和。

4. 资源成本分布

分别拉取普通节点、原生节点、超级节点最近1h的成本，然后乘以24和30，得出各个节点的最近一个月的普通节点成本、原生节点成本和超级节点成本。

5. 命名空间成本分布

通过最近一小时的费用预估集群中每个命名空间一个月的成本。

1. 获取当前集群的所有节点（普通节点、原生节点和超级节点）。
2. 从腾讯云可观测平台 TCOP 拉取普通节点、原生节点和包年包月超级节点上各个 Pod 的 CPU 使用量和内存使用量。
3. 分别拉取到当前集群中 CVM 节点和原生节点的机型信息，并计算出各个机型的 CPU 单价和内存单价（超级节点 CPU 单价和内存单价是固定的，所以不用计算）。计算方式如下：
 - 3.1 根据全网所有机型的价格预估，得出 CPU 单价为 x 元/核/小时，内存单价为 y 元/G/小时。
 - 3.2 根据3.1得出 CPU 单价/内存单价的比例，然后将整个机器的价格按照内存进行计费，得出该机器内存单价，再将内存单价乘以比例得出 CPU 单价。
例如：CPU 单价/内存单价= r ，机器为 x 核 y GB，机器价格为 z ，则 CPU/内存单价计算方式：
 - 得出该机器内存单价： $z/(x*r+y)$
 - CPU 单价： $r*(z/(x*r+y))$
4. 根据2中获得的各个 Pod 的 CPU 使用量和内存使用量，再根据 Pod 所在节点的 CPU 单价和内存单价，得出各个 Pod 的价格。
5. 根据2中获得的按量计费超级节点，从腾讯云可观测平台 TCOP 获取按量计费超级节点上 Pod 的刊例价。
6. 根据4和5中计算出来的 Pod 成本信息，聚合出各个命名空间的成本信息。
7. 空闲成本：用当前所有节点的总成本 - 各个命名空间的成本之和。

6. 工作负载成本分布

通过最近一小时的费用预估集群中每个工作负载一个月的成本。

1. 获取当前集群的所有节点（普通节点、原生节点和超级节点）。
2. 从监控拉取普通节点、原生节点和包年包月超级节点上各个 Pod 的 CPU 使用量和内存使用量。
3. 分别拉取到当前集群中普通节点和原生节点的机型信息，并计算出各个机型的 CPU 单价和内存单价（超级节点 CPU 单价和内存单价是固定的，所以不用计算，CPU 单价：36元/核/小时，内存单价：18元/G/小时）。计算方式如下：
 - 3.1 根据成本大师的预估（根据全网所有机型的价格），得出 CPU 单价为 x 元/核/小时，内存单价为 y 元/G/小时。
 - 3.2 根据3.1得出 CPU 单价/内存单价的比例，然后将整个机器的价格按照内存进行计费，得出该机器内存单价，再将内存单价乘以比例得出 CPU 单价。
例如：CPU 单价/内存单价= r ，机器为 x 核 y GB，机器价格为 z ，则 CPU/内存单价计算方式：
 - 得出该机器内存单价： $z/(x*r+y)$
 - CPU 单价： $r*(z/(x*r+y))$
4. 根据2中获得的各个 Pod 的 CPU 使用量和内存使用量，再根据 Pod 所在节点的 CPU 单价和内存单价，得出各个 Pod 的价格。
5. 根据2中获得的按量计费超级节点，从腾讯云可观测平台 TCOP 获取按量计费超级节点上 Pod 的刊例价。
6. 根据4和5中计算出来的 Pod 成本信息，聚合出各个 Workload 的成本信息。
7. 空闲成本：用当前所有节点的总成本 - 各个 Workload 的成本之和。

Node Map

最近更新时间：2024-08-09 11:35:11

概述

传统的对象管理页面一般都是通过列表形式展开的，例如 TKE 集群里现有的资源对象列表，但列表的展示方式具备如下缺点：每一行都是一些抽象的数字，可读性不高；对象之间没有明确的标识区分；无法支持某种形态的排序。为此，腾讯云容器服务的 TKE Insight 的能力，通过可视化的形式，将用户所有的资源对象进行展示。该模块具备丰富的过滤查询、类型聚合、状态展示能力，可以帮助用户快速定位目标对象。

Node Map 主要通过可视化的页面展示节点的各项状态和指标，帮助用户了解当前集群节点的资源利用率、装箱率情况，辅助用户分析节点可能存在的问题。

注意事项

Node Map 不支持统计超级节点的各项指标。

使用方式

1. 登录 [容器服务控制台](#)。
2. 在左侧选择 TKE Insight > Node Map。
3. 在 Node Map 页面中，选择地域、集群类型和集群后，进行查看。
4. 右上角可选按照24小时、7天和30天数据，均值或峰值数据筛选，如下图所示：



