腾讯云可观测平台

快速入门

版权所有：腾讯云计算（北京）有限责任公司

【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

快速入门
- 云产品监控
- 应用性能监控
- 终端性能监控
- 前端性能监控
- 云拨测
- 云压测
- ElasticSearch 监控指南
- Prometheus 监控服务
- Grafana 服务
- 创建 Dashboard
- 告警管理
- 事件总线
  - 开通事件总线
  - 快速投递自定义事件
  - 快速配置事件告警推送
- OpenTelemetry 前后链路打通

版权所有：腾讯云计算（北京）有限责任公司

快速入门
云产品监控

最近更新时间：2025-01-09 18:12:02

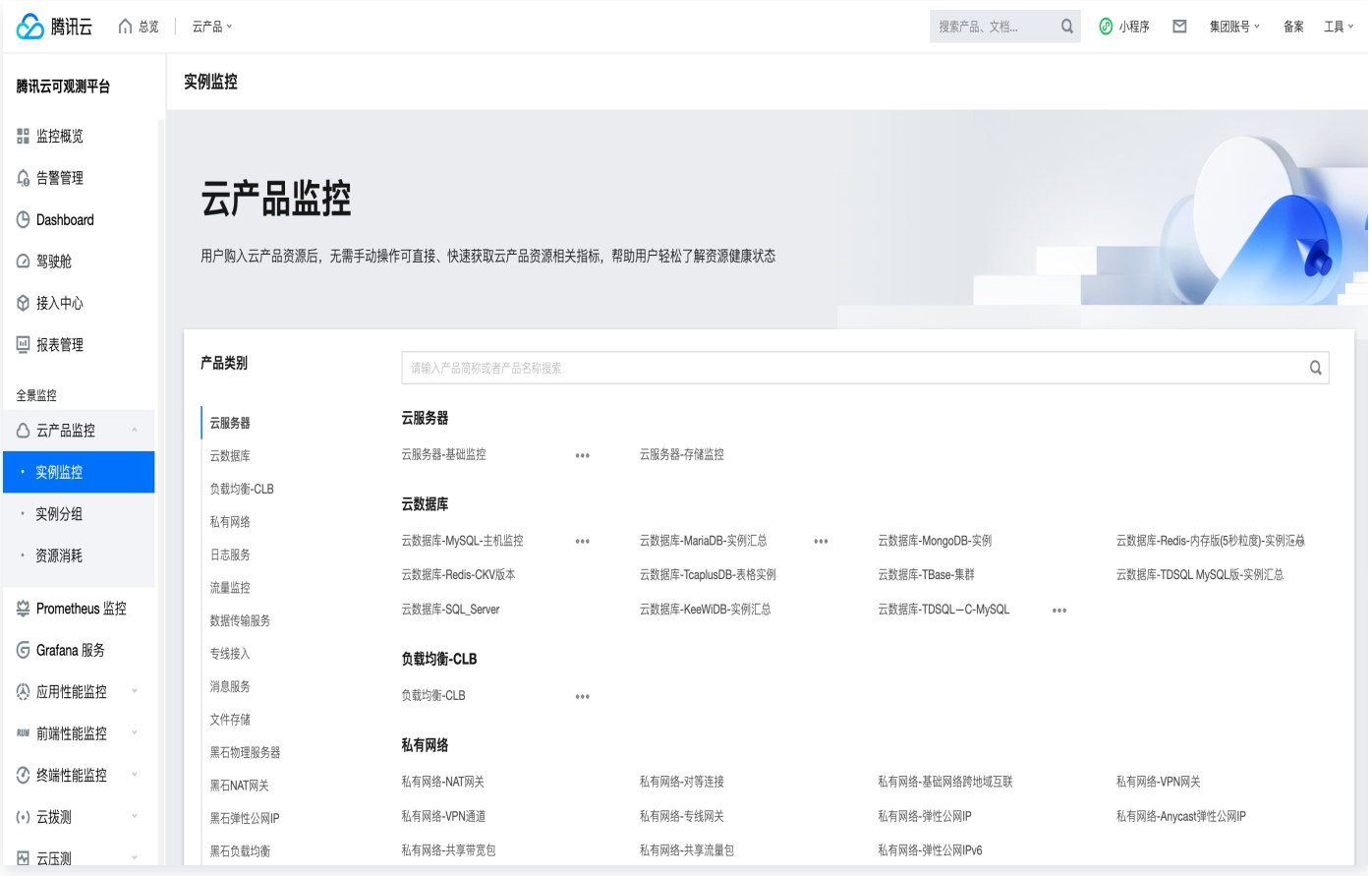
功能介绍

用户购入云产品资源后，无需手动操作，系统会直接、快速获取云产品资源相关指标，帮助用户轻松了解资源健康状态。云服务器监控和裸金属云服务器监控由于产品特性，需要安装相应的监控 Agent 并运行后方可采集监控数据，详情请参见 安装云服务器监控组件 。您也可在购买资源时选择进行自动安装。

操作步骤

> **说明**
> **云产品监控的使用需要用户提前购入云产品资源，并且已经创建云产品实例。**如需购买云产品可前往 腾讯云官网购买。

1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中单击**云产品监控 > 实例监控**。
3. 选择需要查看的云产品类别，点击进入对应产品实例监控页面，即可查看云产品实例监控详情。


腾讯云
总览
云产品
搜索产品、文档...
小程序
集团账号
备案
工具
腾讯云可观测平台
实例监控
监控概览
告警管理
Dashboard
驾驶舱
接入中心
报表管理
全景监控
云产品监控
实例监控
实例分组
资源消耗
Prometheus 监控
Grafana 服务
应用性能监控
前端性能监控
终端性能监控
云拨测
云压测
云产品监控
用户购入云产品资源后，无需手动操作可直接、快速获取云产品资源相关指标，帮助用户轻松了解资源健康状态
产品类别
请输入产品简称或者产品名称搜索
云服务器
云数据库
负载均衡-CLB
私有网络
日志服务
流量监控
数据传输服务
专线接入
消息服务
文件存储
黑石物理服务器
黑石NAT网关
黑石弹性公网IP
黑石负载均衡
云服务器
云服务器-基础监控
云服务器-存储监控
云数据库
云数据库-MySQL-主机监控
云数据库-MariaDB-实例汇总
云数据库-MongoDB-实例
云数据库-Redis-内存版(5秒粒度)-实例汇总
云数据库-Redis-CKV版本
云数据库-TcaplusDB-表格实例
云数据库-TBase-集群
云数据库-TDSQL MySQL版-实例汇总
云数据库-SQL_Server
云数据库-KeeWiDB-实例汇总
云数据库-TDSQL－C-MySQL
负载均衡-CLB
负载均衡-CLB
私有网络
私有网络-NAT网关
私有网络-对等连接
私有网络-基础网络跨地域互联
私有网络-VPN网关
私有网络-VPN通道
私有网络-专线网关
私有网络-弹性公网IP
私有网络-Anycast弹性公网IP
私有网络-共享带宽包
私有网络-共享流量包
私有网络-弹性公网IPv6

应用性能监控

最近更新时间：2025-05-12 14:09:32

本文将为您介绍如何快速使用应用性能监控。

步骤1：登录注册

登录 腾讯云官网。如果没有账号，请参见 账号注册。

步骤2：授予 APM 预设策略

应用性能监控 APM 默认创建了预设策略 `QcloudAPMFullAccess` （应用性能监控（APM）全读写访问权限）和 `QcloudAPMReadOnlyFullAccess` （应用性监控（APM）只读访问权限），您可以通过搜索策略名称快速进行预设策略授权，更多信息请参见 策略授予 。
在大数多使用场景下，只需要对子用户授予 APM 预设策略，就可以让子用户正常访问应用性能监控 APM。如果需要根据具体的资源和操作进行精细化授权，请参见 自定义策略 进行操作。

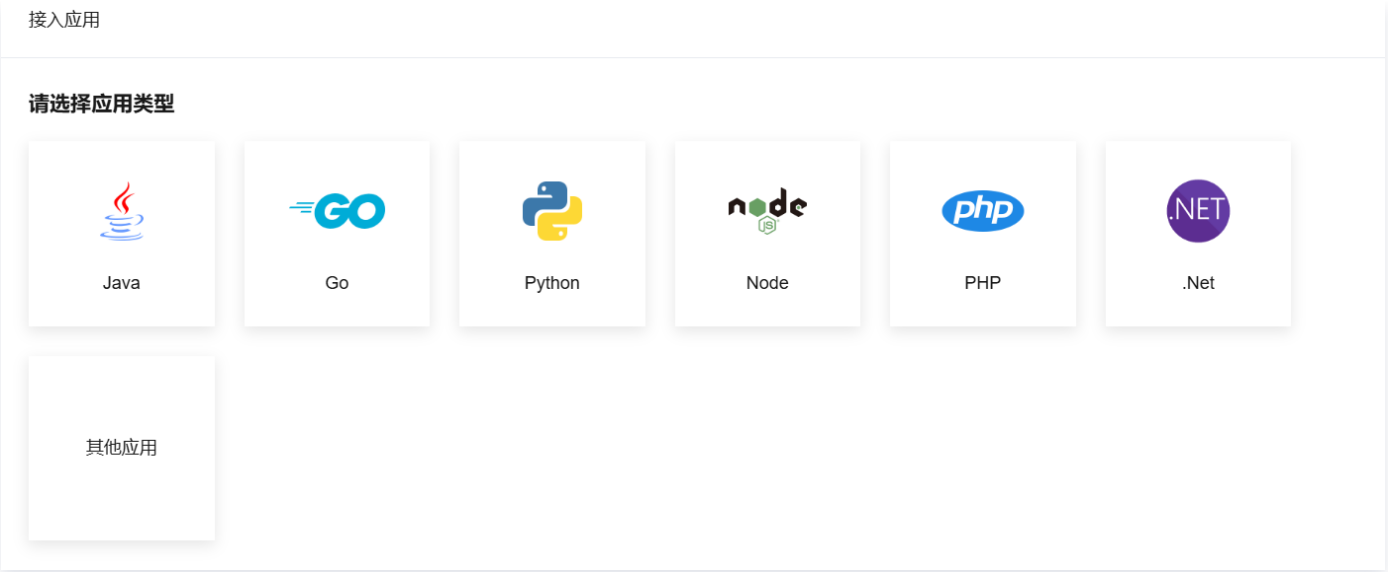
步骤3：创建业务系统

业务系统用于分类管理应用，每个业务系统有唯一的 Token，应用接入的时候需要指定 Token。可以在业务系统级别设置存储时长、上报限额等参数，也可以基于业务系统实现权限管理和分账，不同业务系统之间的监控数据完全隔离。关于业务系统的划分，请参见 如何划分业务系统 。

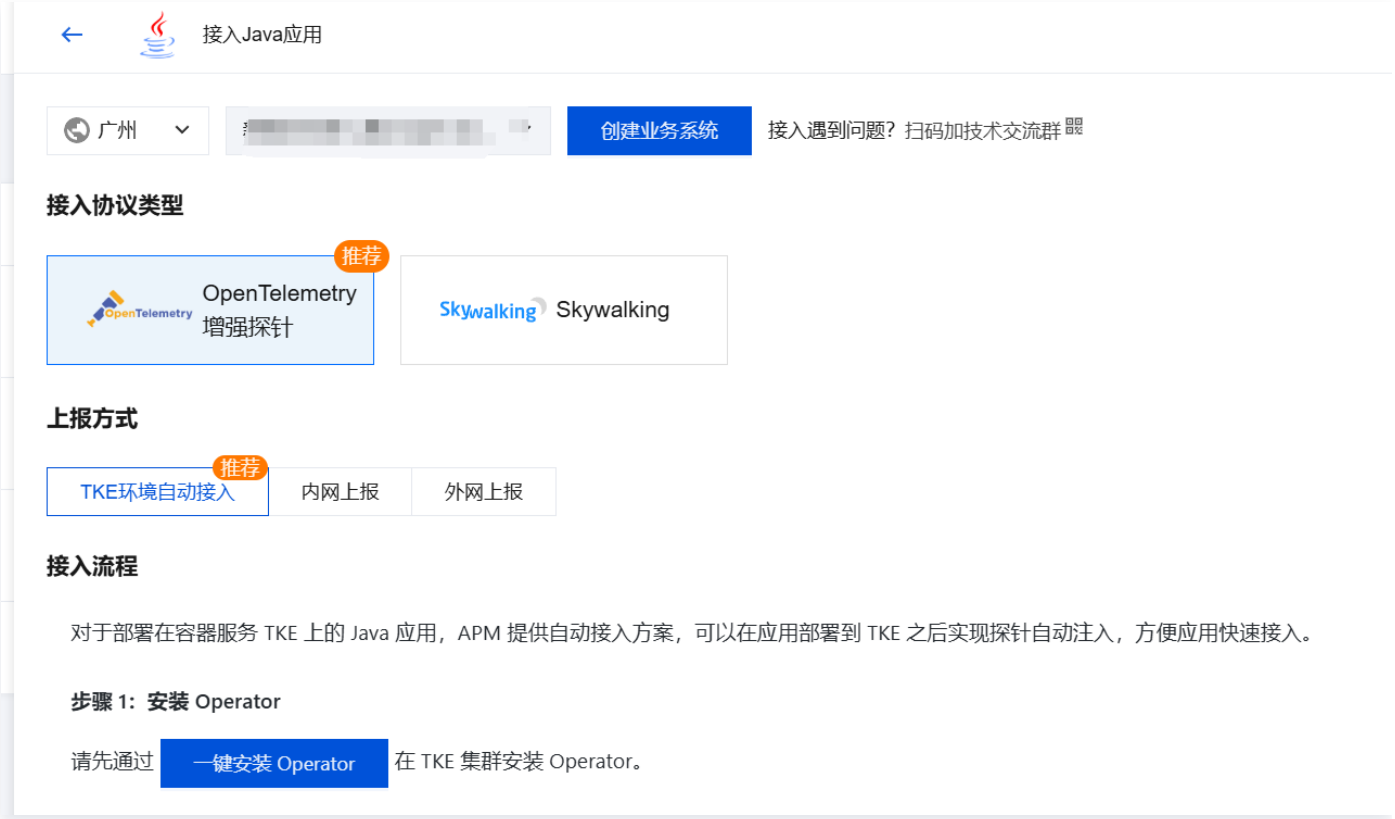
1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中选择**应用性能监控** > **资源管理**。
3. 在**业务系统管理**页中选择对应的地域，单击**新建**，按照提示填写业务名称，选择计费模式、上报地域、链路存储时长等信息，勾选相关协议并单击**确定**。

步骤4：接入应用

1. 在应用列表管理页单击**接入应用**。
2. 在数据接入页选择接入语言。



3. 在接入指南页选择**接入协议类型**和**上报方式**。



4. 根据页面指引完成应用接入，单击**完成**即可完成应用接入。

> **说明：**
> 应用性能监控（APM）提供了多种应用接入方式，以适配不同的编程语言以及应用部署环境，详情请参见 应用接入概述 。

终端性能监控

最近更新时间：2024-11-11 10:15:12

本文将为您介绍如何快速使用终端性能监控。

步骤1：创建业务系统

1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中单击**终端性能监控 > 应用管理 > 业务系统**。
3. 在业务系统管理页单击**创建业务系统**，在弹框中填写业务名称并勾选相关协议即可。

步骤2：接入应用

1. 在左侧菜单栏中单击**前端性能监控 > 数据总览**。
2. 在数据总览页单击**应用接入**，根据下列表格配置应用信息。

配置项	说明
应用名称	自定义应用名称，方便您在终端监控平台辨识该应用。
应用类型	支持 Web 、小程序（微信、QQ）、Hippy、Weex 、 React Native、Flutter、Cocos 应用类型接入。
所属业务系统	该功能用于分类管理您接入的应用，您可以根据研发团队、业务逻辑、应用类别等进行应用分类管理。若您没有可用团队，您可以单击右侧的**创建**链接，填写完信息后，单击**确认**即创建成功。

3. 配置完后单击**完成**即可完成应用初步接入操作，或者单击**接入文档查看**配置**安卓/iOS** 应用接入的详细信息。

> **说明**
> 按照上述步骤接入后即可使用崩溃分析、ANR 分析、卡曼分析、慢启动分析和网络等问题求分析。如需使用 ANR 分析、卡曼分析、慢启动分析和网络和 webview 功能，需参考可接入的指引上报数据。

步骤3：查看监控数据

应用接入成功且有一定的数据上报后，您可以前往 终端性能控制台 查看崩溃数据分析和网络性能、ANR 等监控数据。

前端性能监控

最近更新时间：2024-05-08 17:33:01

本文将为您介绍如何快速使用前端性能监控。

步骤1：创建业务系统

1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中单击**前端性能监控 > 应用管理 > 业务系统**。
3. 在业务系统管理页单击**创建业务系统**，在弹框中填写业务名称并勾选相关协议即可。

步骤2：接入应用

1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中单击**前端性能监控 > 数据总览**。
3. 在数据总览页单击**应用接入**，根据下列表格配置应用信息。

配置项	说明
应用名称	自定义应用名称，方便您在前端监控平台辨识该应用。
应用描述	填写应用描述 ，如应用用途、应用简介等，方便用户了解该应用。
应用类型	支持 Web 、小程序（微信、QQ）、Hippy、Weex 、 React Native、Flutter、Cocos 应用类型接入。
应用仓库地址（可选）	填写您的应用仓库地址，可不填写。
数据上报域名	* 表示不对上报域名进行校验。
所属业务系统	该功能用于分类管理您接入的应用，您可以根据研发团队、业务逻辑、应用类别等进行应用分类管理。若您没有可用团队，您可以单击右侧的**创建**链接，填写完信息后，单击**购买**即创建成功。

4. 配置完后单击下一步，参见下列说明选择一种方式安装 SDK 。

- **npm** 方式安装 SDK（所有应用类型均可使用该方式接入）。下列 Web 应用为例说明如何通过 npm 方式接入 SDK。
  i. 在接入指引页面中复制提供的首行命令，引入 npm 包。
  ii. 在接入指引页面中复制提供的代码初始化 SDK。

- **<script> 标签引入方式接入 SDK（仅支持 Web 接入类型）。**
  i. 在接入指引页面复制提供的 `<script>` 标签 代码。
  ii. 把<script> 标签引入类型下的代码引入到 `<head></head>` 标签中即可。

说明

> 按照上述步骤接入后即可使用数据总览、页面性能、异常分析、页面访问（PV、UV）、API 监控和静态资源功能。如需使用日志查询、自定义测速和自定义事件，需参考接入指引上报数据。

步骤3：查看监控数据

应用接入成功且有一定的数据上报后，您可以前往 前端性能监控控制台 查看异常分析和页面性能、页面访问等监控数据。

云拨测

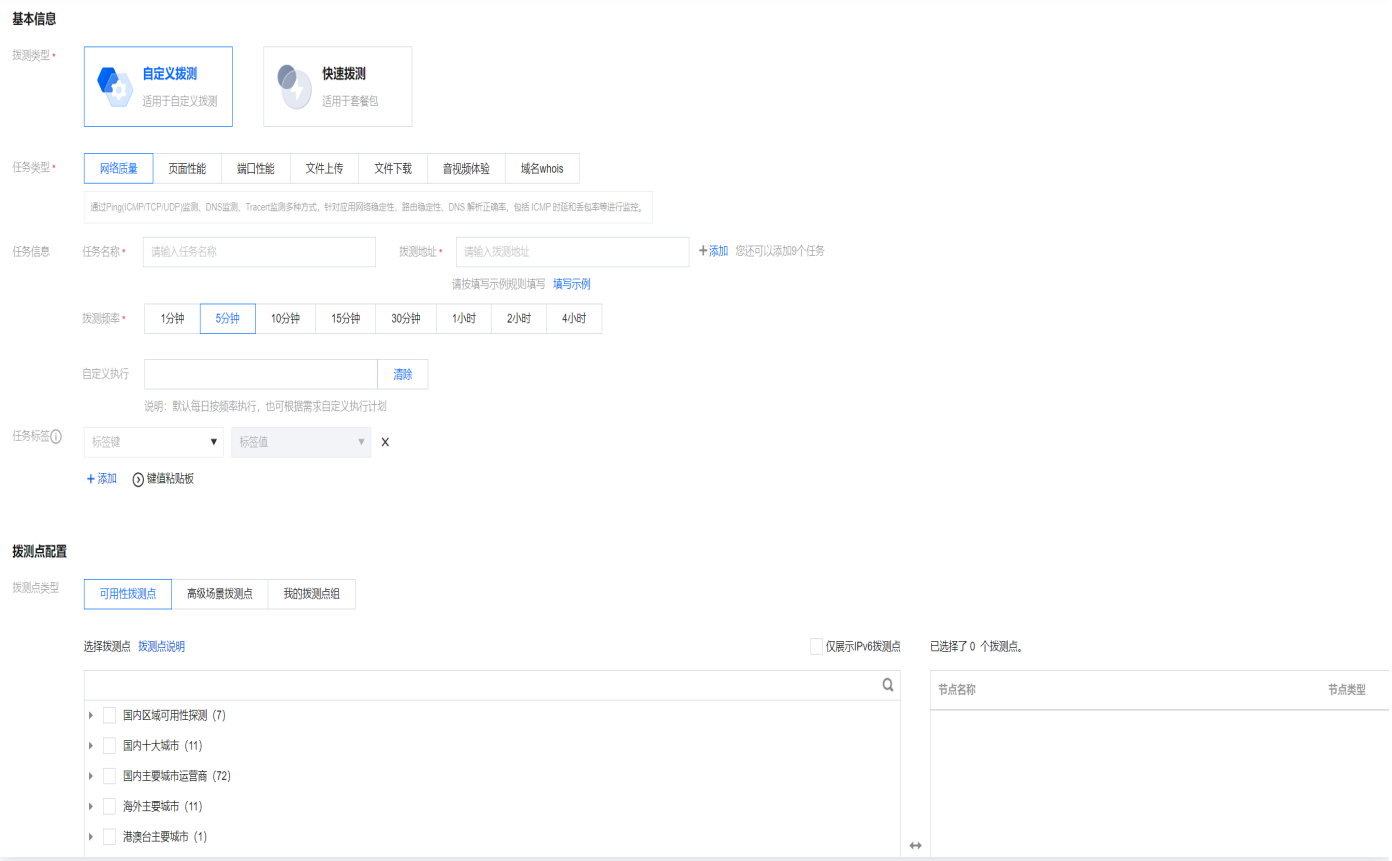
最近更新时间：2024-05-22 11:00:41

本文将为您介绍如何快速使用云拨测功能。

步骤1：新建任务

1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中单击**云拨测 > 任务列表**。
3. 在任务列表页，单击任务列表页面上方的**新建任务**。
4. 进入新建任务页面，根据提示选择拨测类型、任务类型，填写任务名称、拨测地址，并选择拨测频率和拨测点，填写完后单击**创建任务**即可。

