日志服务

操作指南



腾讯云

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】

腾讯云

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

操作指南
资源管理
日志集
日志主题
指标主题
主题用量监控
机器组
闲置资源冻结
权限管理
子账号授权
可授权的资源类型
CLS 访问策略模板
使用临时密钥访问 CLS
日志采集
采集概述
LogListener 采集器
采集机制
LogListener 使用流程
LogListener 安装和部署
LogListener 安装指南（Linux 版）
LogListener 安装指南（Windows 版）
CVM 与 Lighthouse 批量部署 LogListener
自建 K8s 集群安装 LogListener
LogListener 升级指南
LogListener 配置指南
LogListener 服务日志
LogListener 采集配置导入
LogListener 版本变更
密钥加密存储
采集文本日志
使用单行全文提取模式采集日志
使用多行全文提取模式采集日志
使用单行-完全正则提取模式采集日志
使用多行-完全正则提取模式采集日志
使用 JSON 提取模式采集日志
使用分隔符提取模式采集日志
使用组合解析提取模式采集日志
配置时间格式
自定义元数据
采集自建 K8s 集群日志
使用控制台配置自建 K8s 集群日志采集
使用 CRD 配置自建 K8S 集群日志采集
采集 TKE K8s 集群日志
采集 Syslog
采集 Windows 事件日志
采集 Nginx 访问日志
订阅 MySQL Binlog
使用 Kafka 协议上传日志
使用 Kafka 数据订阅采集日志
匿名日志采集
使用 Logback Appender 上传日志

版权所有：腾讯云计算（北京）有限责任公司

使用 Log4j Appender 上传日志
使用 SDK 上传日志
使用 API 上传日志
IOS/Android 移动端网络探测接入指南
FluentBit 日志上传
Logstash 日志上传
数据导入
导入 COS 数据
采集云产品日志
全球加速日志上传
从 Elasticsearch 导入数据
指标采集
指标上报
日志转指标
云产品指标订阅
K8s 指标采集
主机监控指标采集
日志存储
日志存储概述
日志沉降
指标存储
指标存储概述
兼容 Prometheus 接口
指标预聚合
检索分析（日志主题）
检索分析概述
智能编写查询语句
语法规则
统计分析（SQL）
快速分析
SQL 语法
SELECT 语法
AS 语法
GROUP BY 语法
ORDER BY 语法
LIMIT 语法
WHERE 语法
HAVING 语法
嵌套子查询
JOIN 语法
UNION 语法
INTERSECT 语法
EXCEPT 语法
VALUES 语法
UNNEST 语法
SQL 函数
函数概览
通用聚合函数
逻辑运算符
算术及比较运算符
估算函数
同环比函数
字符串函数

版权所有：腾讯云计算（北京）有限责任公司

日期和时间函数
IP 函数
URL 函数
JSON 函数
正则式函数
数组函数
类型转换函数
条件表达式
数学计算函数
数学统计函数
二进制字符串函数
位运算函数
Lambda 函数
窗口函数
空间几何函数
SQL 案例
采样分析
关联外部数据
跨主题关联分析
索引配置
重建索引
多主题检索
上下文检索
自定义跳转
下载日志
兼容 Elasticsearch 接口
检索分析（指标主题）
语法规则
仪表盘
仪表盘概述
创建仪表盘
创建统计图表
添加图表
表格
时序图
柱状图
饼图
单值图
计量仪
地图
桑基图
词云图
漏斗图
日志
文本
热力图
多语句绘图示例
雷达图
拓扑图
数据转换
单位配置
图表时间配置
过滤与变量

交互事件
添加分组
免密分享仪表盘
预置仪表盘
订阅仪表盘
数据处理
数据加工
数据加工概述
创建加工任务
查看加工详情
数据加工函数
函数总览
可视化函数
键值提取函数
富化函数
流程控制
行处理函数
字段处理函数
字典和列表函数
正则处理函数
时间值处理函数
字符串处理函数
类型转换函数
逻辑和数学函数
编解码函数
IP 解析函数
资源函数
加工实战案例
案例总览
日志结构化-单行文本
日志结构化-JSON
日志结构化-多种格式
日志结构化-分隔符
日志结构化-Grok
日志过滤
日志分发
日志脱敏
Logstash 语句参照表
定时 SQL 分析
概述
新建任务
查看任务
投递与消费
投递与消费概述
投递任务角色授权
投递到 COS
新建投递任务
投递日志原文
投递任务管理
配置告警
投递到 Ckafka
创建投递任务
投递任务管理