Node Map 功能说明

Node Map 分为概览和资源对象 heatmap，如下图所示：

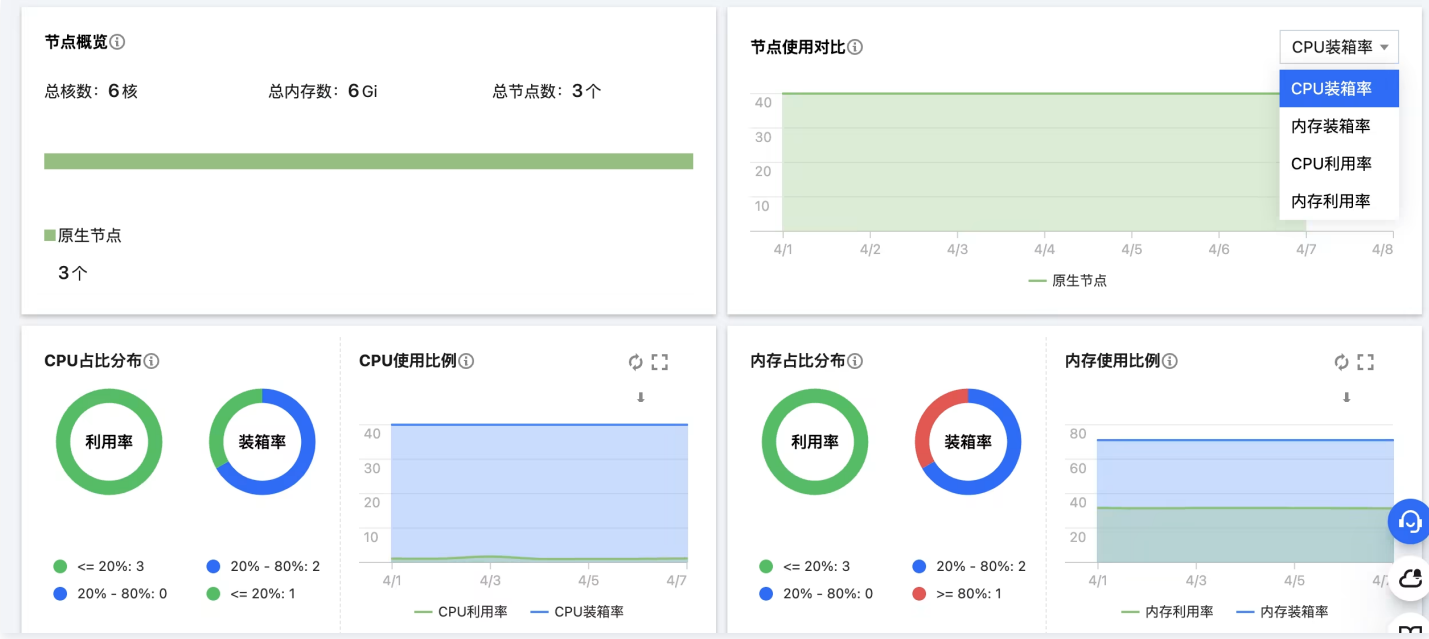


1. 上半区域是集群里的节点概览。

2. 下半区域是集群里的每一个节点资源对象。

集群/节点概览

集群/节点概览如下图所示：



节点概览

- 展示当前集群节点的总核数和总内存数。
- 展示当前集群中不同节点类型节点（原生节点、普通节点、注册节点）数量。

注意：
超级节点没有节点的概念，因此不在统计范围之内。

节点使用对比

对比不同类型节点（原生节点、普通节点、注册节点）CPU 装箱率、内存装箱率、CPU 利用率和内存利用率的情况。

节点使用资源分布

展示了集群中节点（原生节点、普通节点、注册节点）CPU /内存使用情况的分布。

注意：
超级节点没有节点的概念，因此不在统计范围之内。

指标	描述	说明
CPU 利用率分布	根据节点近24小时、7天或30天（右上角周期时间段选择）利用率均值、峰值查看集群中节点（原生节点、普通节点、注册节点）CPU 利用率分布情况。展示利用率在某一区间的节点个数。	CPU 利用率定义为节点真实 CPU 资源使用量除以节点规格。超级节点没有节点的概念，因此不在统计范围之内。
CPU 装箱率分布	根据节点近24小时、7天或30天（右上角周期时间段选择）装箱率均值、峰值查看集群中节点 CPU 装箱率分布情况。展示装箱率在某一区间的节点个数。	CPU 装箱率定义为节点上所有 Pod 的 CPU Request 之和除以节点规格。
内存利用率分布	根据节点近24小时、7天或30天（右上角周期时间段选择）利用率均值、峰值查看集群中节点内存利用率分布情况。展示利用率在某一区间的节点个数。	内存利用率定义为节点真实内存资源使用量除以节点规格。

内存装箱率分布	根据节点近24小时、7天或30天（右上角周期时间段选择）装箱率均值、峰值查看集群中节点内存装箱率分布情况。展示装箱率在某一区间的节点个数。	内存装箱率定义为节点上所有 Pod 的内存 Request 之和除以节点规格。
CPU 使用比例	根据页面右上角的指定周期时间段查看集群整体 CPU 装箱率和利用率的走势情况。	● 利用率统计了集群中原生节点、普通节点、注册节点的利用率情况。 ● 装箱率统计了集群中原生节点、普通节点、注册节点的装箱率情况，超级节点没有节点的概念，因此不在统计范围之内。
内存使用比例	根据页面右上角的指定周期时间段查看集群整体内存装箱率和利用率的走势情况。	● 利用率统计了集群中原生节点、普通节点、注册节点的利用率情况。 ● 装箱率统计了集群中原生节点、普通节点、注册节点的装箱率情况，超级节点没有节点的概念，因此不在统计范围之内。

节点对象查看

节点对象如下图所示，您可以通过节点筛选、分组、状态进行节点的过滤和聚合展示。



- **筛选**：根据相关指标筛选得到目标节点，若您不选择，则是默认选择了所有的节点。支持过滤多个属性，取交集。
- **聚合**：对筛选出来的节点进行分组，每一组节点在同一个浅色框内，一个框内的每个节点具有相同的属性值。
- **状态**：您可以通过**节点状态**进行筛选，支持选择 **CPU 使用率**、**内存使用率**、**CPU 装箱率**和**内存装箱率**。
 - CPU 使用率：节点近24小时、7天或30天 CPU 利用率均值、峰值。
 - 内存使用率：节点近24小时、7天或30天内存利用率均值、峰值。
 - CPU 装箱率：节点近24小时、7天或30天 CPU 装箱率均值、峰值。
 - 内存装箱率：节点近24小时、7天或30天内存装箱率均值、峰值。