> **说明：**
> 如需了解更多新建任务配置信息，请参见 新建自定义拨测 或 新建快速拨测任务 。

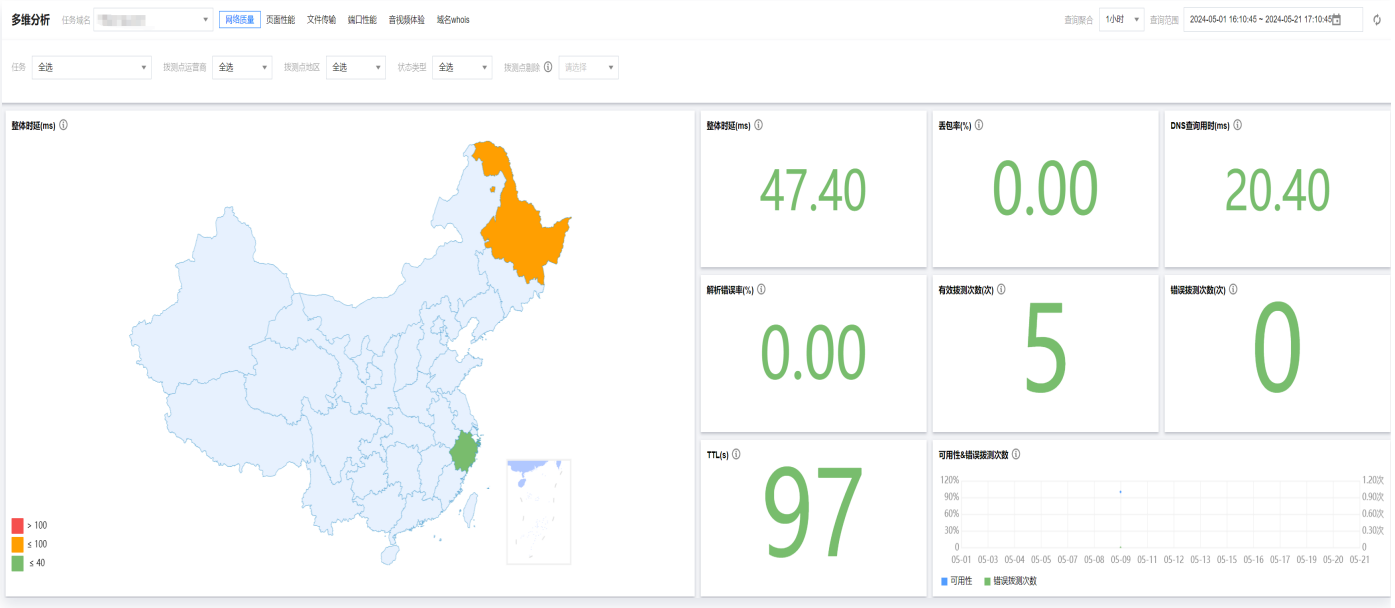


步骤2：多维分析拨测性能数据

1. 进入多维分析页面。

2. 根据您的实际情况筛选拨测域名、拨测类型、时间跨度等信息即可查看拨测任务关键指标数据，还可以查看单个拨测任务详情。

> **说明：**
> 如需了解更多多维分析能力，请参见 多维分析 。



云压测

最近更新时间：2024-08-16 11:52:01

功能介绍

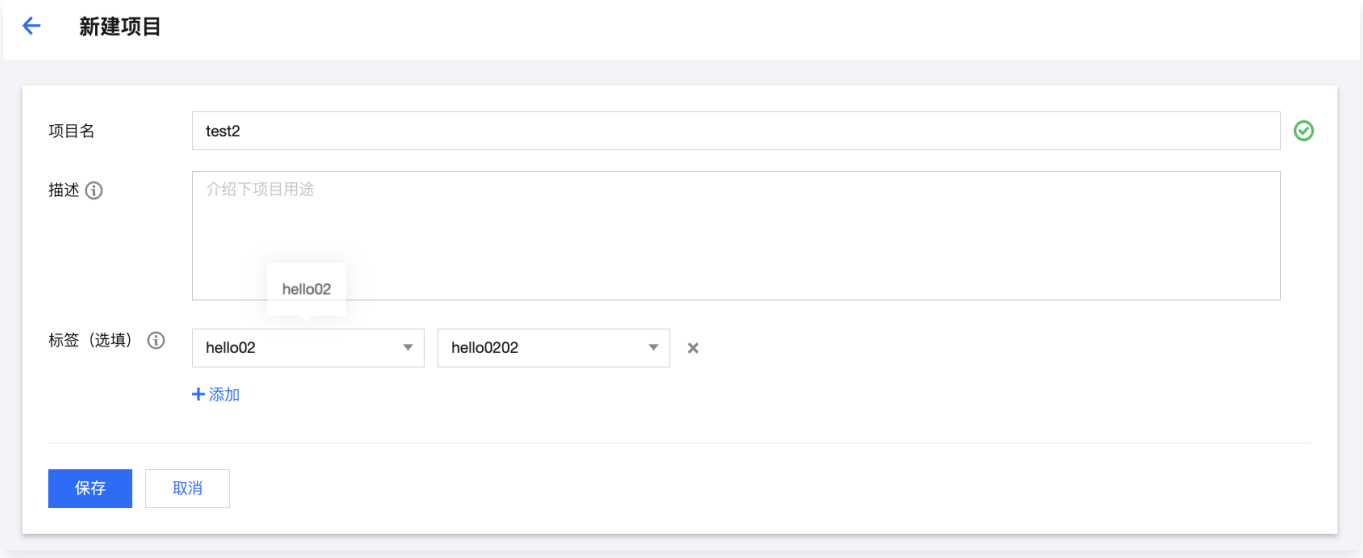
本文介绍如何使用 PTS 脚本模式，通过 JavaScript 脚本快速发起压测，帮您掌握 PTS 的基本用法。

压测场景	描述
脚本模式	使用我们的 JS 代码示例作为脚本的基础，或从头开始。支持 HTTP、Websocket、gRPC 等协议。
简单模式	使用我们的交互式 UI ，组合不同的用户请求。
JMeter	通过使用原生的 JMeter JMX 文件进行压测。
脚本导入	HAR 等 API 导入自动生成测试场景。
流量录制	录制浏览器流量，自动生成测试场景。

操作步骤

步骤1: 创建 PTS 项目

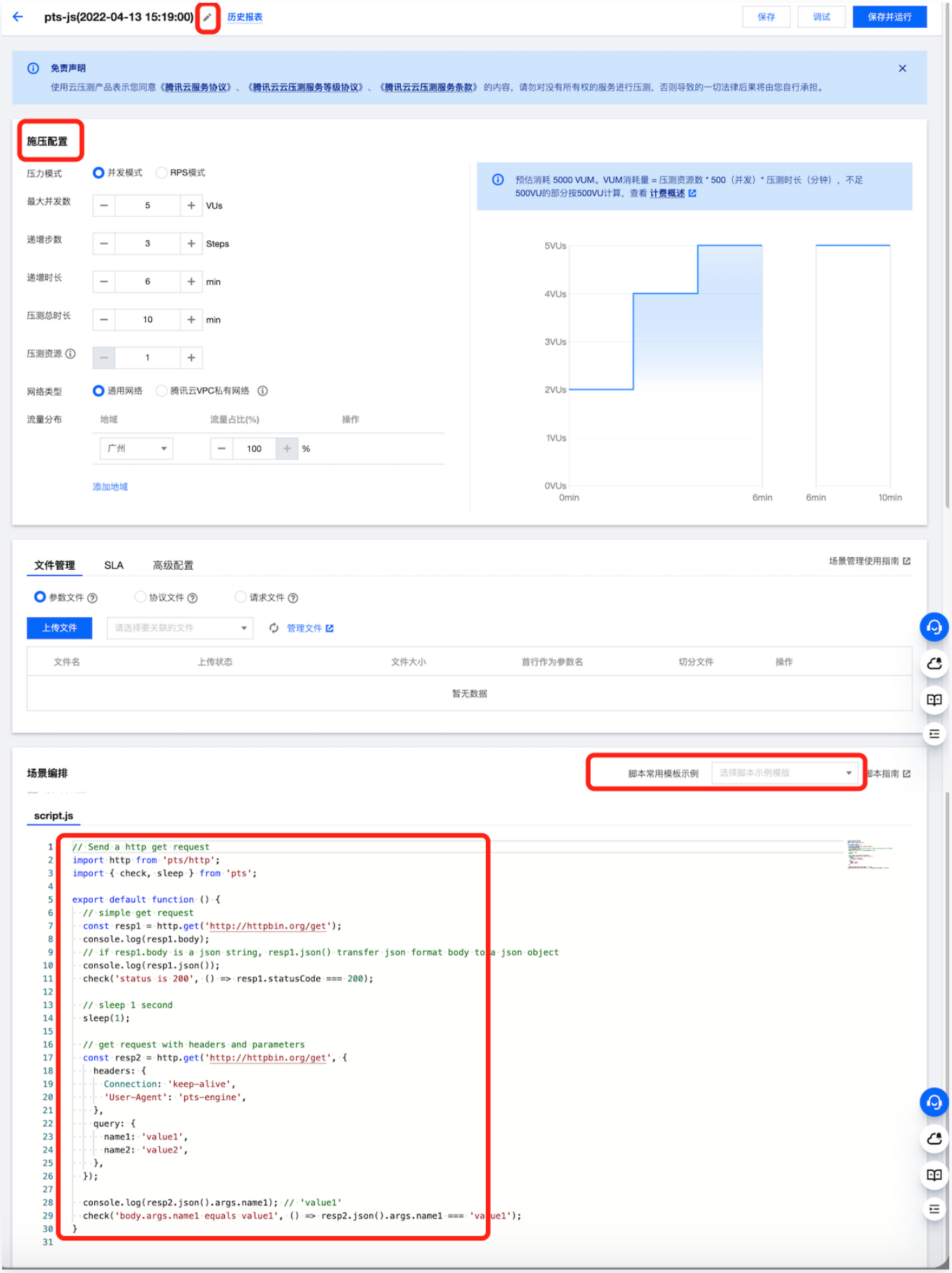
1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中单击**云压测** > **项目列表.**
3. 在项目列表页单击**新建项目** 。
4. 填写**项目名**、**描述和标签**。完成后单击**保存**。



步骤2：创建测试场景

1. 进入创建测试场景页，选择**脚本模式**。

2. 编排测试场景，完成以下设置，然后单击**保存并运行**。新创建的测试场景默认可直接运行。



功能模块说明如下表所示：

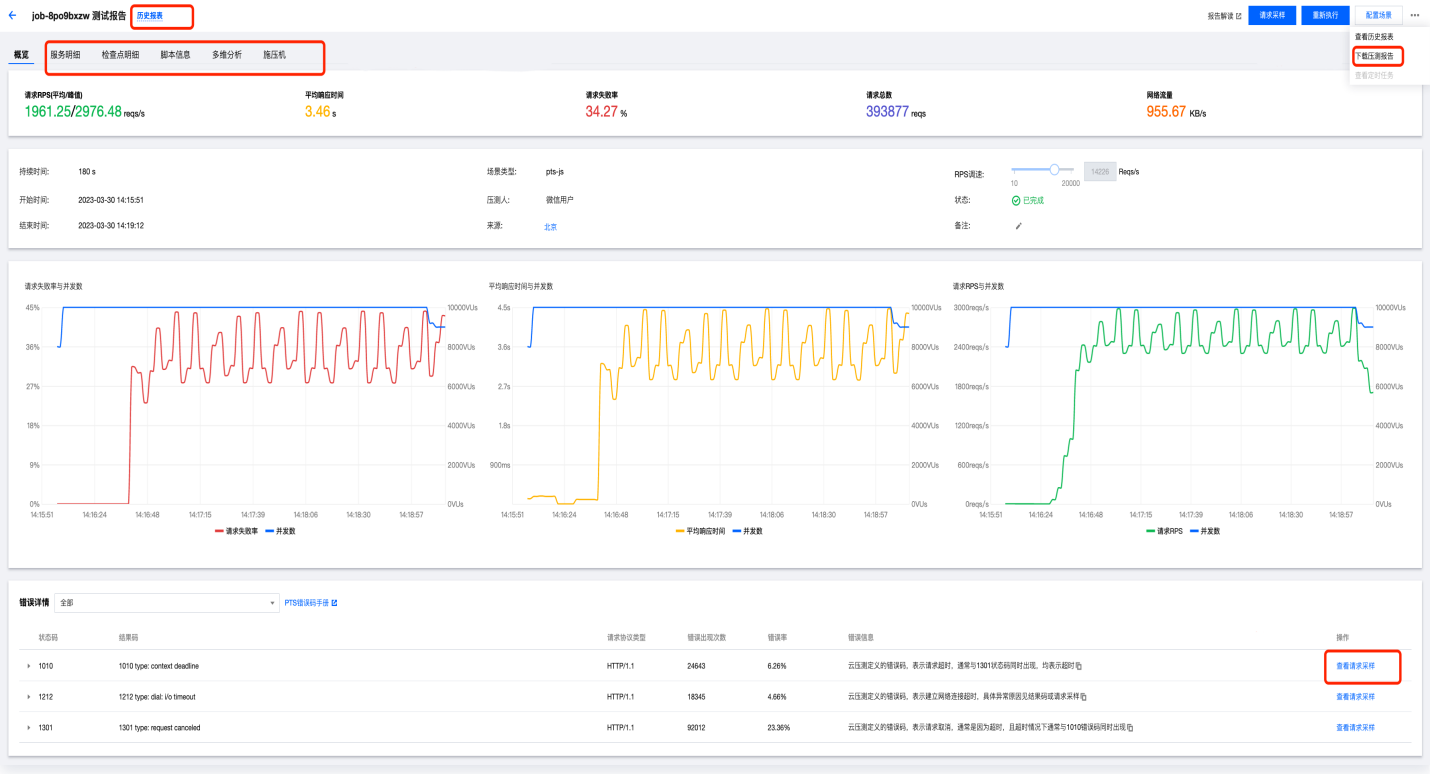
功能模块	描述
场景名字	更新场景名字，方便后续识别场景用途。

施压配置	当前支持压力模型为：并发压力模式(虚拟用户模式)、RPS 压力模式。
	• 并发压力模式(虚拟用户模式)：并发指虚拟并发用户数。从业务角度，也可以理解为同时在线的用户数。
	• RPS 压力模式：每秒请求数量，用于衡量服务端的吞吐量。免去并发数到 RPS 的繁琐转换，来帮忙用户更好的摸底业务性能瓶颈。
	并发模式配置内容
	• 最大并发数：可以理解为被压服务同时在线的用户数。
	• 递增步数：并发压力递增的阶段数量。
	• 递增时长：压力递增持续时间。
	• 压测总时长：压测持续时间。
	• 压测资源：
	  ○ 单压测资源默认提供 500并发，以及对应的底层资源；
	  ○ 若 CPU、内存或出入带宽达到上限，可适当增加压测资源，提高分配给任务的 CPU、内存和出入带宽；
	  ○ 压测资源数增加会带来计费的增加，计费并发 = 压测资源数 * 500。
	• 网络类型：通用网络支持公网访问；腾讯云 VPC 私有网络支持私网访问。
	• 流量分布：选择不同地域施压流量占比。
	RPS 模式配置内容
	• 最大 RPS：压测 RPS 上限，用来摸底业务系统的目标吞吐量。PTS 会根据最大 RPS 为压测任务分配合理的施压资源。
	• 起始 RPS：压测起始 RPS，用户可以在压测过程中手工调整发压 RPS，并观察报表指标变化。
	• 压测总时长：一次压测的总时长。
	• 压测资源：PTS 会根据用户设置的最大 RPS，合理分配压测资源池。如果您的请求响应较慢，您可以通过适当扩大压测资源池，来确保达到目标吞吐量。
	• 流量分布：将压测总流量以一定的百分比，分布于多个地域，以模拟真实场景中，来自不同地域的用户带来的流量。

场景编排	• 提供脚本常用模板示例。 • 支持语法高亮、智能提示、函数引用、代码格式化等。 • JS 编排区域内右键查看更多功能。更多常用的 JS 语法，代码示例请参见 脚本模式压测 。

步骤3：查看压测报告

单击**保存并运行**后，PTS 将拉起压测引擎进行测试。控制台将跳转至施压报告界面。
压测报告生成后，您可以单击导航栏测试场景名称，可查看、下载历史报表。在错误详情列表，您还可单击**查看请求采样**，查看错误请求采样信息。


job-8po9bxzw 测试报告
历史报表
报告解读
请求采样
重新执行
配置场景
查看历史报表
下载压测报告
查看定时任务
概览
服务明细
检查点明细
脚本信息
多维分析
施压机
请求RPS(平均/峰值)
1961.25/2976.48 reqs/s
平均响应时间
3.46 s
请求失败率
34.27 %
请求总数
393877 reqs
网络流量
955.67 KB/s
持续时间:
180 s
开始时间:
2023-03-30 14:15:51
结束时间:
2023-03-30 14:19:12
场景类型:
pts-js
压测人:
微信用户
来源:
北京
RPS调速:
14226
Reqs/s
状态:
已完成
备注:
请求失败率与并发数
平均响应时间与并发数
请求RPS与并发数
请求失败率
并发数
平均响应时间
请求RPS
错误详情
全部
PTS错误码手册
状态码
结果码
请求协议类型
错误出现次数
错误率
错误信息
操作
1010
1010 type: context deadline
HTTP/1.1
24643
6.26%
云压测定义的错误码，表示请求超时，通常与1301状态码同时出现，均表示超时
查看请求采样
1212
1212 type: dial: i/o timeout
HTTP/1.1
18345
4.66%
云压测定义的错误码，表示建立网络连接超时，具体异常原因见结果码或请求采样
查看请求采样
1301
1301 type: request canceled
HTTP/1.1
92012
23.36%
云压测定义的错误码，表示请求取消，通常是因为超时，且超时情况下通常与1010错误码同时出现
查看请求采样

ElasticSearch 监控指南

最近更新时间：2024-05-17 10:47:11

该文档详细介绍了 ElasticSearch 如搜索性能指标、索引性能指标、内存使用和垃圾回收指标等六类监控关键指标、集群和索引两类大盘配置示例，以及 ES 在查询性能差、索引性能差的两种典型问题场景下详细的原因、排查方式和解决方案，同时也介绍了如何通过 Prometheus 监控搭建可靠的监控系统，详尽全面。

ElasticSearch 简介

概要

ElasticSearch（简称ES）是一个分布式、免费、开放的搜索和分析引擎，适用于所有类型的数据，包括文本、数字、地理空间、结构化和非结构化数据，最初由 Elastic 开发并开源。它本质上是一个分布式数据库，底层基于 Lucene 实现。ElasticSearch 屏蔽了 Lucene 的底层细节，提供了分布式特性，同时对外提供了 Restful API。ElasticSearch 以其易用性迅速赢得了许多用户，被用在网站搜索、日志分析等诸多方面。由于 ES 强大的横向扩展能力，甚至很多人也会直接把 ES 当做 NoSQL 来用。

特点

- **实时性**：ElasticSearch 能够实时地存储、检索和分析数据，使得用户能够快速获得最新的搜索结果和分析数据；
- **分布式**：ElasticSearch 采用分布式架构，能够水平扩展，处理 PB 级结构化或非结构化数据，同时具有高可用性和容错性；
- **多样化**的搜索和分析功能：ElasticSearch 支持全文搜索、结构化查询、过滤、地理空间查询和复杂的分析功能；
- **可扩展性**：ElasticSearch 提供了丰富的插件和 API，可以轻松地扩展其功能。

基本概念

节点（Node）

ElasticSearch 是以集群的方式运行的，而节点是组成 ES 集群的基本单位，所以每个 ElasticSearch 实例就是一个节点，每个物理机器上可以有多个节点，使用不同的端口和节点名称。
节点按主要功能可以分为三种：主节点(Master Node)、协调节点(Coordinating Node)和数据节点(Data Node)。

- **主节点**：处理创建，删除索引等请求，维护集群状态信息。可以设置一个节点不承担主节点角色；
- **协调节点**：负责处理请求。默认情况下，每个节点都可以是协调节点；
- **数据节点**：用来保存数据。可以设置一个节点不承担数据节点角色。

集群（Cluster）

ElasticSearch 是一个分布式的搜索引擎，所以一般由多台物理机组成。而在这些机器上通过配置一个相同的 cluster name，可以让其互相发现从而把自己组织成一个集群。