配置告警
投递到 DLC
新建投递 DLC 任务
日志消费
Kafka 协议消费
使用 Kafka 协议消费日志
任务监控和告警
消费 Demo-多语言 SDK
消费 Demo-第三方软件
消费 Demo-流处理
自定义消费
指标主题投递
监控告警
监控告警概述
管理告警策略
配置告警策略
触发条件表达式
管理通知渠道组
管理通知内容模板
告警通知变量
管理集成配置
查看告警历史
告警屏蔽
告警认领与升级
云产品中心
云产品中心概述
CLB 云产品中心
CDN 云产品中心
COS 云产品中心
SCF 云产品中心
TKE 云产品中心
DataSight 独立控制台
使用 DataSight 独立控制台
配置控制台内嵌参数
历史文档
低版本 LogListener 操作指南
低版本 LogListener 异常状态排查

版权所有：腾讯云计算（北京）有限责任公司

操作指南
资源管理
日志集

最近更新时间：2025-06-24 15:47:31

概述

日志服务（Cloud Log Service，CLS）以日志主题及日志集的方式管理其上存储的日志数据，日志主题（Topic）是日志数据进行采集、存储、检索和分析的基本单元，日志集则是对日志主题的分类，方便用户管理日志主题。
本文介绍如何通过控制台管理日志集，在实际操作前，建议您阅读 日志主题与日志集，进一步了解相关概念及针对实际场景的使用建议。