注意：
在节点对象热图中的节点，默认按照当前选择的“节点状态”属性值进行**升序排列**。例如，您选择的节点状态是“CPU 使用率”，则节点对象热图里的节点默认是按照节点近期（右上角周期时间段选择）CPU 使用率均值或峰值升序排列。

筛选后的节点由**绿、蓝、红**三色区分。您可以单击页面右下角的，调整三种状态的阈值范围。如下图所示：



节点操作

当您的鼠标悬停在节点对象上后，您可以查看节点详情，并进行如下操作：



1. 单击详情，查看当前节点的详情信息。如下图所示：



节点详情

功能开关

- Request 推荐：集群级别的功能开关。开启后，可以根据 Workload 的使用情况，给 Workload 推荐合适的 Request。详情请参见 [Request 推荐](#)。
- 原生节点专用调度器：集群级别的功能开关。开启后，可以虚拟放大原生节点规格，调度更多的 Pod。详情请参见 [原生节点专用调度器](#)。

节点信息

- 节点规格：节点的 CPU 和内存量。
- 装箱率：节点上所有 Pod 的 Request 之和除以节点规格（依据页面右上角的页面选择）。
- 24h 利用率均值：节点 CPU 和内存24小时均值利用率。
- 24h 利用率峰值：节点 CPU 和内存24小时峰值利用率。

节点走势图信息

- **CPU 使用比例：**
 - 根据右上角的指定周期时间段查看集群中节点 CPU 使用比例走势情况。
- **内存使用比例：**
 - 根据右上角的指定周期时间段查看集群中节点内存使用比例走势情况。

Pod 详情

● CPU 用量走势图

- **申请量：**Pod 的 Request 数值。
- **实际用量：**Pod 实际的用量。
- **推荐量：**Pod 推荐的用量。需要提前开启 [Request 推荐](#)。

● Pod 信息

- **Pod 名称：**Pod 的名字。
- **Pod 状态：**Pod 的状态。
- **所属 Workload：**当前 Pod 所属的 Workload。
- **Request：**黑字部分是当前 Pod 的 Request 配置，蓝字部分是当前 Pod 通过 Request 推荐，推荐的建议数值（需要打开 [Request 推荐组件](#)）。

⚠ 注意：

1. 单击蓝字部分推荐的 Request，您可以一键更新该 Pod 所属的 Deployment 的 Request 数值。
2. 一键更新 Request 能力属于原生节点专有功能，如果 Workload 有部分 Pod 不在原生节点上，需要先将 Pod 迁移到原生节点。详情请参见 [Request 推荐](#)。

2. 单击**驱逐**，将节点内的 Pod 从节点中驱逐到集群内其他节点。
3. 单击**移出**，将节点移出节点池。
 - 若节点在节点池：当前节点归属节点池管理，需要前往节点池页面处理。
 - 若节点不在节点池：可以将节点从集群中删除。若节点是按量计费的节点，支持同时销毁节点。
4. 单击**取消封锁**，将节点调度进新的 Pod。

💡 说明：

节点在驱逐之后，会自动封锁。单击**取消封锁**后，该节点可以继续调度进新的 Pod。

Workload Map

最近更新时间：2024-08-16 15:11:51

概述

传统的对象管理页面一般都是通过列表形式展开的，例如 TKE 集群里现有的资源对象列表，但列表的展示方式具备如下缺点：每一行都是一些抽象的数字，可读性不高；对象之间没有明确的标识区分；无法支持某种形态的排序。为此，腾讯云容器服务的 TKE Insight 的能力，通过可视化的形式，将用户所有的资源对象进行展示。该模块具备丰富的过滤查询、类型聚合、状态展示能力，可以帮助用户快速定位目标对象。

Workload Map 主要通过可视化的页面展示工作负载的各项状态和指标，帮助用户了解当前工作负载的配置量和实际使用情况，辅助用户分析工作负载可能存在的问题。

注意事项

Workload Map 不支持提供超级节点上的 Pod 推荐值的监控数据。在聚合监控指标时，如果有 Workload 的 Pod 在超级节点上，则在统计 Workload 和 Cluster 级别的推荐值时，会缺少部分数据。