分片 & 副本（Shards & Replicas）

索引可以存储大量的数据，可能会超过单个节点的硬件限制，而且会导致单个节点效率问题。ES 提供了将单个 Index 拆分到多个 Shard 上的能力，可以支持水平扩展，分布式和并行跨 Shard 操作（可能在多个节点），从而提高了性能和吞吐量。
为了避免故障导致节点及分片出现问题，ES 可以为分片设置副本（Replicas），副本通常在不同的节点上，从而保证高可用性。

Index (索引)

Index(索引) 是具有稍微类似特征文档的集合，同在一个索引中的文档共同建立倒排索引。类似于 MySQL 中的 database 概念，但 ES 中的 Index 更加灵活，用起来也更加方便。提交给同一个索引中的文档，最好拥有相同的结构。这样对于 ES 来说，不管是存储还是查询，都更容易优化。

类型（Type）

Document 的类型，类似于关系型数据库中的表的概念。该概念在6.X 时还可以使用，但在 Type 的概念已在7.X 开始废弃，官方认为这是个错误的设计。

Document (文档)

文档是 ES 索引的基本单位，每个索引都是由数量众多的文档组成，Document 相当于传统数据库中的行，ES 中数据以 JSON 的形式来表示。

字段（Fields）

每个 Document 都类似一个 JSON 结构，它包含了许多字段，每个字段都有其对应的值，多个字段组成了一个 Document，可以类比关系型数据库数据表中的字段。

映射（mapping）

相当于数据库中的 schema，用来约束字段的数据类型，每一种数据类型都有对应的使用场景。mapping 中定义了一个文档所包含的所有 field 信息，每个文档都有映射。mapping 不是必须创建，因为 ES 中实现了动态映射。

示例

添加 Index members 的 mapping：

```
PUT members
{
  "mappings": {
   "properties": {
     "id": {
       "type": "long"
   },
  "name": {
    "type": "text"
  },
  "birthday": {
```

```
      "type": "date"
      }
     }
  }
}
```

在 Index members 中添加一个 Document：

```
PUT members/_doc/10086
{
    "id": 10086,
    "name": "法外狂徒张三",
    "birthday": "1990-10-24T09:00:00Z"

}
```

主要适用场景

- **搜索引擎**：在网站、应用或文档存储中提供全文搜索功能。例如，一个电子商务网站使用 ElasticSearch 来让用户快速搜索产品；
- **日志和指标分析**：用于收集、存储和分析日志以及指标数据。例如，一个网络应用程序使用 ElasticSearch 来存储和分析其日志文件，以便监控性能和排查问题；
- **实时数据分析**：用于实时分析和可视化大规模的实时数据。例如，一个金融机构使用 ElasticSearch 来监控交易数据和实时市场动态；
- **内容推荐**：用于根据用户偏好和行为提供个性化的内容推荐。例如，一个新闻网站使用 ElasticSearch 来推荐相关新闻文章给用户；
- **业务指标监控**：用于跟踪和监控业务指标以支持决策。例如，一个企业使用 ElasticSearch 来监控销售数据和库存情况。

监控关键指标

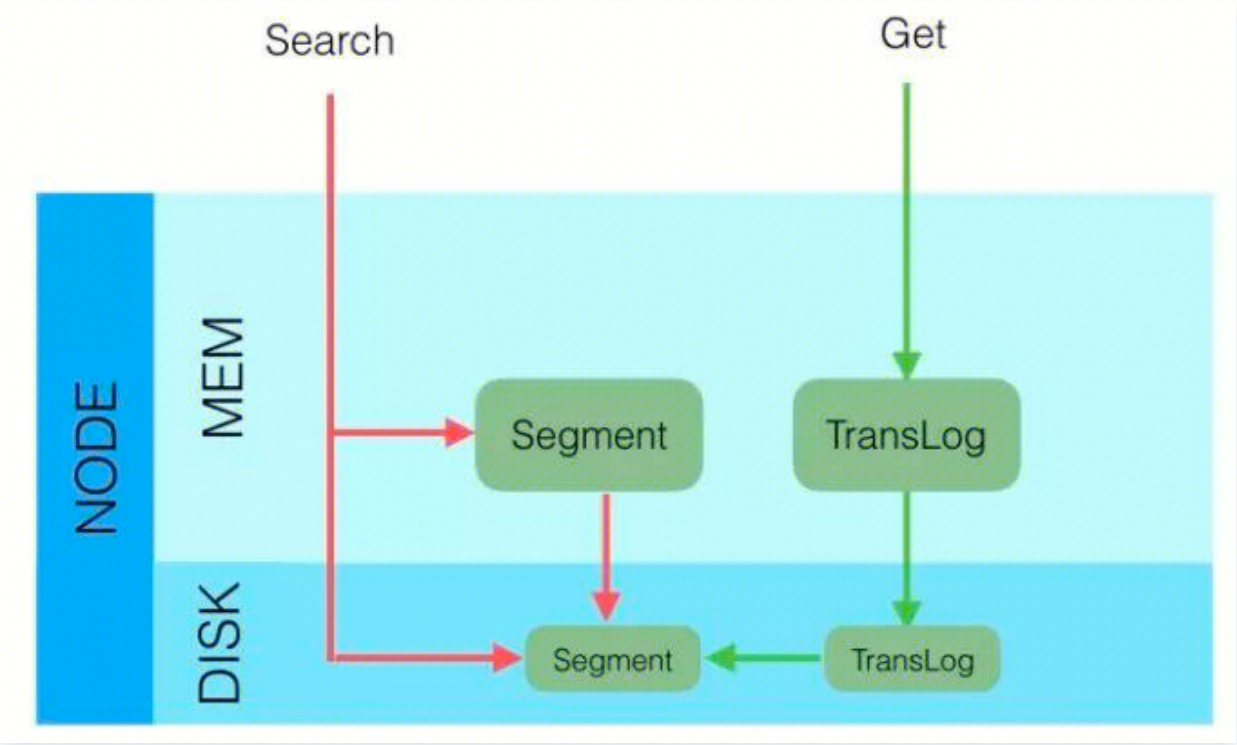
ElasticSearch 提供了大量的指标，可以用于监控各类故障现象：

- 搜索性能指标
- 索引性能指标
- 内存使用和垃圾回收指标
- 主机级别网络系统指标
- 集群健康和节点可用性指标
- 资源饱和度和错误

搜索性能指标

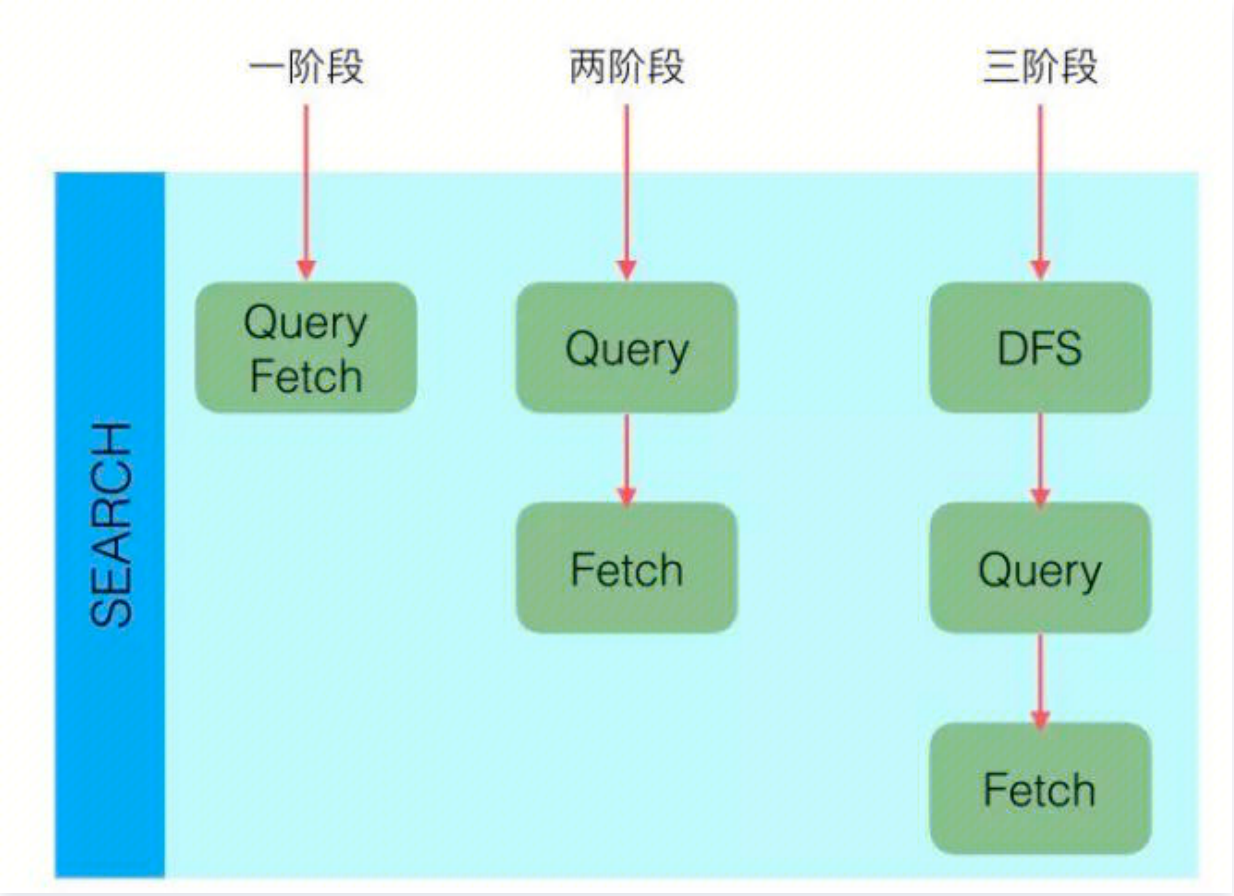
ES 查询可以分为两种：

- 根据 ID 查询 Doc，可实时检索写入的数据。检索流程：检索内存中 Translog → 检索磁盘 Translog → 检索磁盘 Segment；
- 根据 query 查询 Doc，近实时检索写入的数据。检索流程：检索 filesystem cache中 Segment → 检索磁盘 Segment。



检索策略可分为三大类：

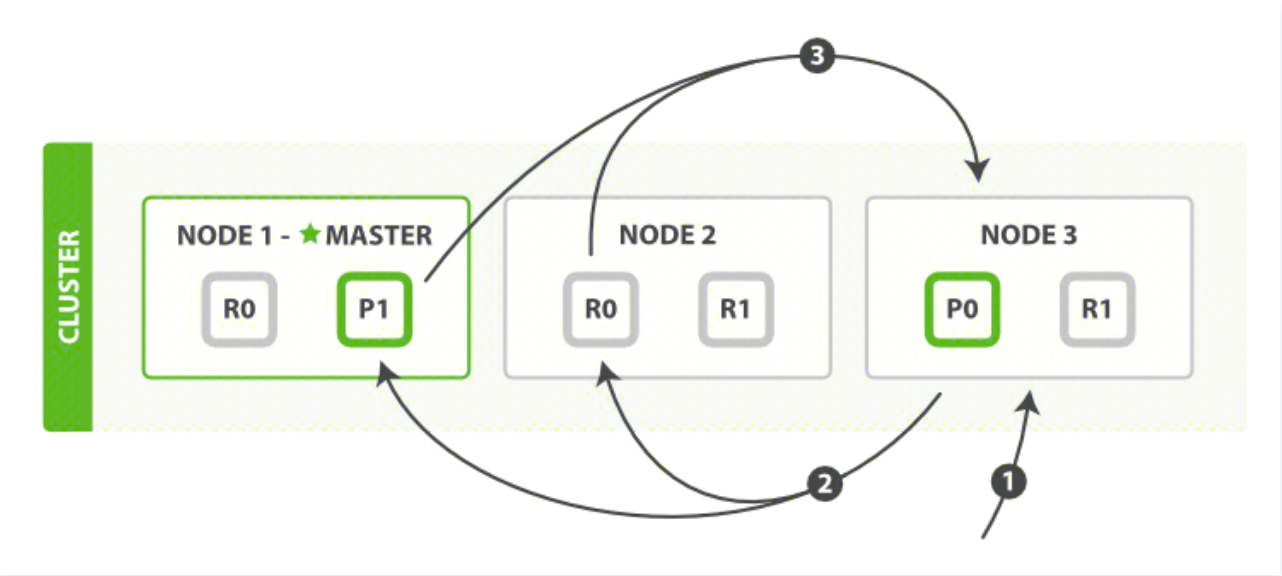
- QUERY_AND_FETCH：查询完就返回整个 Doc 内容，对应根据 ID 查询 Doc；
- QUERY_THEN_FETCH：先查询出对应的 Doc id ，然后再根据 Doc id 匹配去对应的文档；
- DFS_QUERY_THEN_FETCH：在 QUERY_THEN_FETCH 的基础上多了算分环节。



分布式场景下，查询通常分成两个阶段，以 QUERY_THEN_FETCH 为例子：

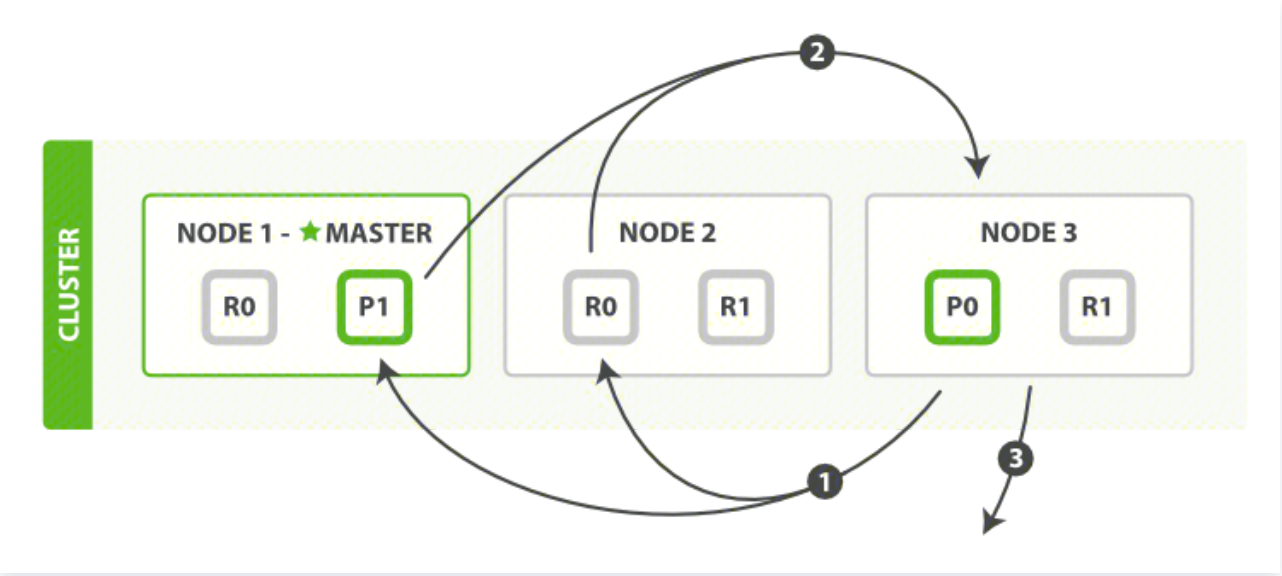
Query 阶段：

1. 客户端发送一个 search 请求到 NODE 3，NODE 3 会创建一个大小为 from + size 的空优先队列；
2. Node 3 将查询请求转发到索引的每个主分片或副本分片中。每个分片在本地执行查询并添加结果到大小为 from + size 的本地有序优先队列中；
3. 每个分片返回各自优先队列中所有文档的 ID 和排序值给协调节点，也就是 NODE 3，它合并这些值到自己的优先队列中来产生一个全局排序后的结果列表。



Fetch 阶段：

1. 协调节点辨别出哪些文档需要被取回并向相关的分片提交多个 GET 请求；
2. 每个分片加载并丰富文档，如果有需要的话，接着返回文档给协调节点；
3. 一旦所有的文档都被取回了，协调节点返回结果给客户端。



相关指标

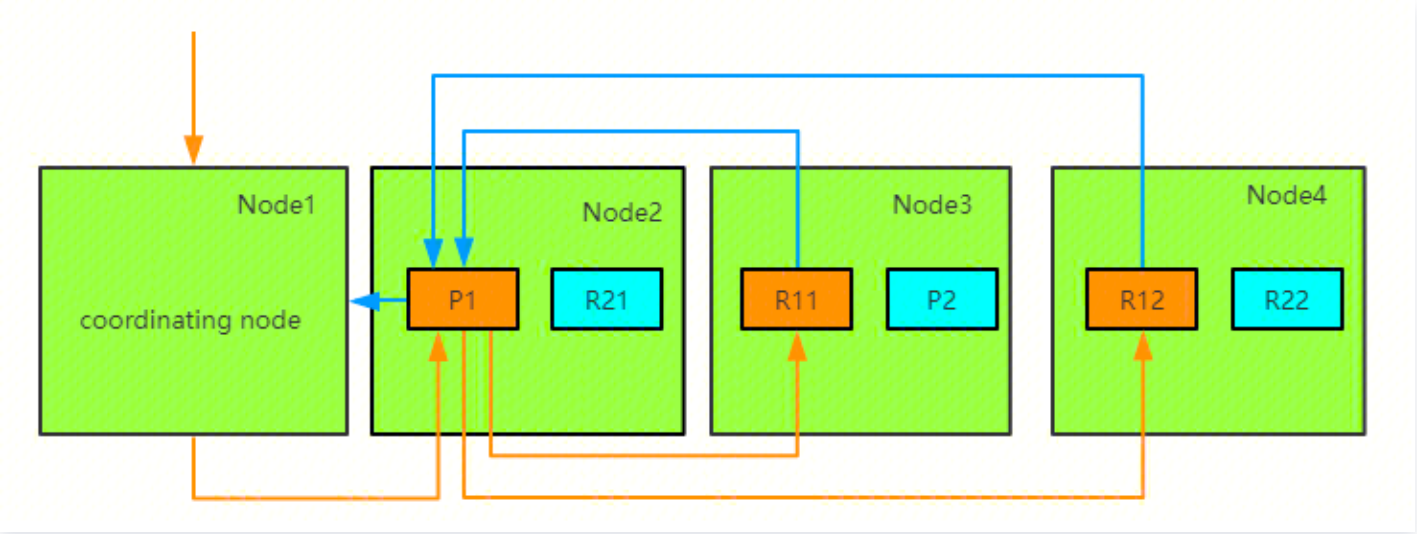
序号	指标	类型	详情
1	elasticsearch_indices_search_query_total	counter	查询总数吞吐量
2	elasticsearch_indices_search_query_time_seconds	counter	查询总时间性能

3	elasticsearch_indices_search_fetch_total	counter	提取总数
4	elasticsearch_indices_search_fetch_time_seconds	counter	花费在提取上的总时间
5	elasticsearch_indices_get_time_seconds	counter	GET 请求总时间
6	elasticsearch_indices_get_missing_total	counter	丢失的文件的 GET 请求总数
7	elasticsearch_indices_get_missing_time_seconds	counter	花费在文档丢失的 GET 请求上的总时间
8	elasticsearch_indices_get_exists_time_seconds	counter	花费在文档存在的 GET 请求上的总时间
9	elasticsearch_indices_get_exists_total	counter	存在的文件的 GET 请求总数
10	elasticsearch_indices_get_total	counter	GET 请求总次数

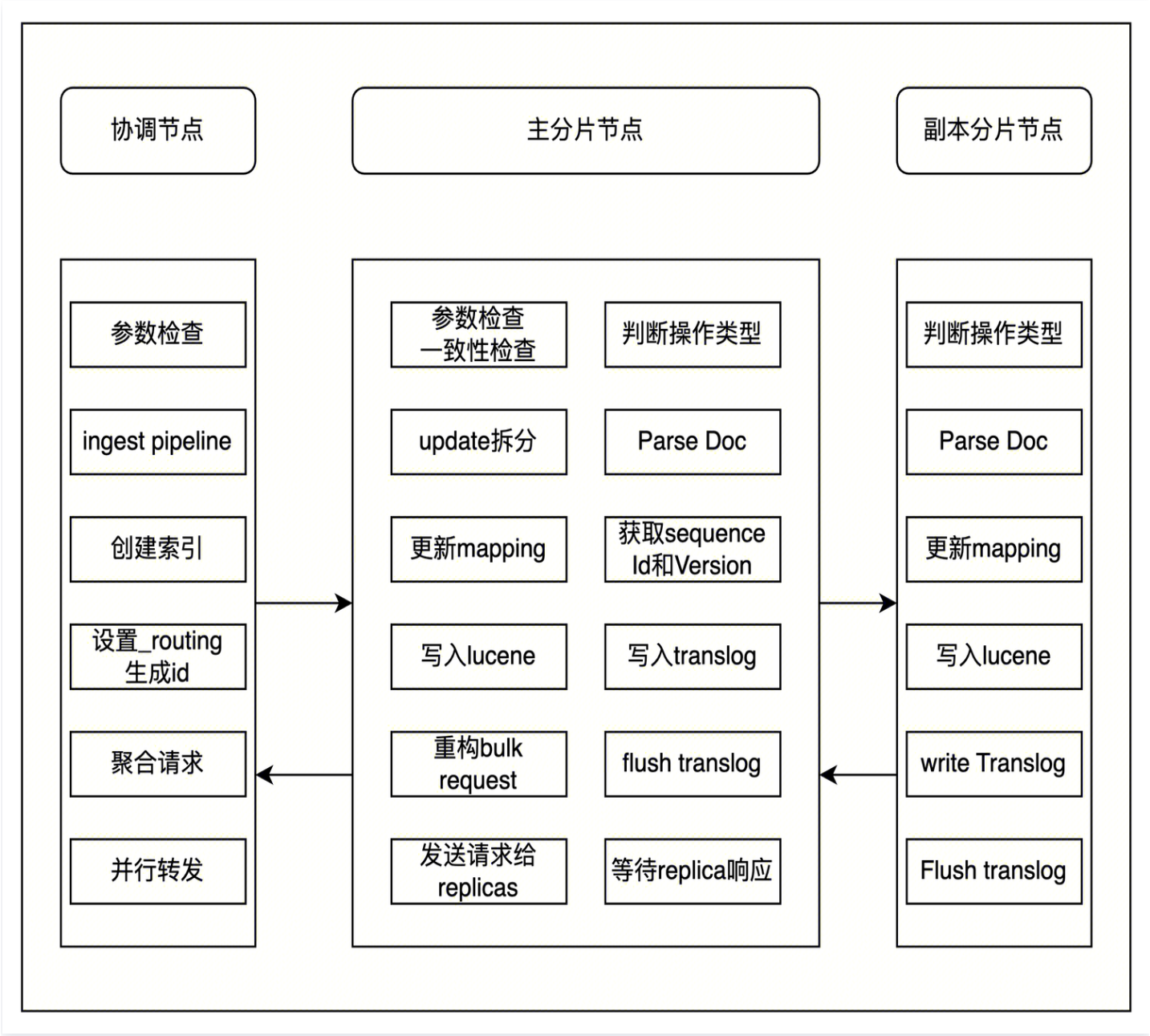
索引性能指标

ES 写入流程介绍

ES 的任意节点都可以作为协调节点(coordinating node)接受请求，当协调节点接受到请求后进行一系列处理，然后通过_routing 字段找到对应的 primary shard，并将请求转发给 primary shard，primary shard 完成写入后，将写入并发发送给各 replica，raplica 执行写入操作后返回给 primary shard， primary shard 再将请求返回给协调节点。大致流程如下图：