前提条件

已登录 日志服务控制台 。

操作步骤

创建日志集

1. 在左侧导航栏中，单击**日志主题**。
2. 在日志主题管理页面，选择地域，单击**管理日志集**。



3. 在右侧展开的日志集管理页面中，单击**创建日志集**。



4. 在弹出的创建日志集窗口中，填写日志集名称及标签。



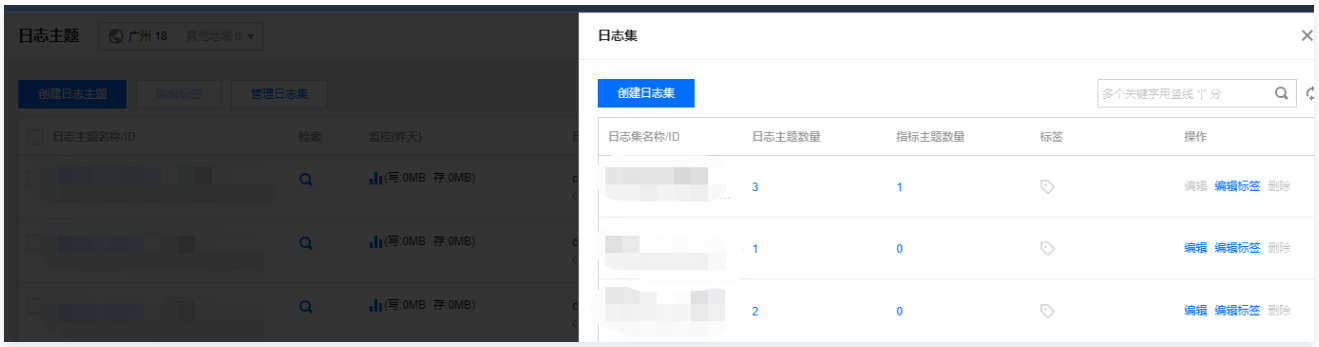
5. 单击**确定**，完成日志集创建。

查看日志集

1. 在左侧导航栏中，单击**日志主题**。
2. 在日志主题管理页面，选择地域，单击**管理日志集**。



3. 在右侧展开的日志集管理页面中，即可查看当前地域下的日志集。



编辑日志集

1. 在左侧导航栏中，单击**日志主题**。
2. 在日志主题管理页面，选择地域，单击**管理日志集**。



3. 在右侧展开的日志集管理页面中，找到您需要编辑的日志集 ID/名称，单击**编辑**。



4. 在弹出的编辑日志集窗口中，修改日志集名称。



5. 单击**确定**，即可保存编辑。

删除日志集

1. 在左侧导航栏中，单击**日志主题**。
2. 在日志主题管理页面，选择地域，单击**管理日志集**。



3. 在右侧展开的日志集管理页面中，找到您需要删除的日志集 ID/名称，单击**删除**。

> **注意：**
> 仅在**日志主题或指标主题数量**为0时，才可删除日志集。若日志集中的主题未清空，则无法对日志集进行删除。



4. 在弹出提示框中，单击**确定**，即可删除当前日志集。

日志主题

最近更新时间：2024-08-15 09:57:01

操作场景

日志服务（Cloud Log Service，CLS）以日志主题及日志集的方式管理其上存储的日志数据，日志主题（Topic）是日志数据进行采集、存储、检索和分析的基本单元，日志集则是对日志主题的分类，方便用户管理日志主题。
本文介绍如何通过控制台管理日志主题，在实际操作前，建议您阅读 日志主题与日志集，进一步了解相关概念及针对实际场景的使用建议。

前提条件

已登录 日志服务控制台 。

操作步骤

创建日志主题

1. 在左侧导航栏中，单击**日志主题**。
2. 在日志主题管理页面，选择日志主题的地域，单击**创建日志主题**。
3. 在弹出的窗口中，填写如下相关信息：
- 日志主题名称：例如 nginx。
- 日志主题 ID：默认自动生成随机 ID，也可自定义 ID。仅支持小写字母、数字和-，且不能以-开头和结尾，长度3至40字符。
- 日志保存策略：
  - 日志接入：默认为标准存储，关于存储类型请参见 存储类型概述 。
  - 保存时间：默认为30天，可自定义设置日志存储时间，日志过期后将自动清除。使用 LogListener 时，如果日志时间未使用采集时间，而将日志内容中的某个时间字段作为日志时间，将以该时间为准决定日志是否过期。
  - 日志沉降：开启后可将历史日志自动沉降为低频存储，详见 日志沉降 。
- 日志集操作：
  - 选择现有日志集：在日志集下拉框中选择目标日志集。
  - 创建日志集：输入新建的日志集名称，可同时为该日志集设定标签，便于从不同维度对资源分类管理。
- 日志主题标签：为当前创建的日志主题设定标签，便于从不同维度对资源分类管理。
- 高级设置：
  - 匿名访问：开启后将支持指定操作匿名访问该日志主题。
  - 匿名操作：开启匿名访问后，目前支持配置以下操作匿名访问该日志主题：
    - JS/HTTP 上传日志：免密通过 **JS、HTTP** 将日志上传至该日志主题。 详情请参见 匿名日志采集 。
    - Kafka 协议上传日志：免密通过 **Kafka** 协议将日志上传至该日志主题。详情请参见 匿名日志采集 。
  - 筛选条件：当开启**匿名访问**时， 支持仅允许满足特定条件的请求匿名访问该日志主题。当前支持配置以下筛选条件：
    VPC_ID：请求的来源 VPC。仅支持腾讯云 VPC，且 VPC 所在地域需与当前日志主题一致。

    > **注意：**
    > 当前仅 Kafka 协议匿名上传日志支持配置筛选条件。

  - 日志主题描述：描述信息，方便后续了解该日志主题用途选择现有日志集：在日志集下拉框中选择目标日志集。
  - 主题分区：
    - 初始分区数量：创建主题时自动创建的分区数量，支持1~10，默认为1。主题分区说明详见 主题分区 。
    - 自动分裂：是否根据主题写流量大小自动添加分区，默认开启。
    - 最大分区数量：开启自动分裂时，最大的分区数量，默认50。
4. 单击**确定**，即可创建日志主题。
   日志主题新增成功后，可在日志管理页面单击日志主题 ID/名称，查看日志主题的详情信息。

> **注意：**
> - 若收集云服务器或其他云产品日志，日志服务地域建议选择相同地域。

- 日志主题创建后，无法修改所属的地域及日志集，无法修改接入存储类型。

编辑日志主题

1. 在左侧导航栏中，单击**日志主题**。
2. 在日志主题管理页面，找到您需要编辑的日志主题 ID/名称，在列表右侧单击**编辑**。
3. 在弹出的窗口中，修改日志主题基本信息。
4. 单击**确定**，日志主题信息即可完成更新。