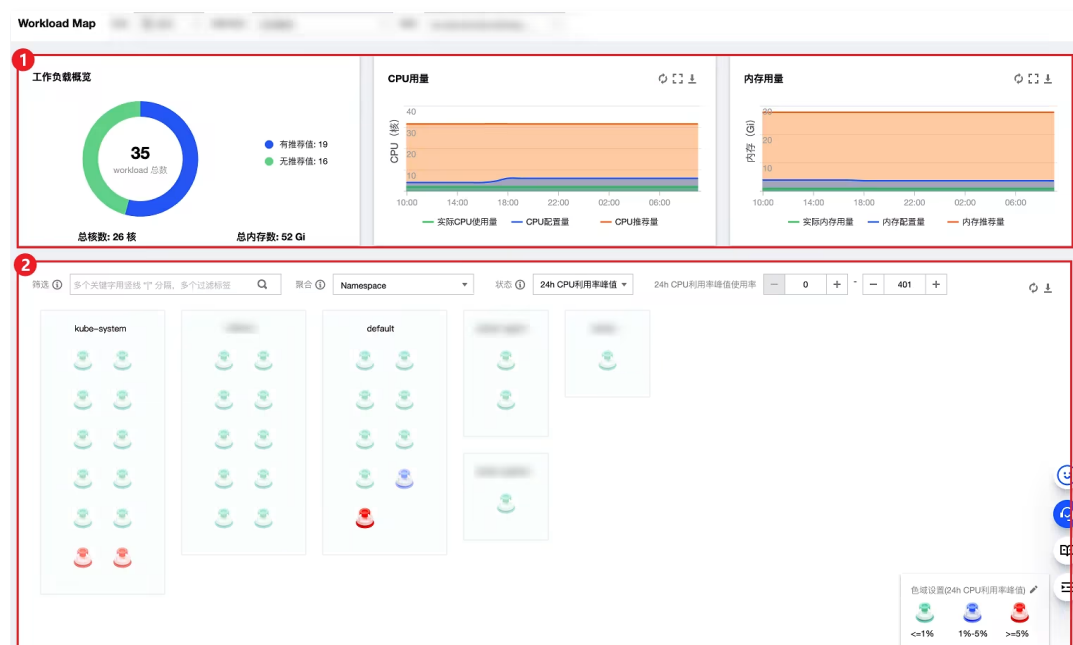
使用方式

1. 登录 [容器服务控制台](#)。
2. 在左侧选择 TKE Insight > Workload Map。
3. 在 Workload Map 页面中，选择地域、集群类型和集群后，进行查看。
4. 右上角可选24小时、7天和30天数据，如下图所示：



Workload Map 功能说明

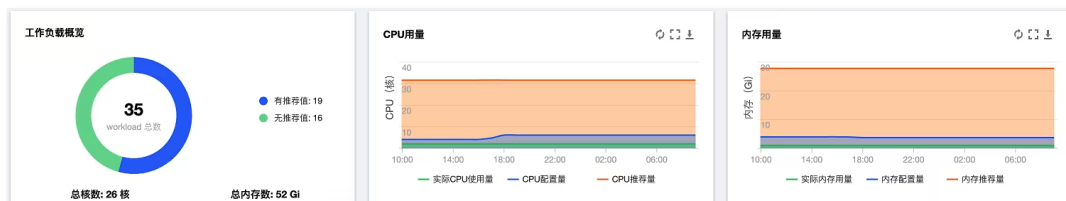
Workload Map 分为概览和资源对象热图，如下图所示：



1. 上半区域是集群里的 Workload 概览。
2. 下半区域是集群里的每一个 Workload 资源对象。

工作负载概览

工作负载概览如下图所示：



- **工作负载概览：**展示当前集群中是有推荐 Request 的工作负载的数量和所占比例；展示当前集群节点的总核数和总内存数。

⚠ 注意：

首先需要打开 Request 推荐功能，才会推荐工作负载调整 Request 数值。详情请参见 [Request 推荐](#)。

- **CPU 用量：**集群中工作负载的 CPU 用量情况。（统计了集群中原生节点、普通节点、注册节点上 Pod 的用量；只有均值）
 - **配置量：**集群中所有工作负载中所有 Pod 的 Request 的 CPU 之和。
 - **实际用量：**集群中所有工作负载中所有 Pod 实际的 CPU 用量之和。
 - **推荐量：**集群中所有工作负载中所有 Pod 推荐的 CPU 用量之和。需要提前开启 Request 推荐。
- **内存用量：**集群中工作负载的内存用量情况。（统计了集群中原生节点、普通节点、注册节点上 Pod 的用量；只有均值）
 - **配置量：**集群中所有工作负载中所有 Pod 的 Request 的内存之和。
 - **实际用量：**集群中所有工作负载中所有 Pod 实际的内存用量之和。
 - **推荐量：**集群中所有工作负载中所有 Pod 推荐的内存用量之和。需要提前开启 Request 推荐。

工作负载对象查看和操作

默认展示热力图视图，单击右上角**切换列表**可展示列表视图：



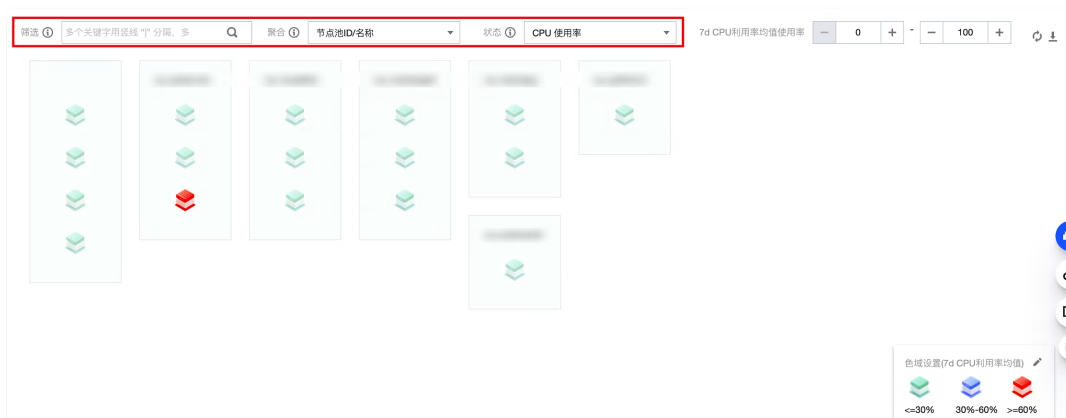
列表视图里，单击**切换图示**可切换热力图视图：



热力图视图

工作负载对象查看

工作负载对象如下图所示，您可以通过工作负载的筛选、聚合、状态进行 Workload 的过滤和聚合展示。（此处统计了集群中所有节点上的工作负载，包含了原生节点、普通节点、注册节点、超级节点。）