整体写入流程如下图所示：



说明

- **近实时性-refresh 操作**：当一个文档写入 Lucene 后是不能被立即查询到的，ElasticSearch 提供了一个 refresh 操作，会定时调用 Lucene 的 reopen (新版本为 openIfChanged)为内存中新写入的数据生成一个新的 Segment，此时被处理的文档均可以被检索到。refresh 操作的时间间隔由 refresh_interval 参数控制，默认为1s，当然还可以在写入请求中带上 refresh 表示写入后立即 refresh，另外还可以调用 refresh API 显式 refresh。
- **merge 操作**：由于 refresh 默认间隔为1s，因此会产生大量的小 Segment，为此 ES 会运行一个任务检测当前磁盘中的 Segment，对符合条件的 Segment 进行合并操作，减少 Lucene 中的 Segment 个数，提高查询速度，

降低负载。不仅如此，merge 过程也是文档删除和更新操作后，旧的 Doc 真正被删除的时候。用户还可以手动调用 _forcemerge API 来主动触发 merge，以减少集群的 Segment 个数和清理已删除或更新的文档。

- **Translog**：当一个文档写入 Lucence 后是存储在内存中的，即使执行了 refresh 操作仍然是在文件系统缓存中，如果此时服务器宕机，那么这部分数据将会丢失。为此 ES 增加了 Translog。

  当进行文档写操作时会先将文档写入 Lucene，然后写入一份到 Translog，写入 Translog 是落盘的(如果对可靠性要求不是很高，也可以设置异步落盘，可以提高性能，由配置 `index.Translog.durability` 和 `index.Translog.sync_interval` 控制)，这样就可以防止服务器宕机后数据的丢失。由于 Translog 是追加写入，因此性能比较好。与传统的分布式系统不同，这里是先写入 Lucene 再写入 Translog，原因是写入 Lucene 可能会失败，为了减少写入失败回滚的复杂度，因此先写入 Lucene。
- **flush 操作**：每30分钟或当Translog 达到一定大小(由index.Translog.flush_threshold_size 控制，默认512mb), ES 会触发一次 flush 操作，此时 ES 会先执行 refresh 操作将 buffer 中的数据生成 Segment，然后调用 Lucene 的 commit 方法将所有内存中的 Segment fsync 到磁盘。此时 Lucene 中的数据就完成了持久化，会清空 Translog 中的数据(6.x版本为了实现 sequenceIDs，不删除 Translog)。

相关指标

根据上述流程可知，对于一个写入较为频繁的系统，refresh 和 flush 操作相关的指标较为重要，merge 相关的指标也需要关注，同时整个写入的耗时、搜索文档总数也需要关注，故而需要关注的写入相关指标为：

序号	指标	类型	详情
1	elasticsearch_indices_indexing_index_total	counter	被索引的文档总数
2	elasticsearch_indices_indexing_index_time_seconds_total	counter	索引文档花费的总时间
3	elasticsearch_indices_refresh_total	counter	索引 refresh 的总数
4	elasticsearch_indices_refresh_time_seconds_total	counter	refresh 索引总共话费的时间
5	elasticsearch_indices_flush_total	counter	flush 索引到磁盘的总数
6	elasticsearch_indices_flush_time_seconds	counter	flush 索引到磁盘的总花费时间
7	elasticsearch_indices_merges_total	counter	Total merges（merge 操作总次数）
8	elasticsearch_indices_merges_total_time_seconds_total	counter	索引 merge 操作总花费时间

9	elasticsearch_indices_merges_Docs_total	counter	merge 的文档总数
10	elasticsearch_indices_merges_total_size_bytes_total	counter	merge 的数据量合计
11	elasticsearch_indices_indexing_delete_total	counter	索引的文件删除总数
12	elasticsearch_indices_indexing_delete_time_seconds_total	counter	索引的文件删除总时间

内存使用和垃圾回收指标

ES 是使用 Java 进行编写，运行在 JVM 之上，故而 ES 会以两种方式使用节点上的全部可用内存：JVM 堆和文件系统缓存(file system cache)，因此，在 ES 运行期间，整个 JVM 的垃圾回收持续时间和频率将很值得监控。

相关指标

序号	指标	类型	详情
1	elasticsearch_jvm_gc_collection_seconds_sum	counter	JVM GC 垃圾回收时间
2	elasticsearch_jvm_gc_collection_seconds_count	counter	JVM GC 垃圾搜集数
3	elasticsearch_jvm_memory_committed_bytes	gauge	JVM 最大使用内存限制
4	elasticsearch_jvm_memory_max_bytes	gauge	配置的最大 JVM 值
5	elasticsearch_jvm_memory_pool_max_bytes	counter	JVM 内存最大池数
6	elasticsearch_jvm_memory_pool_peak_max_bytes	counter	最大的 JVM 内存峰值
7	elasticsearch_jvm_memory_pool_peak_used_bytes	counter	池使用的 JVM 内存峰值
8	elasticsearch_jvm_memory_pool_used_bytes	gauge	目前使用的 JVM 内存池
9	elasticsearch_jvm_memory_used_bytes	gauge	JVM 内存使用量

主机级别网络系统指标

主机级别资源及网络使用情况也需要关注。

版权所有：腾讯云计算（北京）有限责任公司

相关指标

序号	指标	类型	详情
1	elasticsearch_process_cpu_percent	gauge	CPU 使用率
2	elasticsearch_filesystem_data_free_bytes	gauge	磁盘可用空间
3	elasticsearch_process_open_files_count	gauge	ES 进程打开的文件描述符
4	elasticsearch_transport_rx_packets_total	counter	ES 节点之间网络入流量
5	elasticsearch_transport_tx_packets_total	counter	ES 节点之间网络出流量

集群健康和节点可用性指标

ES 是按集群部署的，需要关注集群相关指标。

相关指标

序号	指标	类型	详情
1	elasticsearch_cluster_health_status	gauge	集群状态，green（ 所有的主分片和副本分片都正常运行）、yellow（所有的主分片都正常运行，但不是所有的副本分片都正常运行）red（有主分片没能正常运行），值为1的即为对应状态。
2	elasticsearch_cluster_health_number_of_data_nodes	gauge	node 节点的数量
3	elasticsearch_cluster_health_number_of_in_flight_fetch	gauge	正在进行的碎片信息请求的数量
4	elasticsearch_cluster_health_number_of_nodes	gauge	集群内所有的节点
5	elasticsearch_cluster_health_number_of_pending_tasks	gauge	尚未执行的集群级别更改
6	elasticsearch_cluster_health_initializing_shards	gauge	正在初始化的分片数
7	elasticsearch_cluster_health_unassigned_shards	gauge	未分配分片数

版权所有：腾讯云计算（北京）有限责任公司

8	elasticsearch_cluster_health_active_primary_shards	gauge	活跃的主分片总数
9	elasticsearch_cluster_health_active_shards	gauge	活跃的分片总数（包括复制分片）
10	elasticsearch_cluster_health_relocating_shards	gauge	当前节点正在迁移到其他节点的分片数量，通常为0，集群中有节点新加入或者退出时该值会增加。
11	elasticsearch_breakers_tripped	counter	熔断发生次数
12	elasticsearch_breakers_limit_size_bytes	gauge	熔断内存限制大小
13	elasticsearch_breakers_estimated_size_bytes	gauge	熔断器内存大小

资源饱和度和错误

ElasticSearch 节点使用线程池来管理线程如何消耗内存和 CPU，所以需要监控线程相关的指标。

相关指标

序号	指标	类型	详情
1	elasticsearch_thread_pool_completed_count	gauge	线程池操作完成线程数
2	elasticsearch_thread_pool_active_count	gauge	线程池活跃线程数
3	elasticsearch_thread_pool_largest_count	gauge	线程池最大线程数
4	elasticsearch_thread_pool_queue_count	gauge	线程池中的排队线程数
5	elasticsearch_thread_pool_active_count	gauge	线程池的被拒绝线程数
6	elasticsearch_indices_fielddata_memory_size_bytes	gauge	fielddata 缓存的大小（字节）
7	elasticsearch_indices_fielddata_evictions	gauge	来自 fielddata 缓存的驱逐次数

监控大盘

Prometheus 监控服务提供了开箱即用的 Grafana 监控大盘，根据预设大盘可以直接监控 ES 集群各类重要指标状态，能够进行问题的快速定位。本文集群大盘使用了腾讯云 ES 作为数据源，腾讯云 Elasticsearch Service（ES）是云端全托管海量数据检索分析服务，拥有高性能自研内核，集成X-Pack。ES 支持通过自治索引、存算分离、集群巡检等特性轻松管理集群，也支持免运维、自动弹性、按需使用的 Serverless 模式。使用 ES 您可以高效构建信息检索、日志分析、运维监控等服务，它独特的向量检索还可助您构建基于语义、图像的 AI 深度应用。

集群大盘

集群大盘提供了集群相关的监控信息，包括了集群节点、分片、资源利用率、熔断、线程池等监控信息。

集群概要

该监控提供集群概览信息，可大致概览集群状况，可以看出集群健康状况、节点数、数据节点、熔断器熔断次数、利用率、集群 pending 任务数、ES 进程打开的文件描述符等信息。集群健康监控 red 为1，green 时为5，yellow 为23。

分片监控（Shards）

分片监控提供集群中的主节点数、副本节点数，同时提供正在初始化的、正在迁移的、延迟分配的、未分配的分片信息。

熔断监控（Breakers）

该监控提供各类熔断器熔断的次数，以及熔断内存使用量。可以通过该监控去排查熔断发生的熔断器类别、熔断限制、熔断发生时的内存使用量以及是哪个节点发生的熔断。

- Tripped for breakers：分析各类型的熔断器熔断的次数；
- Estimated size in bytes of breaker：分析给类型熔断器熔断内存使用量。

节点监控

- Load average：分析各节点的短期平均负载；
- CPU usage：分析 CPU 使用率；
- GC count：分析 JVM GC 运行次数；
- GC time：分析 JVM GC 运行时间。

节点监控提供了各节点的短期平均负载、CPU 使用情况、JVM GC 运行相关数据、数据存储使用情况、网络使用情况等。可以通过该监控发现并快速定位节点资源问题。

- JVM memory usage：分析 JVM 内存使用量、内存最大限制以及池内存使用峰值；
- JVM memory committed：分析各区域提交内存使用量；
- Disk usage：分析数据存储使用情况；
- Network usage：分析网络使用情况，包括发送和接收。

线程池监控（Thread Pool）

- Thread Pool operations rejected：分析线程池中各类型操作拒绝率；
- Thread Pool operations queued：分析线程池中各类型线程排队数；
- Thread Pool threads active：分析线程池中各类型活跃线程数；
- Thread Pool operations completed：分析线程池中各类型线程完成数。

索引大盘

Translog

该监控提供 Translog 相关指标

- Total Translog Operations：分析 Translog 操作总数；
- Total Translog size in bytes：查看 Translog 内存使用趋势，分析性能是否影响写入性能。

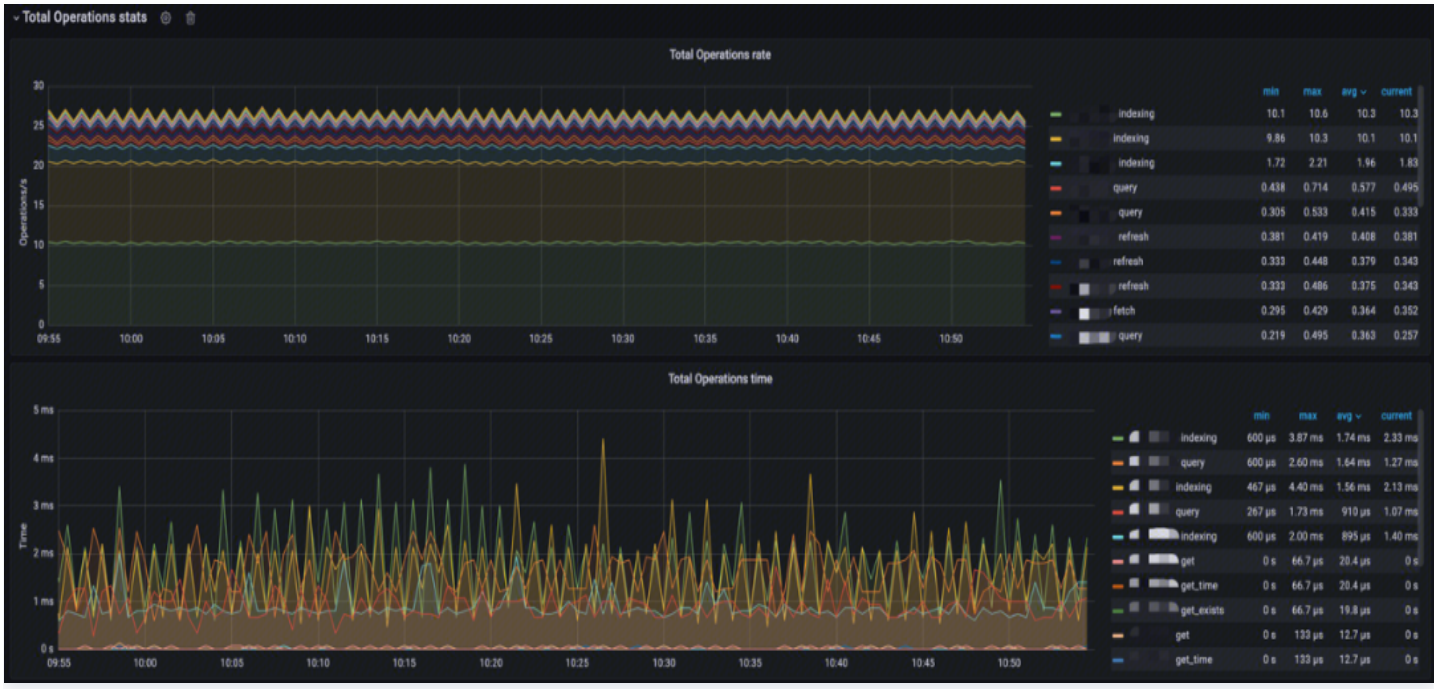
文档（Documents）

- Documents count on node：分析节点 Index 文档数，看索引是否过大；
- Documents indexed rate：分析 Index 索引速率；
- Documents deleted rate：分析 Index 文档删除速率；
- Documents merged rate：分析索引文档 merge 速率；
- Documents merged bytes：分析 Index merge 内存大小。

延时

- Query time：分析 Index 查询耗时；
- Indexing time：分析索引文档耗时；
- Merging time：分析索引 merge 操作耗时；
- Throttle time for index store：Index 写入限制时间，分析 merge 与写入的合理性。

索引操作次数及耗时（Total Operation stats）



- Total Operations rate：分析 Index 索引操作速率、查询速率、查询 fetch 速率，merge 速率、refresh 速率、flush 速率、GET 存在的速率、GET 不存在的速率、GET操作速率、Index 删除速率；
- Total Operations time：分析 Index 索引操作耗时、查询耗时、查询 fetch 耗时，merge 耗时、refresh 耗时、flush 耗时、GET 存在的耗时、GET 不存在的耗时、GET操作耗时、Index 删除耗时。

线程池（Thread Pool）

- Thread Pool operations rejected：分析线程池中各类型操作拒绝率；
- Thread Pool operations queued：分析线程池中各类型线程排队数；
- Thread Pool threads active：分析线程池中各类型活跃线程数；
- Thread Pool operations completed：分析线程池中各类型线程完成数。

缓存（Caches）

- Field data memory size：分析 Fielddata 缓存大小；
- Field data evictions：分析 Fielddata 缓存达到驱逐限制后驱逐的数据的速率；
- Query cache size：分析查询缓存大小；
- Query cache evictions：分析 Query 缓存达到驱逐限制后驱逐的数据的速率；
- Evictions from filter cache：分析过滤查询达到驱逐限制后驱逐的数据的速率。

段（Segments）

- Documents count on node：分析节点上的索引 Segment 计数；
- Current memory size of Segments in bytes：分析当前索引 Segment 内存大小。

典型问题场景

ElasticSearch 查询性能差

ElasticSearch 查询性能变差的原因有很多，需要通过监控指标判断具体症状，然后根据症状进行相应处理。

非活动状态下资源利用率也很高

每个分片都消耗资源（CPU/内存）。即使没有索引/搜索请求，分片的存在也会产生集群开销。

- **原因**：集群中分片太多，以至于任何查询的执行速度看起来都很慢；
- **排查方法**：检查集群大盘中的分片监控，查看分片是否过多；
- **解决方案**：减少分片计数，实施冻结索引和/或添加附加节点来实现负载平衡。考虑结合使用 ElasticSearch 中的热/温架构（非常适合基于时间的索引）以及滚动/收缩功能，以高效管理分片计数。要想顺利完成部署，最好先执行适当的容量计划，以帮助确定适合每个搜索用例的最佳分片数。

线程池存在大量的“rejected”

线程池中存在大量拒绝，从而导致查询不能被正常执行。

- **原因**：查询面向的分片太多，超过了集群中的核心数。这会在搜索线程池中造成排队任务，从而导致搜索拒绝。另一个常见原因是磁盘 I/O 速度慢，导致搜索排队或在某些情况下 CPU 完全饱和；
- **排查方法**：查看集群大盘线程池监控中的拒绝率监控，判断是否有大量拒绝；
- **解决方案**：创建索引时采用1个主分片:1个副本分片 (1P:1R) 模型。使用索引模板是一个在创建索引时部署此设置的好方法。（ElasticSearch 7.0 或更高版本将默认 1P:1R）。ElasticSearch 5.1 或更高版本支持搜索任务取消，这对于取消任务管理 API 中出现的慢查询任务非常有用。

高 CPU 使用率和索引延迟

指标相关性表明，当集群不堪重负时，CPU 使用率和索引延迟都会很高。

- **原因**：集群索引量大会影响搜索性能。
- **排查方法**：查看索引大盘节点监控中的 CPU 使用率、JVM CPU 使用率监控查看 CPU 利用率，然后通过延时告警中的索引耗时面板查看索引延迟状况；
- **解决方案**：提高 refresh 间隔 index.refresh_ interval 至30s，通常有助于提高索引性能。这可以确保分片不必因为每1秒默认创建一个新分段而造成工作负载增大。

副本分片增加后延迟增大

在副本分片计数增加（例如，从1增加到2）后，可以观察到查询延迟。如果存在较多的数据，那么缓存的数据将很快被逐出，从而导致操作系统页面错误增加。

- **原因**：文件系统缓存没有足够的内存来缓存索引中经常查询的部分。ElasticSearch 的查询缓存实现了 LRU 逐出策略：当缓存变满时，将逐出最近使用最少的数据，以便为新数据让路。

- **排查方法**：在分片有增加时，查看索引大盘延时中的查询耗时监控，观察查询延迟是否增大，若有增大，查看索引大盘缓存面板中的查询缓存及查询缓存驱逐监控，缓存变高，驱逐量增大，就是该问题；
- **解决方案**：为文件系统缓存留出至少50%的物理 RAM。内存越多，缓存的空间就越大，尤其是当集群遇到 I/O 问题时。假设堆大小已正确配置，任何剩余的可用于文件系统缓存的物理 RAM 都会大大提高搜索性能。除了文件系统缓存，ElasticSearch 还使用查询缓存和请求缓存来提高搜索速度。所有这些缓存都可以使用搜索请求首选项进行优化，以便每次都将某些搜索请求路由到同一组分片，而不是在不同的可用副本之间进行交替。这将更好地利用请求缓存、节点查询缓存和文件系统缓存。

共享资源时利用率高

操作系统显示出持续的高 CPU/磁盘 I/O 利用率。停止第三方应用程序后，可以看到性能会提高。

- **原因**：其他进程（例如 Logstash）和 ElasticSearch 本身之间存在资源（CPU 和/或磁盘 I/O）争用。
- **排查方法**：查看集群大盘节点监控面板中的 CPU、磁盘、网络等利用率监控，发现持续居高，此时停止第三方应用，就会发现 CPU、磁盘、网络等利用率下降，同时性能提高；
- **解决方案**：避免在共享硬件上与其他资源密集型应用程序一起运行 ElasticSearch。