删除日志主题

1. 在左侧导航栏中，单击**日志主题**。
2. 在日志主题管理页面，找到您需要删除的日志主题 ID/名称，在列表右侧单击**删除**。

> **说明：**
> 页面左上角可切换至不同地域分别删除其内的日志主题。如需删除所有的日志主题，建议使用具备全部资源权限的账号进行操作，否则部分日志主题可能由于无权限查看而无法全部删除。

3. 在弹出的窗口中，单击**确定**，即可删除当前日志主题。

> **注意：**
> 日志主题一旦删除，主题配置和日志数据不可恢复，请谨慎操作！

指标主题

最近更新时间：2024-08-08 15:20:21

操作场景

日志服务（Cloud Log Service，CLS）以 指标主题 和日志集的方式管理其存储的指标数据，指标主题（Metric Topic）是指标数据进行采集、存储和查询的基本单元，日志集则是对指标主题的分类，方便用户管理指标主题，日志集下可同时包含指标主题及日志主题。本文为您介绍如何管理指标主题。

> **说明：**
> 指标主题已于2024年7月1日结束公测并正式收费，详见 计费概述 。

前提条件

已登录 日志服务控制台 。

操作步骤

创建指标主题

1. 在左侧导航栏中，单击**指标主题**。
2. 在指标主题管理页面，选择指标主题的地域，单击**创建指标主题**。
3. 在弹出的窗口中，填写如下相关信息：
   - 指标主题名称：例如 nginx_metric。
   - 指标主题 ID：默认自动生成随机 ID，也可自定义 ID。仅支持小写字母、数字和-，且不能以-开头和结尾，长度3至40字符。
   - 指标保存策略：
     保存时间：默认为30天，可自定义设置数据存储时间，数据过期后将自动清除，也可永久保存。
   - 日志集操作：
     - 选择现有日志集：在日志集下拉框中选择目标日志集。
     - 创建日志集：输入新建的日志集名称，可同时为该日志集设定标签，便于从不同维度对资源分类管理。
   - 指标主题标签：为当前创建的指标主题设定标签，便于从不同维度对资源分类管理。
   - 高级设置：
     - 指标主题描述：描述信息，方便后续了解该指标主题用途。
     - 主题分区：
       - 初始分区数量：创建主题时自动创建的分区数量，支持1~10，默认为1。主题分区说明详见 主题分区 。
       - 自动分裂：是否根据主题写流量大小自动添加分区，默认开启。
       - 最大分区数量：开启自动分裂时，最大的分区数量，默认50。
4. 单击**确定**，即可创建指标志主题。
   指标主题新增成功后，可在指标主题管理页面单击创建的指标主题名称/ID，查看指标主题的详细信息。

> **注意：**
> 指标主题创建后，无法修改所属的地域及日志集。

编辑指标主题

1. 在左侧导航栏中，单击**指标主题**。
2. 在指标主题管理页面，找到您需要编辑的指标主题名称/ID，单击**编辑**。
3. 在弹出的窗口中，修改指标主题基本信息。
4. 单击**确定**，指标主题信息即可完成更新。

删除指标主题

1. 在左侧导航栏中，单击**指标主题**。
2. 在指标主题管理页面，找到您需要删除的指标主题名称/ID，单击**删除**。
3. 在弹出的窗口中，单击**确定**，即可删除当前指标主题。

版权所有：腾讯云计算（北京）有限责任公司

> **注意：**
> 指标主题一旦删除，相关配置和数据不可恢复，请谨慎操作！

主题用量监控

最近更新时间：2024-10-18 10:20:11

操作场景

日志服务（Cloud Log Service，CLS）提供两种方案对日志主题及指标主题的使用量进行监控，包括流量及存储量等，帮助用户掌握各个主题的使用情况，合理控制产品使用成本。

监控方案概览

监控方案	功能入口	费用	方案优势	方案劣势	适用场景
基础方案：腾讯云可观测平台方案	腾讯云可观测平台	免费	监控数据与 CVM、CLB、CDN 等云产品的监控数据都集中在腾讯云可观测平台中，数据较集中。	• 统计分析能力较弱，不支持按用量对主题进行排序，不支持按照日志集对主题用量进行汇总。 • 无指标主题用量监控。	主题数量较少，无统计分析需求，只需要简单的查询指定单个主题的用量情况。
【推荐】高级方案：日志服务自监控方案	CLS 主题用量监控_总览仪表盘	免费	• 统计分析能力强，预置了多种维度的分析仪表盘，例如查询 Top 用量主题、按日志集对主题用量进行汇总，还可通过 PromQL 进行自定义统计分析。 • 指标全面，涵盖日志主题及指标主题各类用量监控指标。 • 查询指标及配置告警均可在日志服务产品内部完成，无需跳转。	暂时仅支持北京、上海、广州、南京地域，目前以白名单方式供部分用户使用。如需其它地域或开通白名单，请 联系我们 。	主题数量较多，需要从多种维度查找出使用量多、波动异常的主题或日志集。