- **筛选：**根据相关指标筛选得到目标 Workload，若您不选择，则是默认选择了所有的 Workload。支持过滤多个属性，取交集。
- **聚合：**对筛选出来的 Workload 进行分组，每一组节点在同一个浅色框内，一个框内的每个 Workload 具有相同的属性值。

- **状态**：您可以通过**状态**进行筛选，支持选择 **CPU 使用率**和**内存使用率**。

- **CPU 使用率**：Workload CPU 利用率（CPU 实际使用量/配置Request用量）均值，根据页面右上角选择，分24小时、7天、30天数据。
- **内存 使用率**：Workload 内存 利用率（内存 实际使用量/配置Request用量）均值，根据页面右上角选择，分24小时、7天、30天数据。

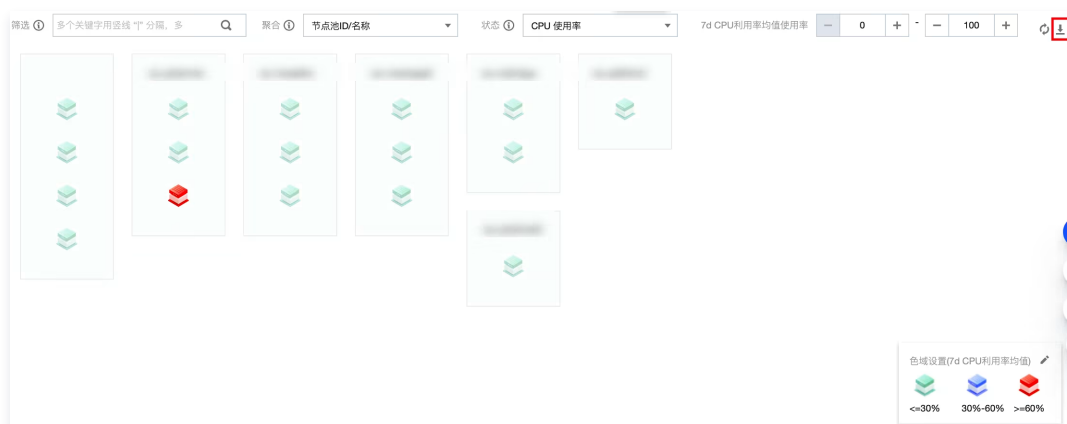
⚠ 注意：

在工作负载对象热图里的节点，默认按照当前选择的“状态”属性值进行**升序排列**。例如，您选择的工作负载状态是“CPU 使用率”，则工作负载对象热图里的工作负载默认是按照工作负载CPU 利用率均值升序排列。

筛选后的 Workload 由绿、蓝、红三色区分。您可以单击页面右下角的，调整三种状态的阈值范围。如下图所示：



- **下载**：您可以单击工作负载概览右侧的下载标识，下载本页 Workload 资源列表信息：

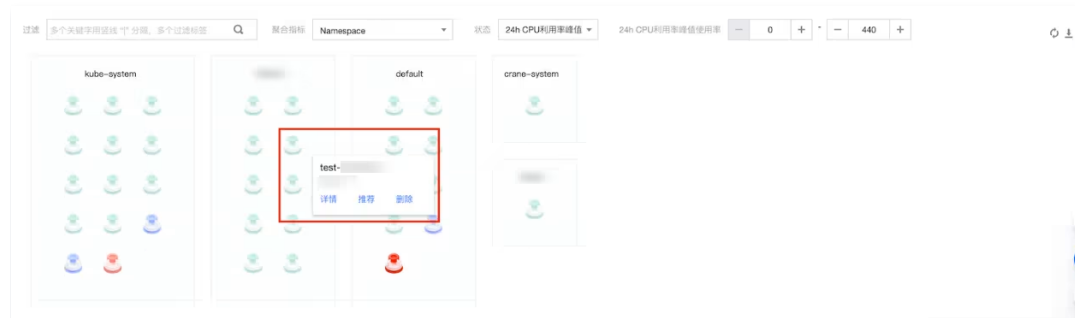


下载列表的字段说明：

- **名称**：工作负载的名称
- **命名空间**：工作负载所属的命名空间
- **工作负载类型**：工作负载所属的 K8s 类型
- **24h CPU 利用率均值**：工作负载近24小时 CPU 利用率的均值。单位：%
- **7d CPU 利用率均值**：工作负载近7天 CPU 利用率的均值。单位：%
- **30d CPU 利用率均值**：工作负载近30天 CPU 利用率的均值。单位：%
- **24h CPU 利用率峰值**：工作负载近24小时 CPU 利用率的峰值。单位：%
- **7d CPU 利用率峰值**：工作负载近7天 CPU 利用率的峰值。单位：%
- **30d CPU 利用率峰值**：工作负载近30天 CPU 利用率的峰值。单位：%
- **24h 内存利用率均值**：工作负载近24小时内内存利用率的均值。单位：%
- **7d 内存利用率均值**：工作负载近7天内内存利用率的均值。单位：%
- **30d 内存利用率均值**：工作负载近30天内内存利用率的均值。单位：%
- **24h 内存利用率峰值**：工作负载近24小时内内存利用率的峰值。单位：%
- **7d 内存利用率峰值**：工作负载近7天内内存利用率的峰值。单位：%
- **30d 内存利用率峰值**：工作负载近30天内内存利用率的峰值。单位：%