ElasticSearch 索引性能差

ElasticSearch 索引性能变差的原因同样有很多，具体情况具体分析。

硬件资源不足

硬件资源是一切的基础，其性能决定着运行在其上的集群的性能上限。

- **原因**：硬盘速度慢、CPU 负载高、内存不足等会导致写入性能下降；
- **排查方法**：查看集群大盘节点监控面板中的 CPU、磁盘、网络等利用率监控，各指标持续居高；
- **解决方案**：升级硬件、增加节点或者使用更快的存储设备。

索引设置不当

索引的分片、副本数及刷新时间间隔等均会影响索引性能。

- **原因**：不合理的索引设置，如过多的分片数、不合理的副本数、不适当的刷新间隔等会影响写入性能。分片越多会导致写入变慢，多副本会极大的影响写入吞吐，刷新操作属于代价很高的操作，过于频繁的刷新操作会影响集群整体性能；
- **排查方法**：查看集群大盘中的分片大盘，查看分片数、副本数是否过多，判断其合理性。
- **解决方案**：优化索引设置，调整分片和副本数量，增大刷新间隔等。

索引压力过大

集群的写入能力存在其上限，写入速度不能超过特定限制。

- **原因**：大量的写入请求超过了集群的处理能力，导致写入性能下降；
- **排查方法**：查看索引大盘中的 Total Operations rate 查看各类型索引操作速率，通过 Total Operations time 查看各类型索引操作耗时来综合判断写入能力是否达到上限；
- **解决方案**：通过水平扩展增加节点、优化写入请求分发策略、使用异步写入等方式缓解写入压力。

版权所有：腾讯云计算（北京）有限责任公司

索引数据过大

索引应该有数据上限，超过一定数量就会导致性能的大幅度下降。

- **原因**：索引过大会导致写入性能下降，特别是在硬盘空间不足的情况下；
- **排查方法**：查看索引大盘文档监控查看文档总数、文档索引速率、文档删除速率。若文档删除速率为0，则说明索引生命管理周期有问题，存在生命管理周期的情况下文档数和索引速率还是居高，则为考虑索引其他问题；
- **解决方案**：定期对索引进行优化、使用索引生命周期管理功能、或者将数据分散到多个索引中。

索引数据热点

实际使用中经常会出现，某些特定的业务使用较大，相应的索引负担就比较大了。

- **原因**：部分热点数据集中写入会造成部分节点负载过重，而其他节点负载较轻；
- **排查方法**：查看集群大盘中的节点监控，查看是否部分节点各类负载较高，其余节点较低；
- **解决方案**：重新分片、使用索引别名进行数据迁移、或者调整数据写入策略。

共享资源时利用率高

原因及处理同查询性能变差 。

监控系统搭建

自建 Prometheus 的痛点

1. 自建需要安装并配置 Prometheus、Grafana、AlertManager 、Exporter 等，过程复杂，周期冗长；
2. 配置复杂，需要配置 Exporter 和服务发现等来监控 ElasticSearch 集群；
3. 开源 Grafana 大盘缺少结合 ElasticSearch 原理/特征和最佳实践进行深入优化；
4. 告警配置复杂，且告警系统需要自主搭建，提高复杂度；
5. 需要定期更新和维护 Prometheus 及其相关组件，以确保其正常运行和监控效果。

腾讯云 Prometheus 监控接入

Prometheus 监控服务提供了基于 Exporter 的方式来监控 ElasticSearch 运行状态，具体接入流程可见 ElasticSearch Exporter 接入 。

自建与腾讯云 Prometheus 监控对比

对比类型	自建 Prometheus	腾讯云 Prometheus 监控
成本	需自行配置和部署 Prometheus 及相关组件，采集组件需要自行扩缩容，运维成本高	开箱即用的 Prometheus+Grafana+告警中心一体化平台，全托管，免运维
服务发现	自行配置，维护成本高	图形界面可直接选择需要上报的指标，大大降低维护复杂度
Grafana 大盘	开源大盘信息较少，无法有效排查问题	专业的 ElasticSearch 监控大盘，方便用户快速定位问题及性能优化

告警	需要用户自行部署告警相关内容，且配置较为复杂	提供告警平台，同时提供清晰界面进行告警配置，简化操作

版权所有：腾讯云计算（北京）有限责任公司

Prometheus 监控服务

最近更新时间：2024-12-05 16:40:32

功能介绍

Prometheus 监控服务在继承开源 Prometheus 监控能力的同时，还提供高可用的 Prometheus 托管服务及开源可视化的 Grafana。为您减少用户的开发及运维成本。

> **说明**
> 对于已创建腾讯云 容器服务 TKE 的用户，您可以在 Prometheus 监控服务控制台 创建 Prometheus 实例并安装 Prometheus 监控插件对其进行监控，同时 Prometheus 监控服务集成 Grafana 及预定义 Dashboard 来查看不同维度的性能指标数据。

前提条件

创建腾讯云容器服务 托管版集群 。

操作步骤

步骤1：创建 Prometheus 实例

1. 登录 Prometheus 监控服务控制台 。
2. 单击**新建**，进入新建购买页，可根据自己的实际情况购买对应的实例，详情请参见 创建实例 。

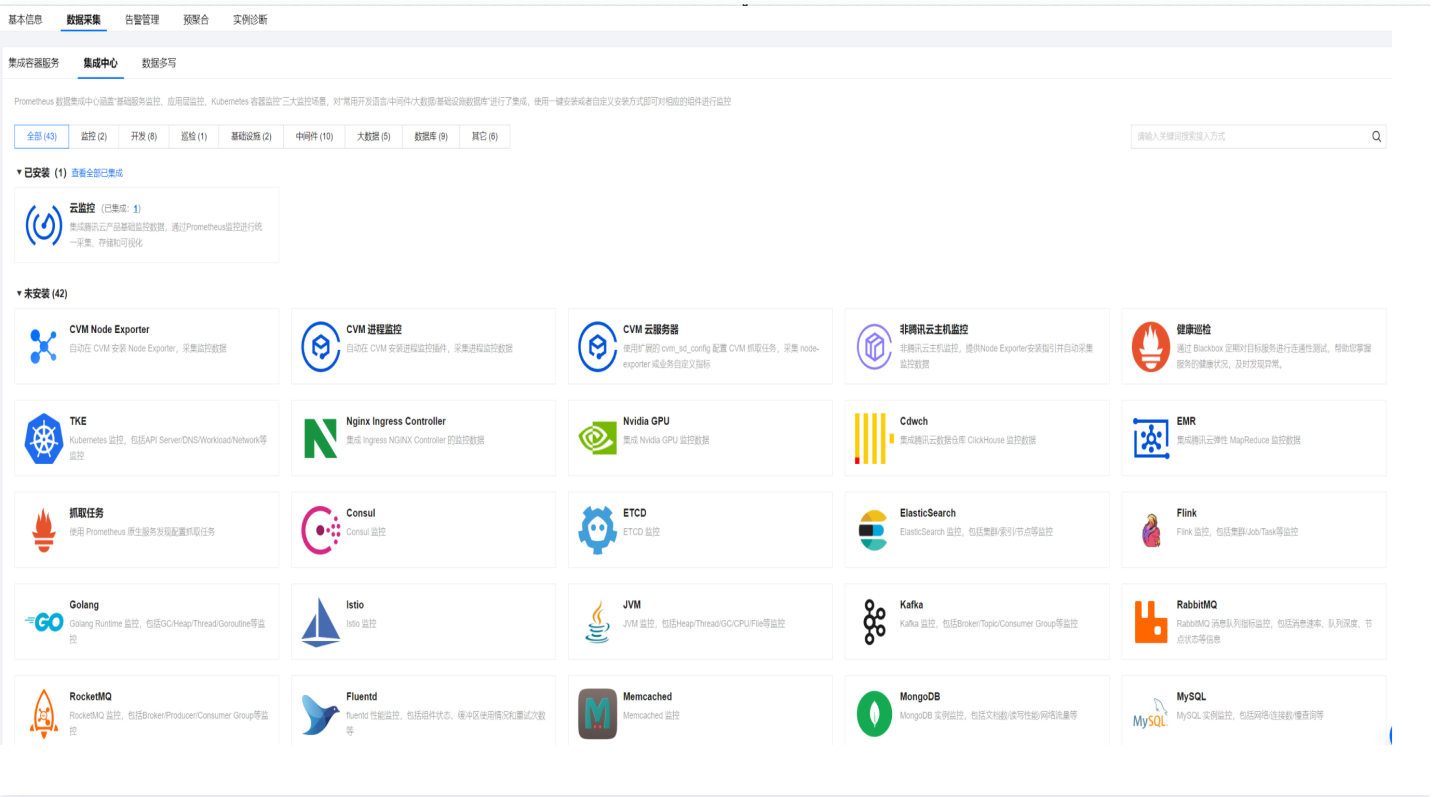
步骤2：集成容器服务

1. 在 Prometheus 实例列表中，单击新建的**实例 ID/名称** 。
2. 进入 Prometheus 管理中心，在顶部导航栏中单击**数据采集**。
3. 在集群容器服务页面进行下列操作。
   - 关联集群：将集群和 Prometheus 实例关联，参考指引 关联集群 。
   - 数据采集配置：支持通过控制台新增或 Yaml 文件配置两种方式，配置新的数据采集规则来监控您的业务数据，参考指引 数据采集配置 。
   - 精简基础监控指标：选择需要上报的指标，避免不必要的费用支出。参考指引 精简基础监控指标 。

> **注意：**
> 配置数据采集，需要先进行初始化操作，创建采集端所需的 EKS 集群。具体收费说明可参考 相关资源使用及计费说明 。

步骤3：接入服务

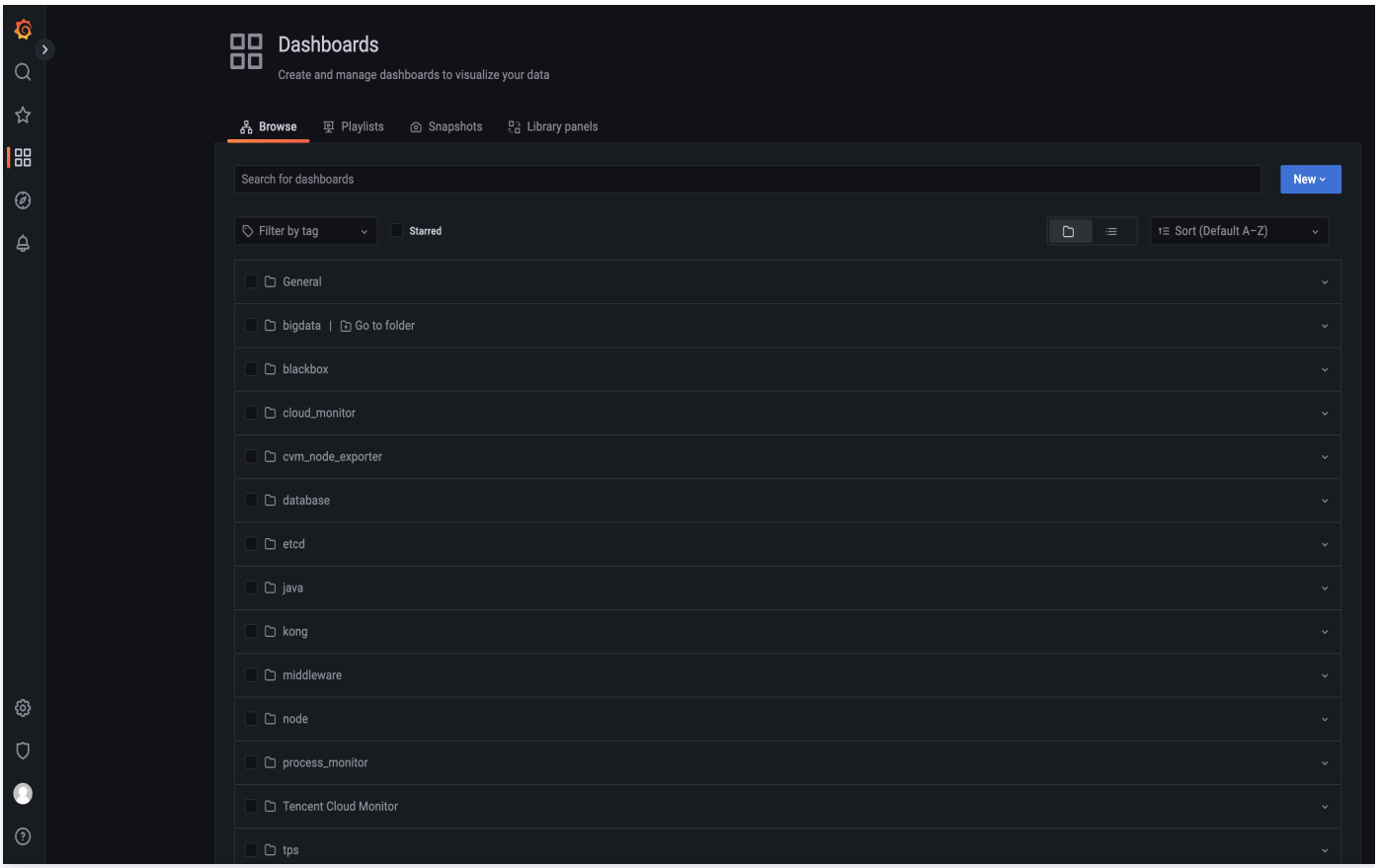
为了方便用户接入，Prometheus 监控服务对常用的 `监控` / `开发` / `中间件` / `大数据` 等组件进行了集成，用户只需根据指引即可对相应的组件进行监控，同时提供了开箱即用的 Grafana 监控大盘。



步骤4：Grafana 查看监控数据

Prometheus 监控服务提供了开箱即用的 Grafana ，同时也集成了丰富的 Kubernetes 基础监控的 Dashboard，以及常用服务监控的 Dashboard，用户可以开箱即用。

1. 在 Prometheus 实例 列表，找到对应的 Prometheus 实例，单击实例 ID 右侧图标，打开您的专属 Grafana，输入账号密码，即可进行 Grafana 可视化大屏操作区。
2. 进入 Grafana，单击图表，展开监控面板。



3. 单击对应的监控图表名称即可查看监控数据。

说明

> 如需了解 Grafana 更多操作说明，请参见 Grafana 官网使用手册 。

Grafana 服务

最近更新时间：2024-07-11 09:41:11

功能介绍

腾讯云可观测平台为您提供高可用 Grafana 托管服务，减少您搭建运维 Grafana 的成本。包括安装、升级、安装插件、安装面板、配置数据源等。

操作步骤

步骤1：创建 Grafana

1. 登录 腾讯云可观测平台 控制台，选择 **Grafana 服务**。
2. 单击**新建**，根据页面提示配置信息，详情参见 新建 Grafana 实例 。
3. 配置完后，单击**立即购买**即可。

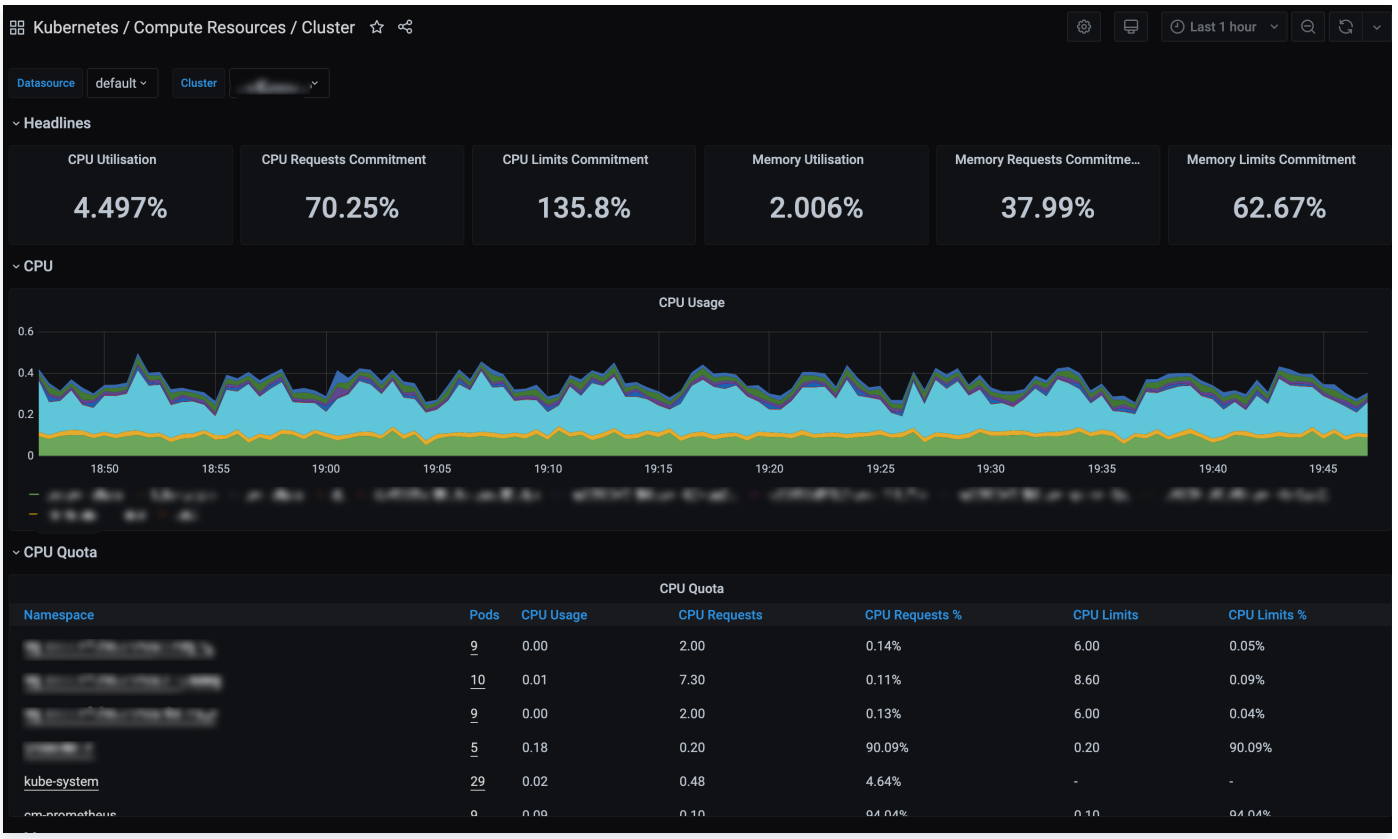
步骤2：配置 Grafana

1. 在 腾讯云可观测平台 > Grafana 服务 中，找到对应的实例，单击操作区中的**管理**。
2. 进入 Grafana 管理页，在左侧菜单栏中单击**配置**。
3. 在 Grafana 配置页左上方单击**编辑**，根据提示修改配置文件。
4. 修改完后，单击**确定**即可。

步骤3：登录 Grafana 查看监控数据

1. 在 腾讯云可观测平台 > Grafana 服务 中，找到对应的实例，单击实例 ID 右侧 。
2. 进入 Grafana，输入 Grafana 登录账号和密码。即可进行 Grafana 可视化大屏操作区。
3. 单击 ，进入监控面板列表页。

4. 在监控面板列表页单击对应的监控图表名称，即可查看监控数据。



说明

如需了解 Grafana 更多操作说明，请参见 Grafana 官网使用手册 。

创建 Dashboard

最近更新时间：2024-05-24 14:25:11

操作背景

Dashboard 提供自定义监控面板的服务，用户可以跨产品跨实例地自定义基础指标，将同一个业务或集群下的资源集中展示。本文为您介绍如何快速创建 Dashboard。

前提条件

1. 登录 腾讯云可观测平台 。
2. 在左侧导航栏中选择 **Dashboard** ，进入 Dashboard 列表页。

操作步骤

步骤1：创建 Dashboard

在 Dashboard 列表页单击左上角的**新建 Dashboard**，进入新建 Dashboard 页面。

步骤2：创建监控图表

1. 在新建 Dashboard 页面的面板中单击**新建图表**创建监控图表。
2. 在图表下方编辑指标信息和图表右侧定义图表名，编辑完后单击右上角的**保存**即可。详情请参见新建图表相关文档 新建指标 。

步骤3：查看监控图表

在完成图表创建后，您可以在 Dashboard 管理页查看您创建的监控图表。

说明：
如需了解更多 Dashboard 设置，请参见 Dashboard 操作指南。

告警管理

最近更新时间：2024-12-03 19:42:42

功能介绍

告警管理提供对监控指标的告警功能。在某些监控指标改变时，可以创建告警来及时通知您采取措施。告警在一定周期内监控某些特定指标，并根据给定的阈值，每隔若干个时间段通过多种方式（微信、短信等）发送告警通知。

操作场景

为了防止云服务器的监控指标达到一定值后，影响您系统的正常运行。您可以对这些监控指标设定告警规则促使告警系统自动检查监控数据，并在监控数据满足条件时，发送告警通知给管理员。帮助您第一时间了解业务异常，并迅速解决。本文主要介绍告警管理模块的功能和操作流程。

操作步骤

创建告警

步骤1：创建告警联系人

1. 登录 访问管理控制台 。
2. 选择**用户列表** > **新建用户**，进入新建子用户页面。
3. 选择**自定义创建** > **仅用于接收消息**。
4. 填写用户信息后，单击**完成**即可。