基础方案：腾讯云可观测平台方案

主题用量监控指标已上报至腾讯云可观测平台，使用该平台的 云产品监控 功能查询指标并配置告警，操作步骤如下：

1. 打开 腾讯云可观测平台 控制台，可查看日志主题列表及对应的用量监控指标。
2. 点击日志主题名称，查看该主题的详细用量监控指标，支持的监控指标详见 日志服务监控指标 。
3. 针对以上指标，可配置告警，操作步骤详见 新建告警策略 。

高级方案：日志服务自监控方案

主题用量指标自动上报至日志服务 自监控指标主题 中，位于日志集 cls_service_logging 下，名称为 cls_service_metric，每个地域一个主题，存储该地域所有日志主题和指标主题的用量指标。该主题不计费，不可编辑，不支持上报自定义指标。

> **注意：**
> 暂时仅支持北京、上海、广州、南京地域，目前以白名单方式供部分用户使用。如需其它地域或开通白名单，请 联系我们 。

操作步骤如下：

1. 打开 CLS 主题用量监控_总览 仪表盘，可查看当前地域所有主题的汇总指标，并可查看 Top 使用量日志集和 Top 使用量主题。
2. 在 Top 使用量日志集中，点击 LogsetID，可查看该日志集下主题的汇总指标及 Top 使用量主题。
3. 在 Top 使用量主题中，点击 TopicID，可查看该主题的详细用量监控指标。
4. 针对以上指标，可配置告警，操作步骤详见 配置告警策略 。

也可在指标主题 cls_service_metric 中直接查询主题用量指标，如下表。指标含义详见 计费概述 。

指标分类	指标名称	指标中文名称	指标单位
流量	LogWriteTraffic	日志主题标准写流量	Byte
	LogColdWriteTraffic	日志主题低频写流量	Byte
	LogIndexTraffic	日志主题标准索引流量	Byte

	LogColdIndexTraffic	日志主题低频索引流量	Byte
	MetricWriteTraffic	指标主题写流量	Byte
	IntranetReadTraffic	内网读流量	Byte
	InternetReadTraffic	外网读流量	Byte
存储量	LogStorage	日志主题标准日志存储量	Byte
	LogColdStorage	日志主题低频日志存储量	Byte
	LogIndexStorage	日志主题标准索引存储量	Byte
	LogColdIndexStorage	日志主题低频索引存储量	Byte
	MetricStorage	指标主题存储量	Byte
数据处理	ETLProcessingTraffic	数据加工量	Byte
其它	LogUploadCount	上报日志条数	条
	MetricUploadCount	上报指标条数	条
	RequestCount	服务请求次数	次

版权所有：腾讯云计算（北京）有限责任公司

机器组

最近更新时间：2024-12-27 15:42:12

操作场景

机器组是腾讯云日志服务（Cloud Log Service，CLS）中 LogListener 所采日志的服务器对象，通过机器组的方式来配置 LogListener 采集日志的服务器。一般而言，可以根据不同业务场景划分不同的机器组，方便您管理日志服务。

创建机器组

通过关联机器 IP 地址创建机器组

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**机器组管理**。
3. 选择您日志服务所在的地域，例如广州，单击**新建机器组**。
4. 在弹出的窗口中，设置如下信息。



- 机器组名称：例如 cls_test。
- 系统环境：请选择加入该机器组的机器的系统环境。支持选择 Windows 或 Linux。
- 机器组类型：选择**机器组 IP 地址**。
- IP 地址：输入需要加入与机器组的机器 IP 地址。

> ⚠ **注意：**
> - 所关联机器需已安装 LogListener，详情请参见 LogListener 安装指南（Linux版）或 LogListener 安装指南（Windows版）。
> - 每行填写一个 IP 地址。