工作负载操作

当您的鼠标悬浮在工作负载对象上后，您可以查看当前工作负载的详情信息，并进行如下操作：



1. 单击详情，查看当前工作负载的详情信息。

工作负载详情

● 功能开关

- **Request 推荐**：集群级别的功能开关。开启后，可以根据 Workload 的使用情况，给 Workload 推荐合适的 Request。详情请参见 [Request 推荐](#)。
- **推荐值（蓝色字体）**：标识当前工作负载的 Request 推荐值，单击后支持更新（仅在开启 Request 推荐且有推荐值的情况下展示）。

● 工作负载信息

- **24h 利用率均值**：工作负载 CPU 和内存24小时均值利用率。
- **24h 利用率峰值**：工作负载 CPU 和内存24小时峰值利用率。

● 工作负载走势图信息

- **CPU 用量**：当前工作负载的 CPU 用量情况。
 - **申请量**：当前工作负载中所有 Pod 的 Request 的 CPU 数值之和。
 - **实际用量**：当前工作负载中所有 Pod 实际的 CPU 用量之和。
 - **推荐量**：当前工作负载中所有 Pod 推荐的 CPU 用量之和。需要提前开启 Request 推荐。
- **内存用量**：当前工作负载的内存用量情况。
 - **申请量**：当前工作负载中所有 Pod 的 Request 的内存数值之和。
 - **实际用量**：当前工作负载中所有 Pod 实际的内存用量之和。
 - **推荐量**：当前工作负载中所有 Pod 推荐的内存用量之和。需要提前开启 Request 推荐。

Pod 详情

● CPU 用量走势图

- **申请量**：Pod 的 Request 数值。
- **实际用量**：Pod 实际的用量。
- **推荐量**：Pod 推荐的用量。

● Pod 详情

- **Pod 名称**：Pod 的名字。
- **Pod 状态**：Pod 的状态。
- **所在节点**：当前 Pod 所属的节点。
- **节点类型**：当前 Pod 所属节点的节点类型。

2. 单击推荐，展示当前工作负载的 Request 推荐值（仅开启 Request 推荐且有推荐值的情况下，支持单击）。

3. 单击删除，将删除该工作负载。

列表视图

以列表展示某一类型的 Workload 在某些 namespace 下的详情信息：

切换图示

Deployment

所有命名空间

名称只能搜索一个关键字, Label格式要求:

Q

🔄

📄

名称	工作负载类型	命名空间	当前Request CPU / 内存 ①	推荐资源量 ①
craned	Deployment	crane-system	grpc-ds-server: 无限制 / 无限制 craned: 无限制 / 无限制	grpc-ds-server: 0.125 核 / 125 Mi craned: 0.125 核 / 125 Mi
wordpress	Deployment	default	wordpress: 0.3 核 / 512 Mi	wordpress: 0.125 核 / 250 Mi
coredns	Deployment	kube-system	coredns: 0.1 核 / 29.297 Mi	coredns: 0.125 核 / 125 Mi
crane-descheduler	Deployment	kube-system	descheduler: 0.2 核 / 200 Mi	descheduler: 0.125 核 / 125 Mi
crane-scheduler	Deployment	kube-system	scheduler: 0.2 核 / 200 Mi	scheduler: 0.125 核 / 125 Mi
crane-scheduler-controller	Deployment	kube-system	controller: 0.2 核 / 200 Mi	controller: 0.125 核 / 125 Mi
csi-cbs-controller	Deployment	kube-system	csi-provisioner: 0.1 核 / 50 Mi csi-attacher: 0.1 核 / 50 Mi csi-snapshotter: 0.05 核 / 25 Mi snapshot-controller: 0.05 核 / 25 Mi csi-resizer: 0.1 核 / 50 Mi cbs-csi: 0.1 核 / 50 Mi	csi-provisioner: 0.125 核 / 125 Mi csi-attacher: 0.125 核 / 125 Mi csi-snapshotter: 0.125 核 / 125 Mi snapshot-controller: 0.125 核 / 125 Mi csi-resizer: 0.125 核 / 125 Mi cbs-csi: 0.125 核 / 125 Mi
l7-lb-controller	Deployment	kube-system	l7-lb-controller: 无限制 / 无限制	l7-lb-controller: 0.125 核 / 125 Mi

共 8 条

20 条 / 页

⏪

⏴

1

⏵

⏩

🔄

📄

🔍

☰

1. 单击[下载](#)，下载列表展示的 csv 格式数据。
2. 单击[名称](#)，展示该 Workload 的详情信息。
3. 单击[推荐资源量](#)，可按照推荐资源量更新 Workload 的 Request 配置。
4. 针对工作负载副本数为0、首次运行时间不超过24小时情况，资源推荐量可能为空，更多信息可参见 [Request 智能推荐](#)。

成本优化

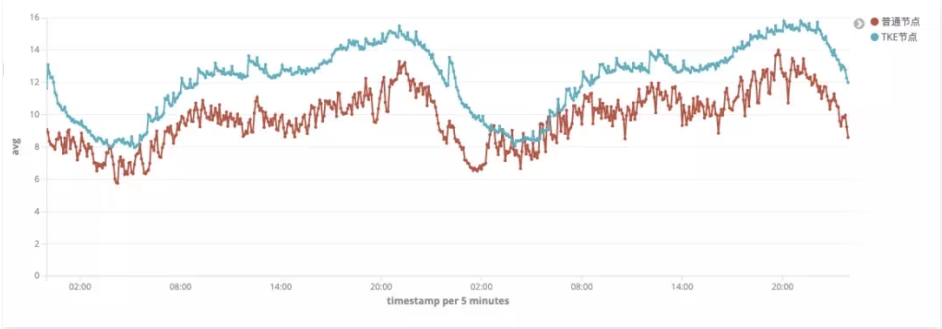
Request 智能推荐

最近更新时间：2024-11-05 14:34:52

简介

组件介绍

Kubernetes 可以显著提升业务编排能力和资源利用率，但如果没有额外的支持，其提升效果会非常有限。根据 TKE 团队之前的统计数据（参见[Kubernetes 降本增效标准指南 | 容器化计算资源利用率现象剖析](#)），TKE 节点的资源平均利用率仅为 14%左右，如下图所示：