> **说明**
> 更多用户管理操作步骤请参见 用户管理 。

步骤2：创建告警联系组

1. 登录 访问管理控制台 。
2. 选择**用户组** > **新建用户组**，进入新建用户组页面。
3. 输入用户组名，单击**下一步**。
4. 关联策略可不选择，选择**下一步** > **完成**，即可创建用户组。
5. 在用户组列表中，找到上述步骤创建的用户组，在其右侧操作栏中，单击**添加用户** > 勾选对应的用户。
6. 完成后，单击**确定**即可。

> **说明**
> - 更多用户组操作步骤，请参见 新建用户组 。
> - 云产品等基础资源监控与自定义监控的子账号权限独立。子账号默认没有权限查询其他子账号信息，需开通相关权限查询。

- 若子账号需要在云产品等基础资源监控下查看用户组权限，需主账号登录 访问管理控制台 授权子账号 "QcloudMonitorFullAccess" 权限（若仅开通此权限，云服务资源监控和自定义监控的告警接收对象不同步）。
- 若子账号需要在自定义监控下查看用户组权限，需要主账号在 访问管理模块 授权子账号 "QcloudCamReadOnlyAccess" 权限。

步骤3：创建告警策略:

1. 登录 腾讯云可观测平台 。
2. 选择**告警管理** > **告警配置** > **告警策略**，进入策略管理页面。
3. 单击**新建策略**，配置告警策略。
   - 配置基础选项：策略名称、备注、监控类型、策略类型、策略所属项目、所属标签。
   - 选择告警对象。
   - 设置触发条件范围。
   - 设置告警渠道。
4. 完成以上设置后， 单击**完成**即可。

> **说明**
> 查看更多告警设置，请参见 新建告警策略 。

查看告警

1. 登录 腾讯云可观测平台 。
2. 在左侧导航栏中，单击**监控概览**，进入监控概览页即可查看告警。

> **说明**
> 如需了解更多监控概览信息，请参见 查看告警 。

事件总线
开通事件总线

最近更新时间：2024-11-27 11:45:23

操作场景

腾讯云可观测平台-事件通过 访问管理（Cloud Access Management，CAM）来实现权限管理。CAM 是腾讯云提供的权限及访问管理服务，主要用于帮助客户安全管理腾讯云账户下的资源的访问权限。用户可以通过 CAM 创建、管理和销毁用户（组），并使用身份管理和策略管理控制其他用户使用腾讯云资源的权限。使用事件 EventBridge 前，您需在产品页开通该服务。本文介绍如何开通及使用事件 EventBridge。

操作步骤

步骤一：进入事件页面

首次执行下列步骤后将会出现 CAM 授权页提示。

1. 登录 腾讯云可观测平台 控制台。
2. 进入**事件集**页面，根据控制台提示完成服务开通和角色创建（此操作必须主账号完成）。
3. （可选）登录 CAM 控制台 为子账号授权。
4. 服务角色创建完成后，即可使用事件相关功能，完成后续资源创建。

步骤二： 开通事件服务

主账号第一次使用事件服务时，根据 CAM 要求，您需要开通事件服务角色 **EB_QCSRole**，并授权服务角色相关权限来调用其它服务。

进入**角色管理**页面，根据提示完成授权即可：

步骤三：子账号授权

> **注意**
> 子账号使用前，主账户需要在 CAM 角色管理控制台 先确认是否已经成功创建 `EB_QCSRole` ，如果没有，请按照 主账号授权 授权操作完成授权，否则子用户无法正常使用 EB 控制台和通过 EB 调用其他云上资源。

1. **主账号**登录 CAM 控制台，选择对应子账号，选择**关联策略**。

2. 选择**策略表中关联策略** > **新建自定义策略**。

3. 选择**按策略语法创建** > **空白模板**，自定义名称后，在策略内容中输入下列语法内容：

```
{
    "version": "2.0",
    "statement": [
```

```
        {
            "effect": "allow",
            "action": [
                "apigw:DescribeServicesStatus",
                "apigw:DescribeApi",
                "apigw:DescribeService",
                "apigw:CreateService",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetRole",
                "cam:GetGroup",
                "scf:ListNamespaces",
                "scf:ListFunctions",
                "scf:ListVersionByFunction",
                "scf:ListAliases",
                "scf:CreateFunction",
                "scf:GetFunction",
                "tdmq:CreateSubscription",
                "tdmq:ResetMsgSubOffsetByTimestamp",
                "tdmq:DescribeClusters",
                "tdmq:DescribeEnvironments",
                "tdmq:DescribeTopics",
                "tdmq:DescribeSubscriptions",
                "ckafka:DescribeInstanceAttributes",
                "ckafka:DescribeInstances",
                "ckafka:DescribeTopic",
                "ckafka:DescribeRoute",
                "ckafka:DescribeTopicAttributes",
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:SearchLog",
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "monitor:GetMonitorData",
                "monitor:DescribeAlarmNotices",
                "cam:CreateRole",
                "cloudaudit:*",
                "dts:DescribeSubscribes",
                "es:DescribeInstances",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues"
            ],
            "resource": "*"
        }
    ]
```

```
}
```

4. 完成后，将该自定义策略和预设策略 `QcloudEBFullAccess` 绑定给子账户，子账户即可正常使用服务。

快速投递自定义事件

最近更新时间：2024-07-03 16:59:42

场景简介

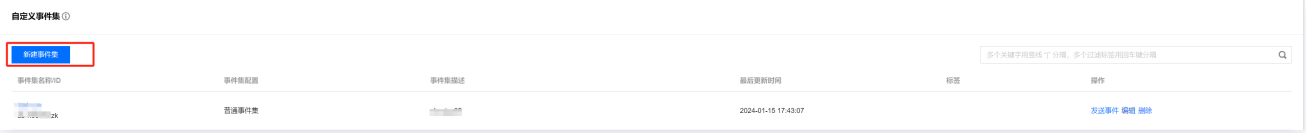
除了官方云服务产生的云产品事件与操作审计事件外，您自己的应用程序产生的事件只能发布到自定义事件集。您可以在 创建自定义事件集 后通过 配置事件连接器 对自定义事件进行投递。您也可以使用 API/SDK 投递您的自定义事件。本文为您展示如何使用**事件连接器**进行自定义事件投递。

> **说明：**
> 除了使用事件连接器外，您也可以通过调用 API 对自定义事件进行投递。具体请参见 API 文档 。

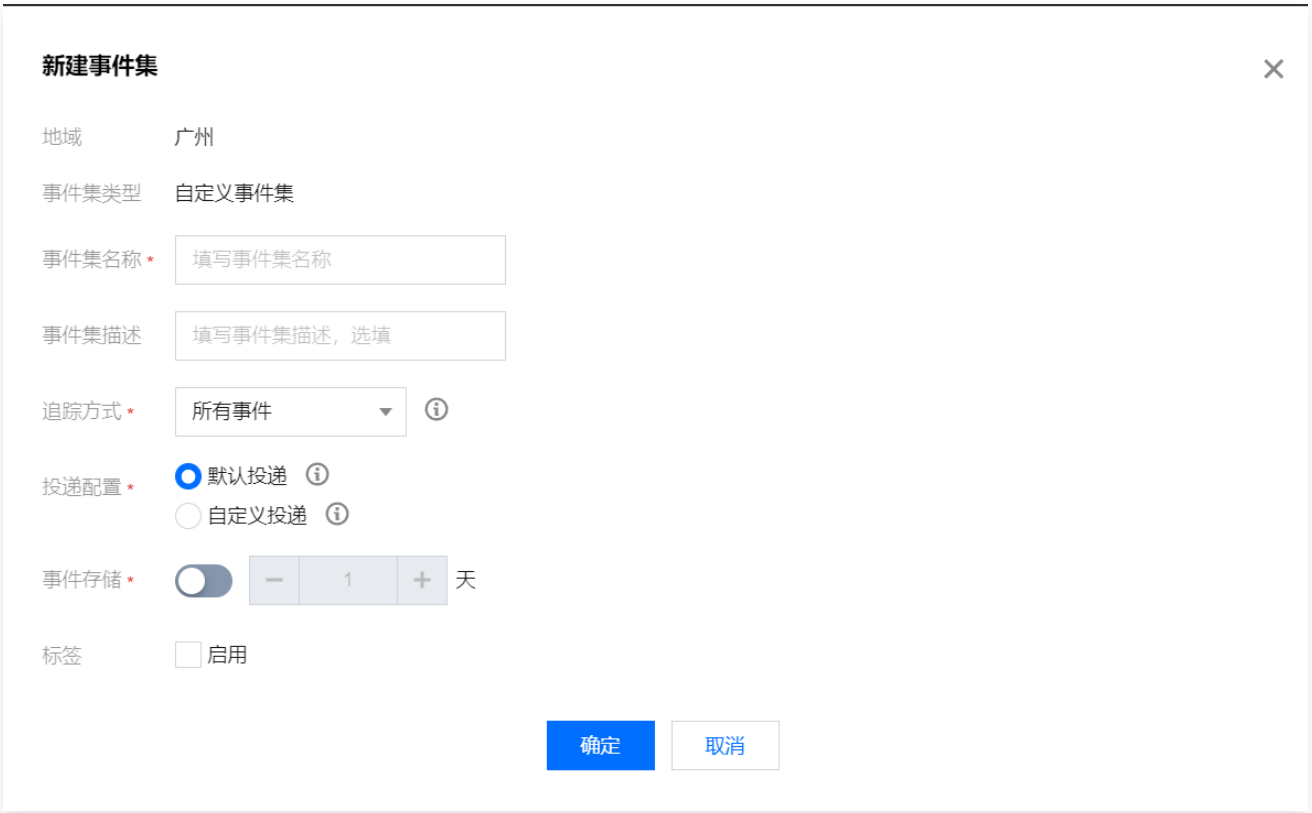
自定义事件投递步骤

步骤1. 创建自定义事件集

1. 登录 事件总线控制台 ，单击**新建事件集**。



2. 填写事件集名称与事件集描述：

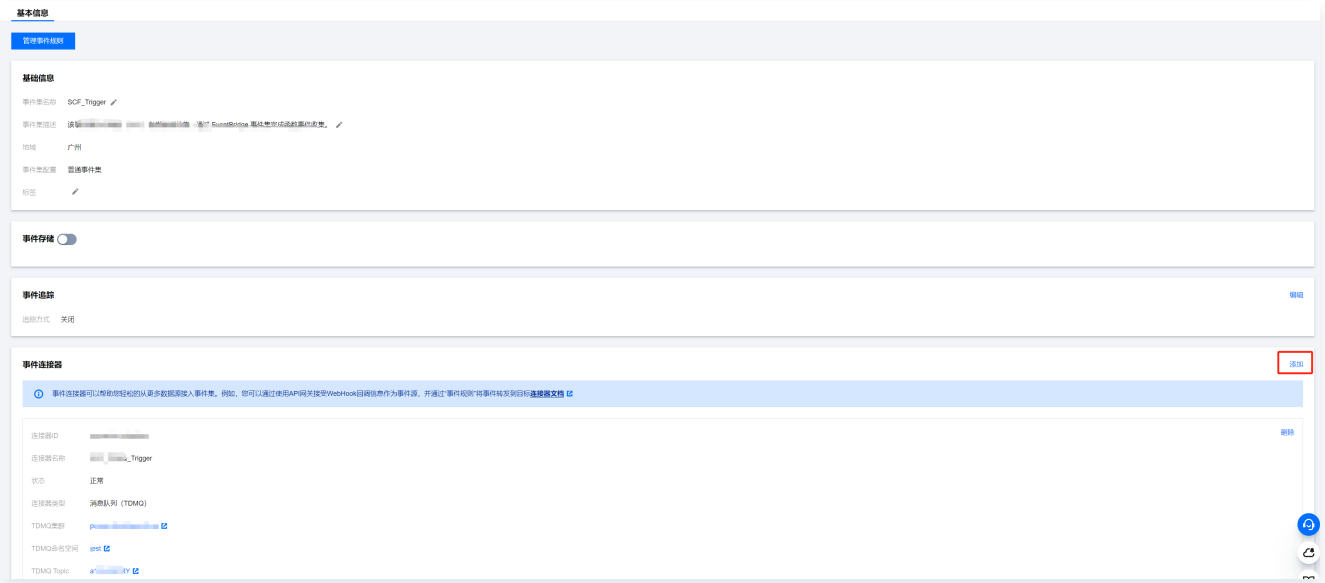


3. 单击**确定**后完成创建。

步骤2. 创建事件连接器

1. 在事件集页单击自定义事件集名称，进入事件集详情页。

2. 进入事件集详情页，单击事件连接器栏中的**添加**。



3. 以 Ckafka 事件连接器为例。**连接器类型**选择**消息队列 (Ckafka)连接器**，其余配置项按照提示填写，单击**确定**完成创建。其他类型的连接器配置方法，请参见 事件连接器 。

4. 单击**确定**。配置完成后，事件连接器将可以从 CKafka 拉取消息，并用消息内容生成一个事件投递到事件集中。以一条内容为 “Hello from Ckafka again!” 的消息为例，连接器将会生成以下事件：

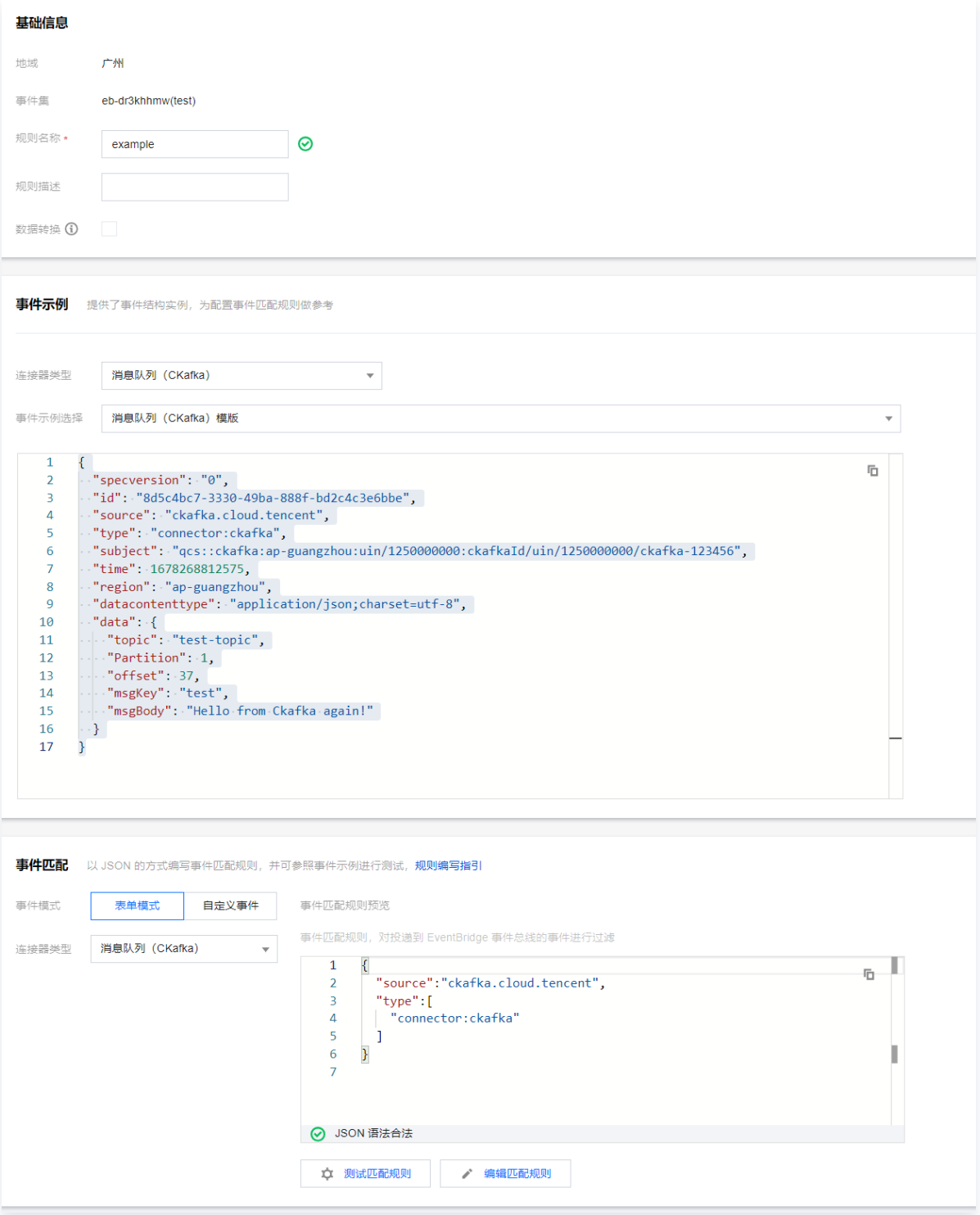
```
{
    "specversion":"1.0",
    "id":"13a3f42d-7258-4ada-da6d-******3b4662",
    "type":"connector:kafka",
    "source":"ckafka.cloud.tencent",
    "subject": "qcs::ckafka:ap-
guangzhou:uin/1250000000:ckafkaId/uin/1250000000/ckafka-123456",
    "time":"1615430559146",
    "region":"ap-guangzhou",
    "datacontenttype":"application/json;charset=utf-8",
    "data":{
        "topic":"test-topic",
        "partition":1,
        "offset":37,
        "msgKey":"test",
        "msgBody":"Hello from Ckafka again!"
      }
}
```

有关事件格式的详细解释，请参见 事件结构 。

> **注意：**
> 目前只支持云上 Ckafka 实例投递，请确认您的 Ckafka 实例没有配置用户名密码等信息，否则连接器可能无法成功获取消息。

步骤3. 创建事件规则

1. 点击左侧导航栏中的**事件规则**。
2. 在**事件规则**页面顶部，选择对应目标事件集，并单击**新建**。
3. 根据页面提示填写相关信息：



事件模式的含义是接收所有来自 Ckafka 连接器的消息。事件模式的具体编写方法，请参见 事件模式 。

4. 单击**下一步**后配置事件目标，可选的事件目标包括 SCF 云函数 、日志服务 CLS 、消息推送 、 Ckafka 及 ElasticSearch 。此处以云函数为例，事件内容将以参数的形式传递给云函数：

此处 test 函数的功能是打印接收到的事件，您可以根据实际业务场景编写您的函数。另外，您也可以通过平台提供的模板函数快速将事件投递到 Ckafka 或下游的 SaaS 服务。具体可参见 SCF 目标投递 。

步骤4. 测试事件

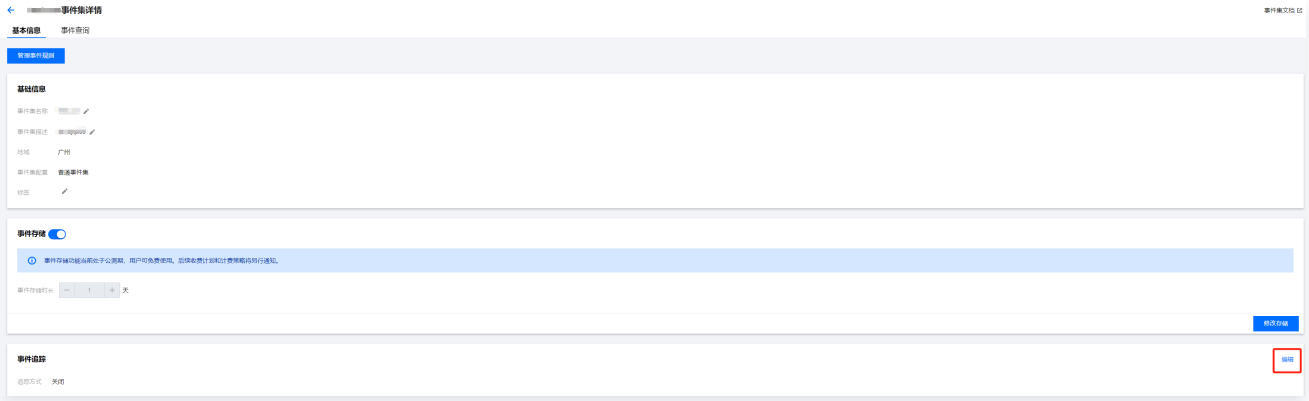
发送一条消息至 target-topic 后，可在对应云函数日志页看到如下信息：