- 所填 IP 必须与机器 LogListener 安装目录下 `/etc/loglistener.conf` 中的 group_ip 保持一致。
- 若机器配置了机器标识，则仅可以通过 机器标识创建机器组，不可通过 IP 创建。您可通过查看机器 LogListener 安装目录下 `/etc/loglistener.conf` 中的 group_label 是否为空，判断该机器是否配置机器标识。

- LogListener 自动升级：默认开启，请根据实际需求进行配置。
- LogListener 离线自动剔除策略：请参见 机器离线自动剔除 。
- LogListener 服务日志：默认开启 Loglistener 服务日志，记录 Loglistener 运行状态、采集状况等重要日志，详情请参见 LogListener 服务日志 。

5. 单击**确定**，即可完成创建。

通过机器标识创建机器组

由于每次 IP 变更需要修改对应的机器组配置，如果您的机器 IP 地址经常变动，利用机器 IP 配置机器组操作较为繁琐。
CLS 支持通过机器标识动态配置机器组，您只需要在 Loglistener 的配置信息中填入机器标识，CLS 即可识别并自动将机器添加至机器组。

> **注意：**
> 通过配置机器标识创建机器组仅支持 LogListener 2.3.0及以上版本，升级详情请参见 LogListener 升级指南 。

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**机器组管理**。
3. 选择您日志服务所在的地域，例如广州，单击**新建机器组**。
4. 在弹出的窗口中，设置如下信息。



- 机器组名称：例如 cls_test。
- 系统环境：请选择加入该机器组的机器的系统环境。支持选择 Windows 或 Linux。

- 机器组类型：选择**机器标识**。
- 机器标识：输入机器的标识。

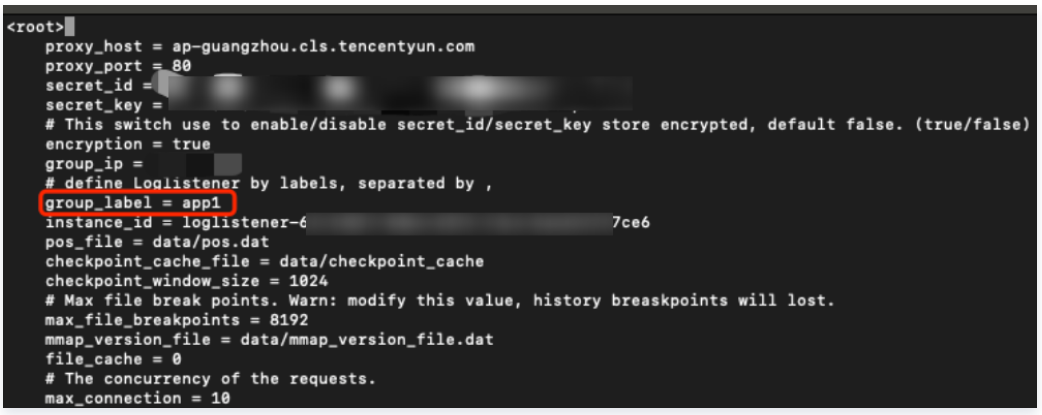
> ⚠ **注意：**
> - 所关联机器需已安装 LogListener，详情请参见 LogListener 安装指南（Linux版）或 LogListener 安装指南（Windows版）
> - 每行填写一个 IP 地址。

- LogListener 自动升级：建议开启 LogListener 自动升级，详情请参见 LogListener 升级指南。
- LogListener 离线自动剔除策略： 请参见 机器离线自动剔除。
- LogListener 服务日志：默认开启 LogListener 服务日志，记录 LogListener 运行状态、采集状况等重要日志，详情请参见 LogListener 服务日志。

5. 单击**确定**。
6. 登录进行机器标识的目标机器，并执行如下命令，打开 Loglistener 安装目录下的 `/etc/loglistener.conf` 文件。此处安装目录以 `/usr/local` 为例：

```
vi /usr/local/loglistener/etc/loglistener.conf
```