Kubernetes 集群的资源利用率不高的主要原因是根据 Kubernetes 的资源调度逻辑，在创建 Kubernetes 工作负载时，通常需要为工作负载配置合适的资源 Request 和 Limit，表示对资源的占用和限制，其中对利用率影响最大的是 Request。为了防止自己的工作负载资源被其他工作负载占用，或者为了应对高峰流量时的资源消耗需求，用户习惯于为 Request 设置较大的数值。Request 和实际使用资源之间的差值是不能被其他工作负载使用的，因此造成了资源浪费。Request 数值设置不合理，导致了 Kubernetes 集群资源利用率低下。

容器服务 TKE 支持在集群中安装 Request 智能推荐组件。该组件可以为 Kubernetes 的 Workload 推荐容器级别的 Request/Limit 数值，从而减少资源浪费。该功能仅适用于标准集群，会对标准集群下所有的 Workload 定时更新推荐的 Request 配置。用户可以按需配置一键更新，但更新后 Pod 会重新调度到原生节点上。

部署在集群内的资源对象

开启集群的 Request 智能推荐，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
analytics.analysis.crane.io	CustomResourceDefinition	-	-
recommendations.analysis.crane.io	CustomResourceDefinition	-	-
crane-system	Namespace	-	-
housekeeper-default	Analytics	-	crane-system
recommendation-config	ConfigMap	-	crane-system
craned	ClusterRole	-	-
craned	ClusterRoleBinding	-	-
craned	Service	-	crane-system
craned	ServiceAccount	-	crane-system
craned	Deployment	-	crane-system

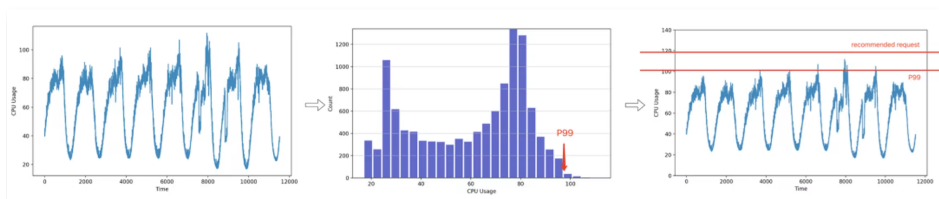
功能说明

- 支持为 Deployment、StatefulSet、DaemonSet 中的每一个 Container 智能推荐合适的资源 Request/Limit。
- 支持一键更新：使用推荐值一键更新 Workload 中 Container 的资源值。

- 支持维持 Request/Limit 比例：推荐的 Request/Limit 会维持初始 Workload 中 Container 设置的 Request/Limit 之间的比例，若 Limit 在创建 Workload 时没有设置，则不会推荐 Limit。
- 控制台的 Request 推荐一键更新能力会默认给工作负载加上 nodeSelector 的属性，Workload 在更新时，Pod 将只能调度到原生节点上，若原生节点资源不足，会引发 Pod 的 Pending。

Request 推荐原理

- 组件在 crane-system 命名空间下创建 Analytics CR 对象，默认覆盖集群中的 Kubernetes 原生工作负载（Deployment、StatefulSet），会分析工作负载最长 14 天的监控数据，12 小时更新一次推荐值。
- 然后根据 Analytics 生成集群内每个工作负载的 Recommendation CR 对象，用于存储推荐的数据。
- Recommendation CR 如果产生了推荐数据，就会把推荐数据写入到对应工作负载的 Annotation 里。



注意事项

环境要求

Kubernetes 版本：1.10+

节点要求

容器服务控制台中一键更新 Workload Request 功能会将工作负载迁移至 [原生节点](#)，若您的集群原生节点上资源不足，会导致 Pod 发生 Pending。

被控资源要求

- 支持 Deployment、StatefulSet。
- 不支持 Job、CronJob，不支持不是由 Workload 管理的 Pod。

推荐阈值

推荐最小值：单个容器推荐的 CPU 最小值是 0.125 核，即 125m；内存的最小值是 125Mi。

使用说明

安装组件

- 登录 [容器服务控制台](#)。
- 在左侧选择 TKE Insight > Node Map。

说明：
您也可以在 TKE Insight > Workload Map 中进行安装。

- 在 Node Map 页面中，鼠标悬浮到页面下方某一个 Node 上，单击详情。
- 在该 Node 的详情页的右上角，打开“Request 推荐”开关，进行调度器的参数配置。



注意：
该功能是集群级别的全局开关，开启后，会自动分析工作负载历史的监控数据，推荐合适的 Request 数值。

- 开启后非立即生效，为准确计算推荐值，需要分析该 Workload 的历史资源使用数据。
- 不同的 Workload 的计算时间长度可能不一致，集群中不同的 Workload 之间互相可能会有影响。
- 开启该功能后，对至少运行一天的 Workload 产生推荐数据。
- 对于开启功能后新建的 Workload，一般情况下，也需要一天的时间才会产生 Workload 的推荐数据。
- 建议工作负载稳定运行一段时间之后，再使用推荐值更新 Workload。