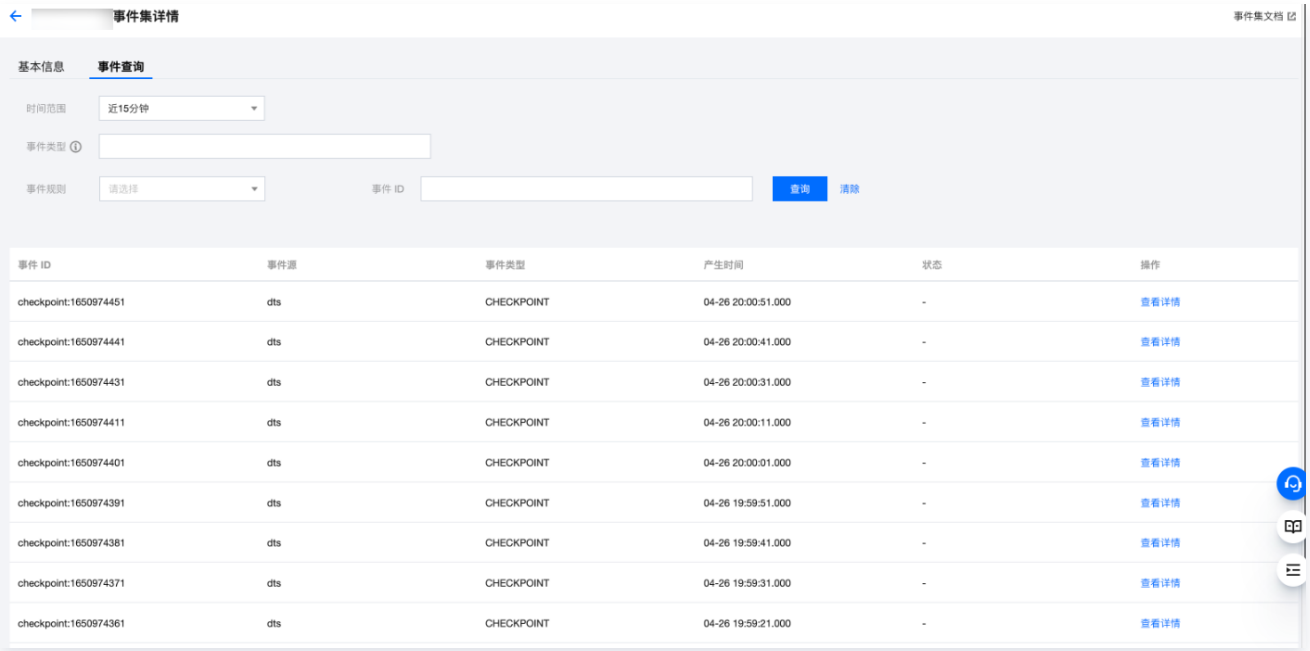
```
START RequestId:79e6d53e-7a98-11ec-8f0d-******4284e2
Received event: {
  "data": {
    "Partition": 1,
    "msgBody": "Hello from Ckafka again!",
    "msgKey": "test",
    "offset": 37,
    "topic": "target-topic"
  },
  "datacontenttype": "application/json;charset=utf-8",
  "id": "13a3f42d-7258-4ada-da6d-******3b4662",
  "region": "ap-guangzhou",
  "source": "ckafka.cloud.tencent",
```

```
  "specversion": "0",
  "status": "",
  "subject": "qcs::ckafka:ap-
guangzhou:uin/1250000000:ckafkaId/uin/1250000000/ckafka-123456",
  "tags": null,
  "time": 1615430559146,
  "type": "connector:kafka"
}
```


步骤5. 追踪历史投递事件

事件总线为用户提供了事件追踪能力。在事件集页单击事件集名称，进入事件集详情页，开启 事件追踪 后，在事件集的事件查询页可以查看事件投递记录。





事件追踪方式若选择“默认投递”，将仅能查询匹配成功但投递到下游失败的事件。若需查询全部事件，请在“事件集-事件追踪”中将投递类型设置为**全部日志**，并勾选“同时上报规则匹配失败日志”。如下图所示：

事件追踪
追踪方式 *
所有事件
投递配置 *
默认投递
自定义投递
日志集 *
请选择
新建日志集
日志Topic *
请选择
新建Topic
投递类型
全部日志
同时上报规则匹配失败日志
确定
取消

快速配置事件告警推送

最近更新时间：2024-05-28 10:58:01

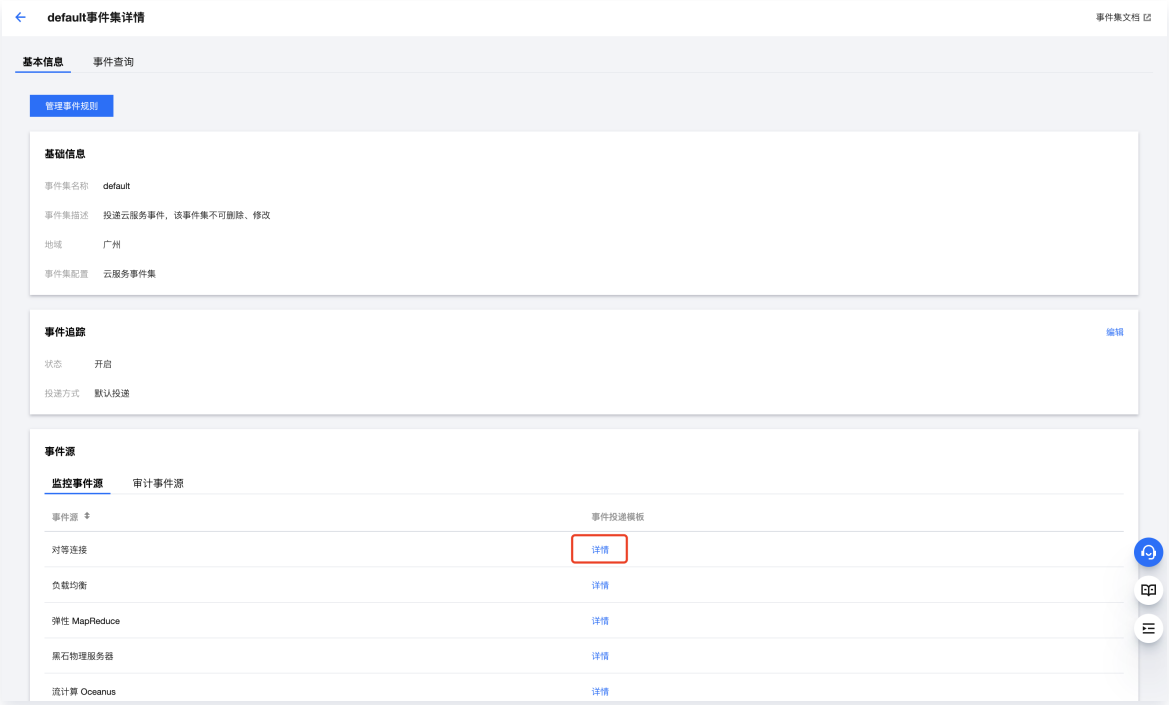
操作场景

开启腾讯云可观测平台-事件服务后，将为您自动在**广州地域**创建默认云服务事件集，所有已接入事件告警的产品所产生的告警事件（监控事件及审计事件）将自动投递至此。您可以通过配置事件规则、投递目标等方式，完成告警链路的配置。

操作步骤

步骤1：查看事件列表

1. 登录 腾讯云可观测平台 > 事件 。
2. 选择事件集所在地域。
3. 单击**云服务事件集 default**，进入云服务事件集 default 详情页，查看目前已经接入云服务事件集的云服务事件。
4. 在**事件连接器**中可以查看目前所有支持告警事件推送的云服务：



单击详情后可以看到目前支持的所有告警事件类型：

示例

以云服务器产生的 ping 不可达事件为例，投递到云服务事件集的标准事件格式如下：

```
{
    "specversion":"1.0",
    "id":"13a3f42d-7258-4ada-da6d-023a333b4662",
    "source":"${ProductName}.cloud.tencent",
    "type":"cvm:ErrorEvent:ping_unreachable",
    "subject":"${CAM 产品六段式}",
    "time": 1615430559146,
    "region":"ap-guangzhou",
    "resource":[
        "qcs::eb:ap-guangzhou:uid1250000000:eventbusid/eventruleid"
    ],
    "datacontenttype":"application/json;charset=utf-8",
    "tags":{
        "key1":"value1",
        "key2":"value2"
     },
    "status":"1",
    "data":{
        "appId":"1250000011",
        "instanceId":"ins-xxxxxxx",
        "projectId":"11",
        "dimensions":{
            "ip":"127.0.0.1"
            },
        "additionalMsg":{
```

```
            "IP":"something unnormal"
            }
        }
    }
```

字段说明

字段	描述	字符串类型
specversion	事件结构体版本（cloudevents 遵循版本，目前仅支持1.0）。	String
id	PUT Event 返回的 ID 信息。	String
type	PUT Event 输入的事件类型。云服务告警事件标准格式为 `${ProductName}:ErrorEvent:${EventType}`，用 “:” 分割类型字段。	String
source	事件来源（云服务事件必传此参数，为 subject 的缩写 ）。云服务默认为 `xxx.cloud.tencent` 。	String
subject	事件来源详情可自定义，云服务默认使用 QCS 描述，例如 `qcs::dts:ap-guangzhou:appid/uin:xxx` 。	String
time	发生事件的时间，0时区毫秒时间戳，例如1615430559146。	Timestamp
datacontenttype	数据类型申明。	String
region	地域信息。	String
status	告警事件状态，分为 “1（异常 error）/0（恢复 recovered）/-（无状态 stateless）” 三类。	String
tags	资源标签。	String
data	PUT Event 输入的事件详情，由各个业务方自定义。	String

步骤2：配置告警事件规则

1. 进入**事件规则**页面，选择对应的事件集后，在事件集下创建事件规则，完成需要配置告警推送的事件筛选。

2. 以 CVM 告警配置为例，您可以选择指定的事件告警类型，也可以选择全部告警事件，详细事件匹配规则请参见 事件模式 。

3. 若要将告警范围限定至某一具体实例，您可以单击**编辑匹配规则**，在事件模式中加入 **subject** 字段。

步骤3： 配置推送目标

事件告警场景下，您可以通过配置**消息推送**投递目标完成告警接收。

消息推送：通过配置消息推送，将您的告警事件推送至指定的消息接收渠道，完成用户及时触达。

配置完成后，即可在腾讯云可观测平台-事件，完成告警事件的查看与推送配置。

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。

OpenTelemetry 前后链路打通

最近更新时间：2024-06-21 15:55:51

腾讯云前端性能监控可联动腾讯云应用性能监控实现前后端一体化监控。本文将以 RUM 和 APM 全链路接入为例，简单介绍如何快速高效改造用户现有的业务，并且通过前后端 trace 打通的案例，实现调用链路追踪，帮助用户解决开发和排障中的实际问题。

背景

业务系统日益复杂

随着互联网产品的快速发展，不断变化的商业环境和用户诉求带来了纷繁复杂的业务需求。业务系统需要支撑的业务场景越来越广、涵盖的业务逻辑越来越多，系统的复杂度也跟着快速提升。与此同时，由于微服务架构的演进，业务逻辑的实现通常需要依赖多个服务间的共同协作。总而言之，业务系统的日益复杂已经成为一种常态。

业务追踪面临挑战

业务系统往往面临着多样的日常客诉和突发问题，“业务追踪”就成为了关键的应对手段。业务追踪可以看做一次业务执行的现场还原过程，通过执行中的各种记录还原出原始现场，可用于业务逻辑执行情况的分析和问题的定位，是整个系统建设中重要的一环。

> **说明：**
> 目前建议前后端均使用 OpenTelemetry 协议的上报方式。

步骤1：接入 RUM SDK

1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中单击**前端性能监控** > **数据总览**。
3. 在数据总览页中单击**应用接入**，根据提示填写配置信息。
4. 配置完后单击**下一步**，进入应用接入页面。根据页面提示接入 RUM SDK，并初始化 SDK。
5. 开启下图对应配置参数，前端方面即已完成链路配置。

```
import Aegis from 'aegis-web-sdk'

const aegis = new Aegis({
  id: 'aZr[redacted]WDx',
  uin: 'xxx',
  reportApiSpeed: true, // 接口测速
  reportAssetSpeed: true, // 静态资源测速
  spa: true,  // 应用页面跳转开启PV计算
  api: {
    apiDetail: true,
    reportRequest: true,
    resHeaders: ['x-req-id', 'server'],
    injectTraceHeader: 'traceparent', // 可选，string，且必须为枚举值 'traceparent', 'sw8', 'b3', 'sentry-trace' 中的一种
    reqHeaders: ['traceparent'], // Array 需要上报的 http 请求 request headers 列表。
    injectTraceUrls: [/consc[redacted].com/], // 可选，Array，数组中传入 string 或者 RegExp。标记哪些请求 url 需要注入 trace 请求头
  },
});
```

> **说明：**
> reqHeaders 需要上报的 HTTP 请求 request header 列表参数，一般与 injectTraceHeader 参数值保持一致。详细接入步骤可参见 应用接入 。

关键配置

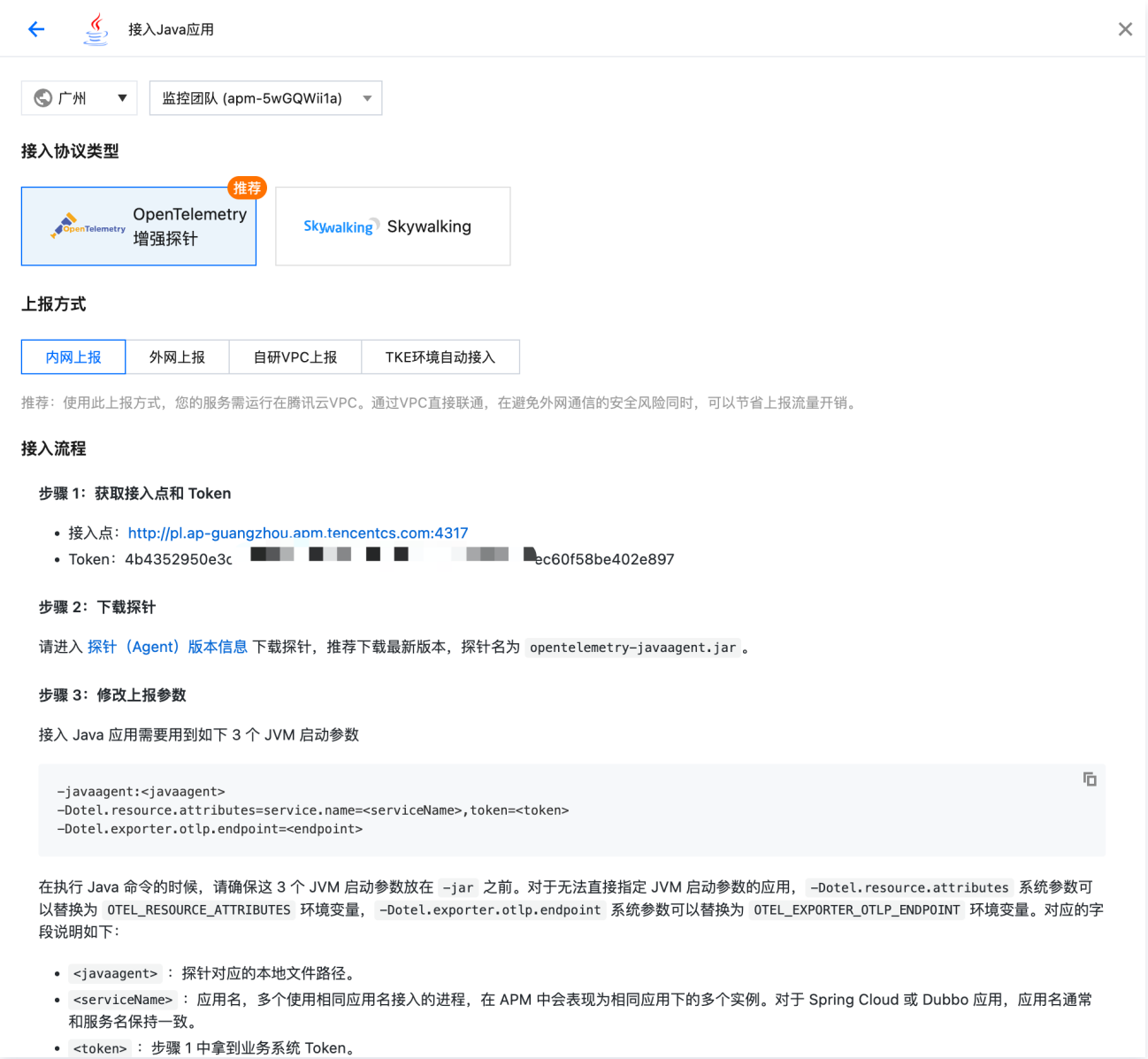
字段 injectTraceHeader 会使 Aegis SDK 给所有监控的接口注入对应的请求头，并且自动生成相关协议字段。目前支持 traceparent、sw8、b3、sentry-trace。

协议	说明
traceparent	OpenTelemetry 的协议字段（推荐使用）
sw8	SkyWalking 的协议字段
b3	Zipkin 跟踪协议字段
sentry-trace	Sentry 的协议字段

步骤2：后端使用 OpenTelemetry 方式接入 APM

1. 在 应用性能监控 > 资源管理 > 资源总览 页面，单击**新建**，创建业务系统。

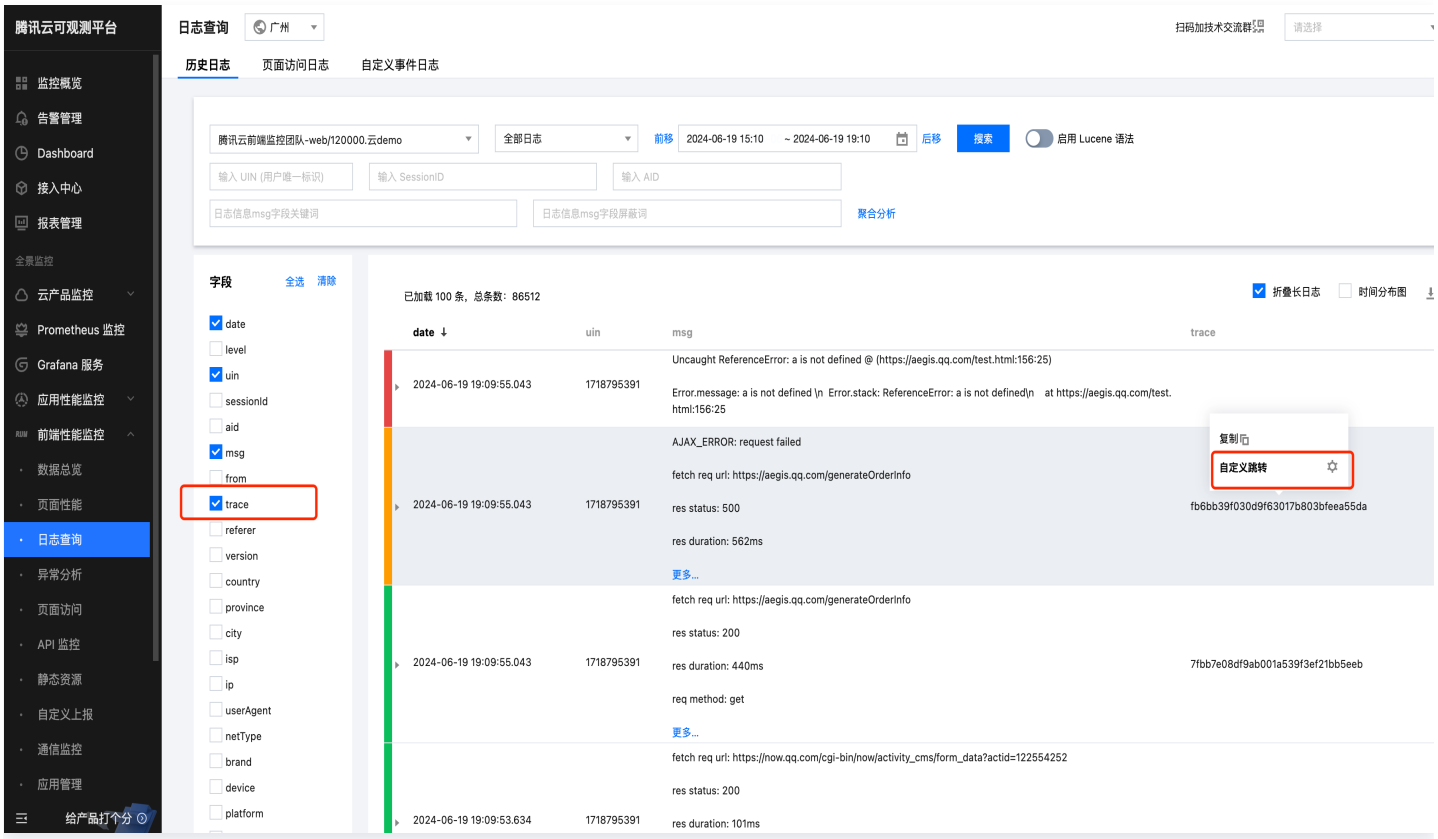
2. 进入 应用监控 页面接入应用，选择对应的业务系统，选择对应的语言和 OpenTelemetry 方式接入应用。



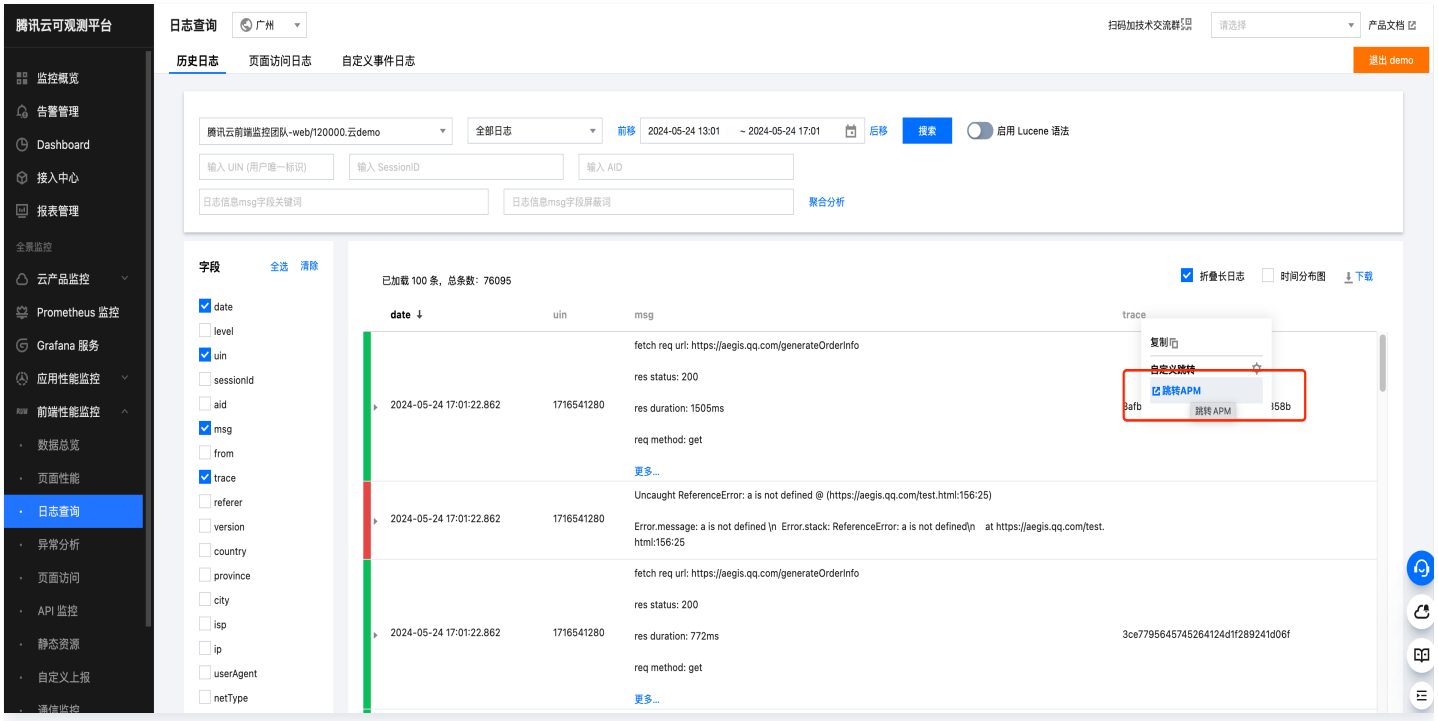
> **说明：**
> 目前支持 Java 、Go、Python、Node、PHP 应用接入方式，详细请参见 应用性能监控 > 接入指南 。

步骤3：使用方法

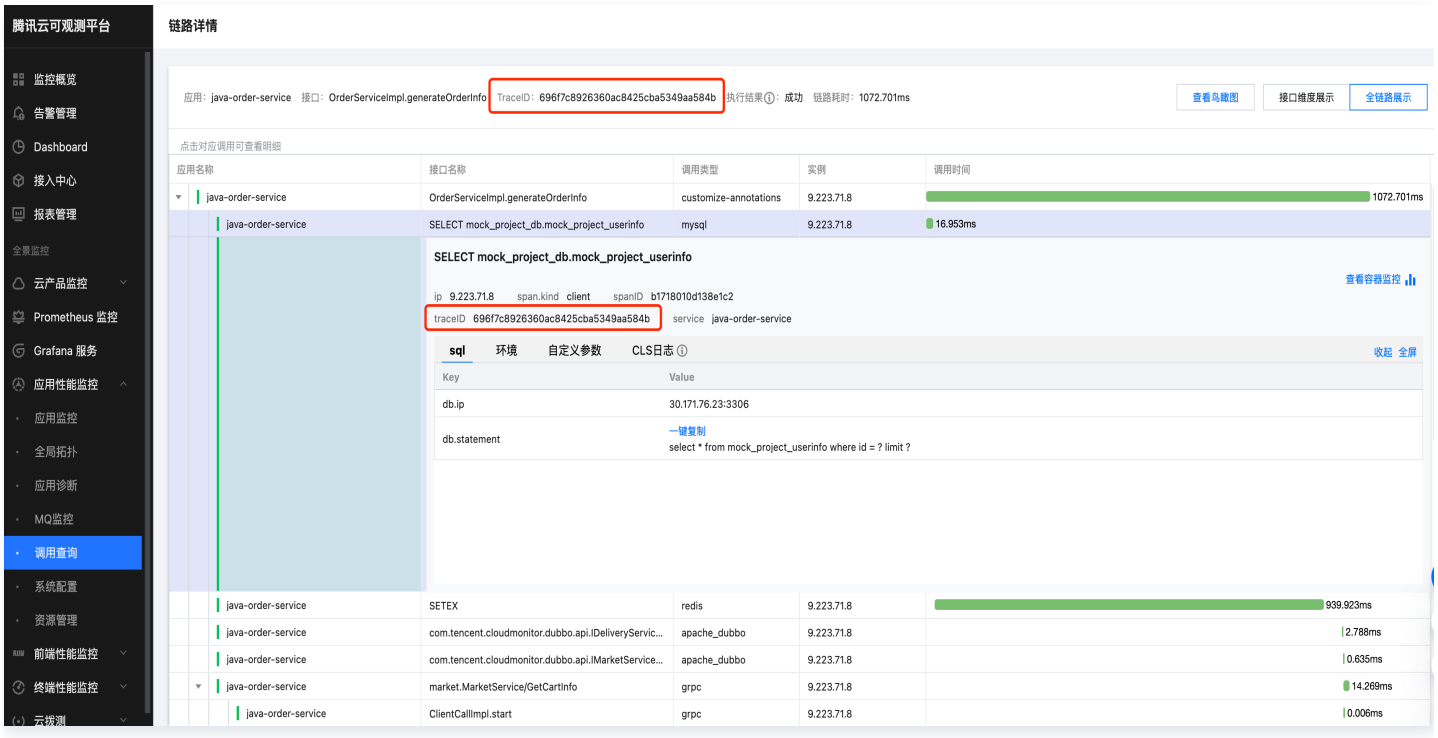
当接口发生请求，产生历史日志时，您在 前端性能监控 > 日志查询 > 历史日志 可以选择字段 Trace，若该日志有 TraceID，可左键单击 TraceID。在弹框中单击**自定义跳转**配置，绑定 APM 业务系统。



绑定后，单击相关链接即可跳转到 APM 链路详情页，查看该条请求在整个链路下的状态。



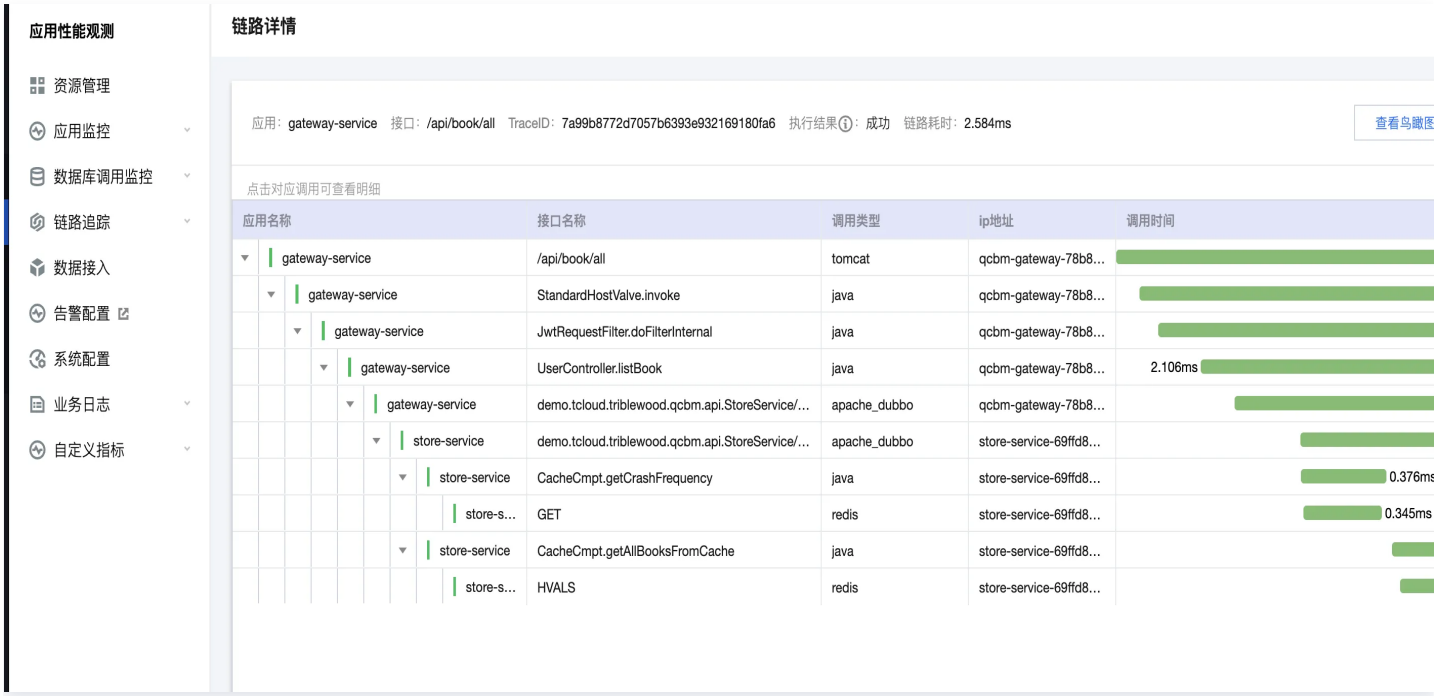
实现调用链路追踪，排查后端链路异常原因，快速定位异常信息。



全链路打通方案

Trace 是如何打通的

接入 OpenTelemetry 后所有打桩的接口都会带上一个请求头，traceparent 或者 b3，这个请求头是 trace 打通的核心，它主要由 traceId 和 spanId 组成，traceId 会跟随这个接口一直向下，所有的内部服务和请求都带有该字段，每层服务生成对应的 span，最后根据 traceId 把所有 span 连起来形成了 trace 数据链。
如果 Aegis SDK 根据用户协议生成对应的 trace 关键字，并且带入到请求的 header 中，发现也可以实现这个效果：我们在项目验证中，去掉了所有 OpenTelemetry 的引入，然后 Aegis SDK 中 mock 了一个 traceparent，发现上报正常，trace 也可以正常生成。



不过对比这个 trace 的图发现这里 trace 起点是 tomcat，不是前端的 HTTP 请求，所以可以判断这里是缺少了 前端的 span。
因为没有在前端接入 OpenTelemetry 的 sdk，所以这里也不难理解为什么 trace 中少了前端的 span 了。
那么如果用户可以接受忽略 trace 缺少前端 span 这种情况，其实当前的方案已经可以满足 90% 的问题查询 case 了，缺点可能就是如果请求链路在网络层或者 CLB 层断掉的话，就没办法发现断掉的原因。除此之外，前端的数据可以在 RUM 上看，后端的链路可以在 APM 上查看。
于是我们对当前 Aegis SDK 进行优化，可以适配不同全链路协议的请求头。
目前适配的协议有：traceparent 、b3 、sw8 、sentry-trace ，根据下游应用选择对应的协议。

SDK 全新接入方案

通过 injectTraceHeader 参数，Aegis SDK 会给所有监控的接口注入对应的请求头，并且自动生成相关协议字段。对比之前需要几十行的接入代码，并且需要非常复杂的 Nginx 转发配置，当前方案极大的减少了开发者的使用负担。从性能的角度来看，当前方案上报的数据量也远远少于之前的方案，开发者也不需要详细理解 trace 协议的原理，只需要配置即可，可以说是比较方便地实现了“无侵入接入”。

```
import Aegis from 'aegis-web-sdk'; // 或者通过 cdn 的方式引入

new Aegis({
    id: 'xxxxxx',
    api: {
        injectTraceHeader: 'traceparent', // 注意这里目前支持 traceparent，b3，
sw8，sentry-trace
        injectTraceIgnoreUrls: ['/v1/traces', /rumt-zh.com/], // 忽略不参与注
入的接口
```

```
    },
});
```

常见问题

前端接口请求头跨域?

```
Access to fetch at '                                    /monitor/rum/log?PID…14%3A53%3A00&env=:1
https:/                                    ey=0WEB0OEX9Y4SQ244' from origin '
https:/                                    .com' has been blocked by CORS policy: Request header field traceparent is
not allowed by Access-Control-Allow-Headers in preflight response.

▶POST                                                   log?PID=146052&level…%3A53%3A24&env=:313
https:/                                    alytics/v2_upload?appkey=0WEB0OEX9Y4SQ244 net::ERR_FAILED

Access to fetch at '                                    /monitor/rum/log?PID…14%3A53%3A00&env=:1
https:/                                    400014144&token=6478159937&product_id=0W…stack=&_dc=0.6994096525482278&er
' from origin 'https://console.cloud.tencent.com' has been blocked by CORS policy: Request header field
traceparent is not allowed by Access-Control-Allow-Headers in preflight response.

▶GET                                                    log?PID=146052&level…%3A53%3A24&env=:313
https://h                                  attaid=00400014144&token=6478159937&product_id=0W…stack=&_dc=0.6994096525482278&er
net::ERR_FAILED

Access to fetch at '                                    /monitor/rum/log?PID…14%3A53%3A00&env=:1
https://h.trace.qq.com/kv?attaid=00400014144&token=6478159937&product_id=0W…stack=&_dc=0.9621460166397551&er
' from origin 'https://console.cloud.tencent.com' has been blocked by CORS policy: Request header field
traceparent is not allowed by Access-Control-Allow-Headers in preflight response.

▶GET                                                    log?PID=146052&level…%3A53%3A24&env=:313
https://h.trace.qq.com/kv?attaid=00400014144&token=6478159937&product_id=0W…stack=&_dc=0.9621460166397551&er
net::ERR_FAILED
```

部分接口具有严格跨域限制，不允许在 header 中添加额外参数，Traceparent 的注入会导致其跨域。可以使用如下两种方式解决：

- 采用 injectTraceIgnoreUrls: Array，数组中传入 string 或者 RegExp。标记不需要注入 trace 请求头的请求 url 。
- 采用 injectTraceUrls: Array，数组中传入 string 或者 RegExp。标记需要注入 trace 请求头的请求 url 进行全链路追踪。（推荐）

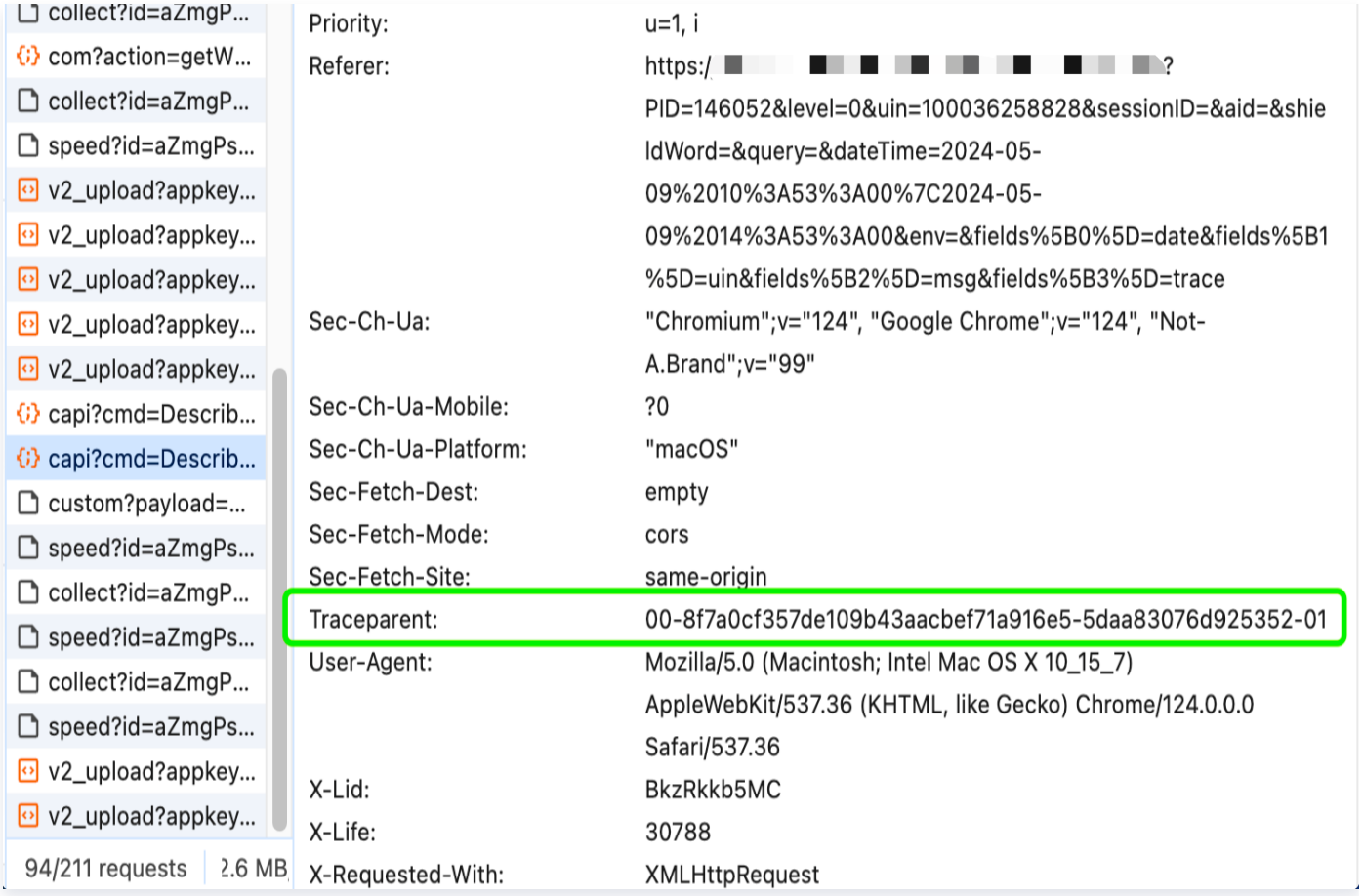
  推荐原因： 用户侧请求环境复杂，采用 injectTraceIgnoreUrls 则需要枚举出全部当前项目关联的 ignore request，恐有遗漏。采用 injectTraceUrls 方式，则链路 request 请求的主动权在用户侧，可灵活配置。

```
new Aegis({
    id: 'xxxxxx',
    api: {
        injectTraceHeader: 'sw8', // 注意这里目前支持 traceparent, b3, sw8, sentry-trace
        // injectTraceIgnoreUrls: ['/v1/traces', /rum-zh.com/], // 忽略不参与注入的接口
        injectTraceUrls: [/console.cloud.tencent.com/], // 标记哪些请求 url 需要注入 trace 请求头
    },
});
```

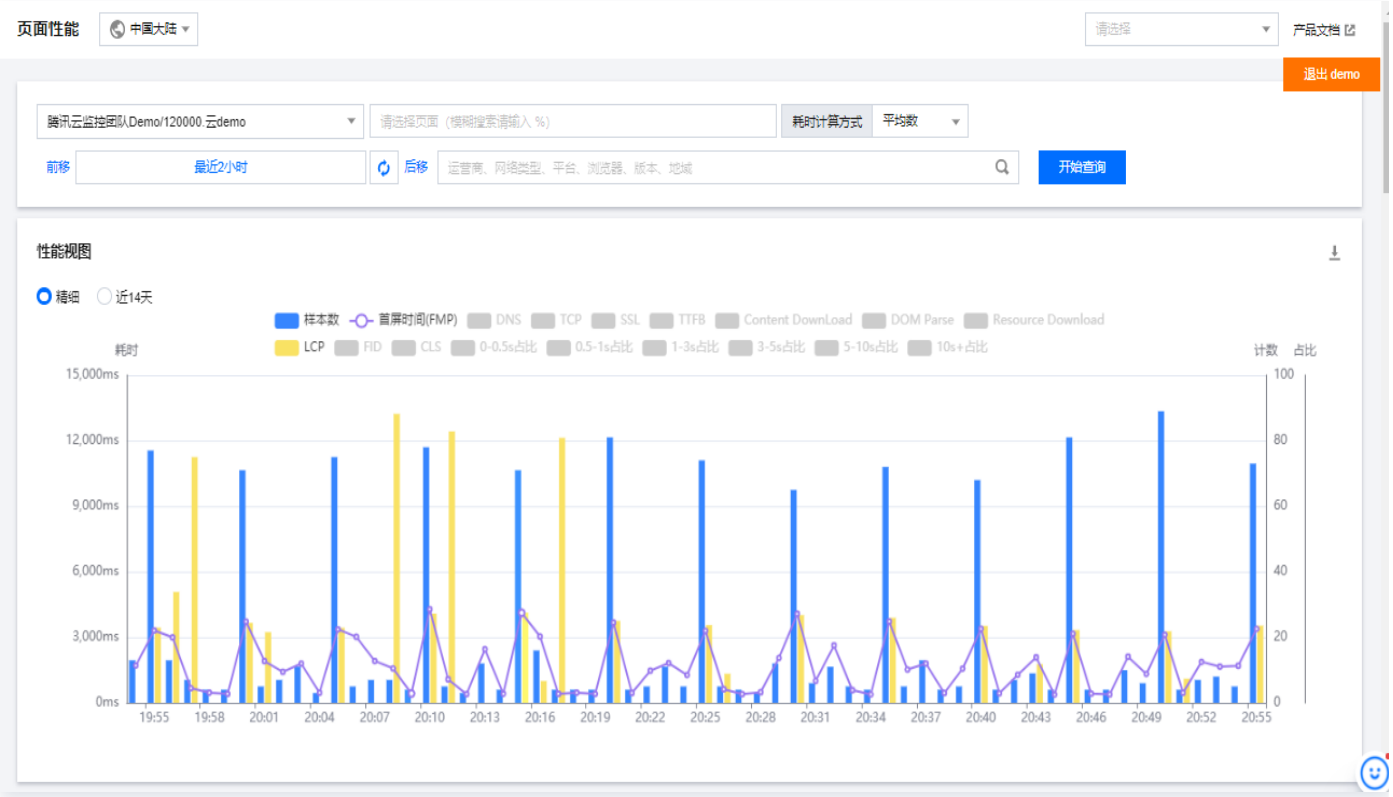
验证 RUM 和 APM 是否上报正常

验证 RUM 是否上报成功

1. 首先，可以在浏览器控制台看到您的业务接口Request headers中存在对应协议参数 ，例如 Traceparent。



2. 其次在 页面性能 中查看是否有数据显示，若有数据上报则表示上报成功。



验证 APM 是否上报成功

在 资源管理页面 查看是否有数据上报量，若有数据上报则表示上报成功。