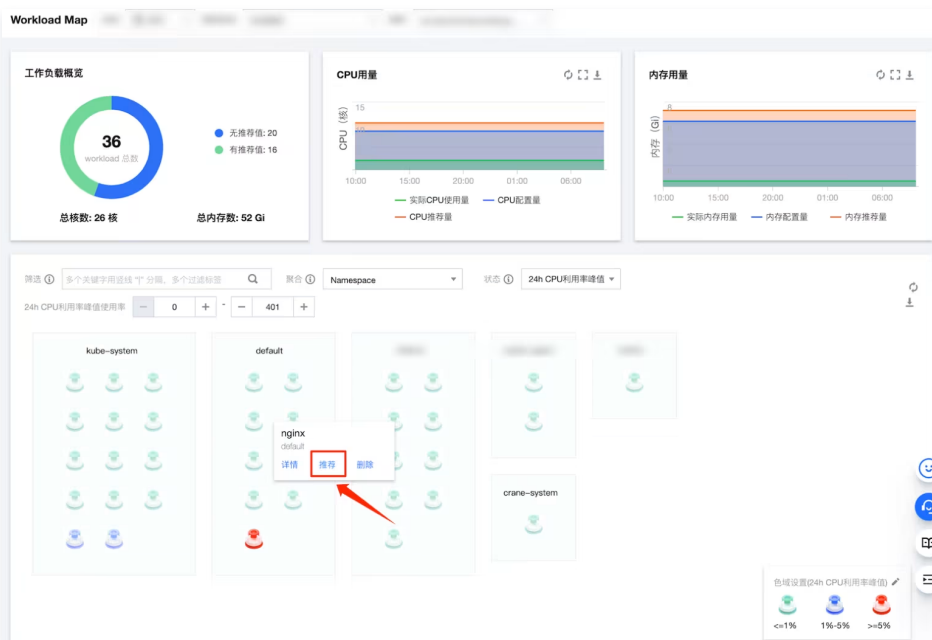
使用组件

1. 登录 [容器服务控制台](#)。
2. 在左侧选择 TKE Insight > Workload Map。

说明：

Workload Map 主要通过可视化的页面展示工作负载的各项状态和指标，帮助用户了解当前工作负载的配置量和实际使用情况，辅助用户分析工作负载可能存在的问题。更多可参考文档 [Workload Map](#)。

3. 在 Workload Map 页面，鼠标悬浮到页面下方某一个 Workload 上，单击推荐。



4. 在弹窗中，单击确认，即可使用推荐的 Request 数值更新原始 Workload 里面的数值。

说明：

容器服务控制台中一键更新 Workload Request 功能会将工作负载迁移至 [原生节点](#)，若您的集群原生节点上资源不足，会导致 Pod 发生 Pending。

后台获取推荐数值

Request 智能推荐组件会将每个工作负载的推荐值保存在该工作负载的 YAML 里，您可以通过标准的 Kubernetes API 获取每个工作负载的推荐值，然后集成到业务的发布系统中。如下所示查看工作负载下每个容器的 Request 推荐量：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    analysis.crane.io/resource-recommendation: |
      containers:
        # 若一个 Pod 里有多个容器，每个容器都有 CPU 和 Memory 的 Request 的推荐值
        - containerName: nginx
```

```
target:
  cpu: 125m
  memory: 125Mi #若这里缺少单位，显示的是字符串"58243235"，省略的单位是byte
```

⚠ 注意:

组件本身不会推荐 Limit，在控制台使用 Request 推荐值更新 Workload 时，会维持该 Workload Request 和 Limit 的比值以保证 QoS 不会发生变化。您如果在后台获取到 Request 推荐值，可以作为参考更新原始 Workload 的资源配置量。

组件权限说明

权限说明

该组件权限是当前功能实现的最小权限依赖。

权限场景

功能	涉及对象	涉及操作权限
记录 pod 的 oom 记录	pod	get/list/watch
根据 node 查找空闲节点并进行推荐	node	get/list/watch
需要更新 workload 的 annotations 记录推荐相关的信息	workload	get/list/watch
需要将异常信息以事件进行记录	event	create/patch/update
监听推荐相关资源的变更，进行资源推荐	analysis.crane.io	所有权限

权限定义

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: craned
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  - pods
  - nodes
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - analysis.crane.io
  resources:
  - "*"
  verbs:
  - "*"
- apiGroups:
  - apps
  resources:
  - daemonsets
  - deployments
  - deployments/scale
  - statefulsets
  - statefulsets/scale
  verbs:
```

```

- get
- list
- watch
- apiGroups:
  - apps
resources:
  - daemonsets/status
  - deployments/status
  - deployments/scale
  - statefulsets/status
  - statefulsets/scale
verbs:
  - update
- apiGroups:
  - autoscaling
resources:
  - horizontalpodautoscalers
verbs:
  - '*'
- apiGroups:
  - autoscaling.crane.io
resources:
  - '*'
verbs:
  - '*'
- apiGroups:
  - ""
resources:
  - events
verbs:
  - create
  - patch
  - update
- apiGroups:
  - prediction.crane.io
resources:
  - '*'
verbs:
  - '*'
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: craned
  namespace: crane-system
rules:
  - apiGroups:
    - ""
    resources:
      - configmaps
      - secrets
    verbs:
      - create
  - apiGroups:
    - ""
    resourceNames:
      - craned
    resources:
      - configmaps

```

```
verbs:
  - get
  - patch
  - update
- apiGroups:
  - ""
  resourceName:
    - clusters-secret-store
  resources:
    - secrets
  verbs:
    - get
- apiGroups:
  - coordination.k8s.io
  resources:
    - leases
  verbs:
    - get
    - patch
    - update
    - create
```