7. 按下 **i** 键，进入编辑模式。
8. 找到 group_label 参数，填写您自定义的机器标识（多个标识以英文逗号分隔）。



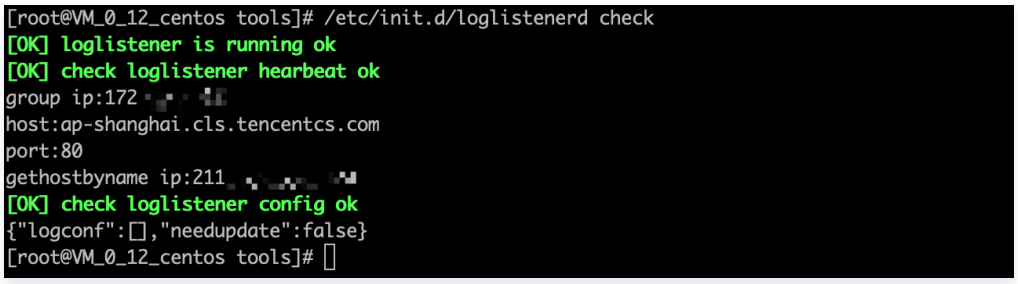
9. 按下 **Esc** 键，退出编辑模式，输入 **:wq**，并按下 **Enter** 键，保存设置。
10. 执行如下命令，重启 LogListener，即可完成创建。

通过 systemd 执行

适用于 Loglistener 2.8.3以上并且操作系统具有 systemd。

```
systemctl restart loglistenerd
```

执行 `/etc/init.d/loglistenerd check` 查看是否重启成功：



不通过 systemd 执行

适用于 LogListener 2.8.3以下或者 LogListener 2.8.3以上但是操作系统没有 systemd。

```
/etc/init.d/loglistenerd restart
```

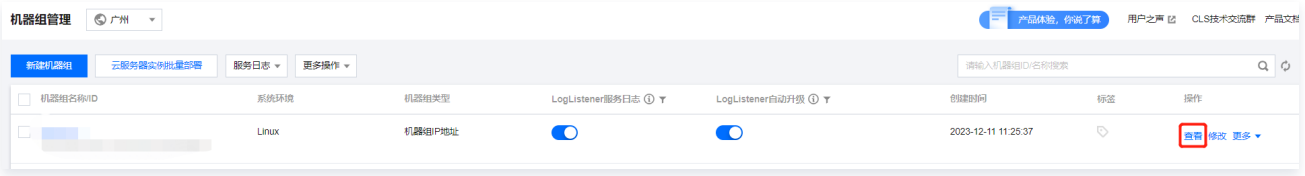
重启成功返回样例：

```
[OK] loglistener is running, ip is 1[redacted]21
[OK] restart loglistener success
```

查看机器状态

机器组与日志服务系统之间采用心跳机制保持连接，成功安装过 LogListener 的机器组会定时向日志服务发送心跳。

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**机器组管理**。
3. 在机器组管理页面，找到需要操作的机器组，单击**查看**。



4. 在弹出的窗口中，即可查看机器组的状态。
5. 通过查看机器组的状态可识别当前该机器是否工作正常。若状态显示正常，则说明您的服务器可以与腾讯云日志服务正常通信。



机器离线自动剔除

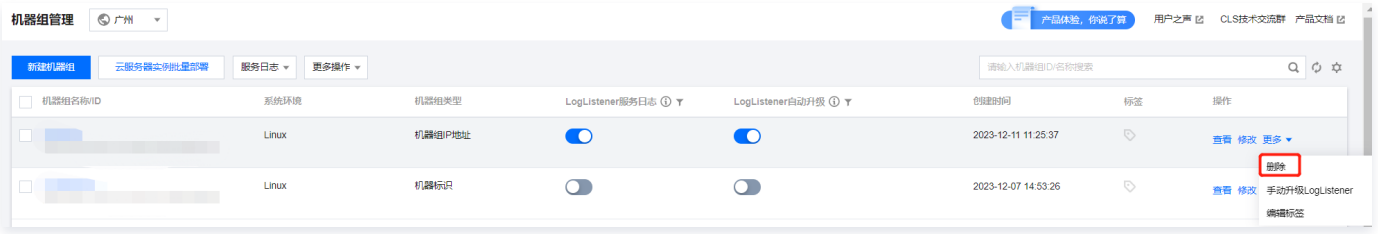
机器标识类机器组关联的机器若出现了离线， 将默认在离线**30天**后自动将离线机器从机器组中剔除， 并取消关联。 若要缩短离线剔除周期， 可执行以下步骤：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**机器组管理**。
3. 选择日志服务所在的地域，例如广州。
4. 单击**新建机器组**，或选择已有的机器组，单击**修改**。
5. 找到 **LogListener 离线自动剔除策略**配置项， 通过设定天数修改机器离线周期。



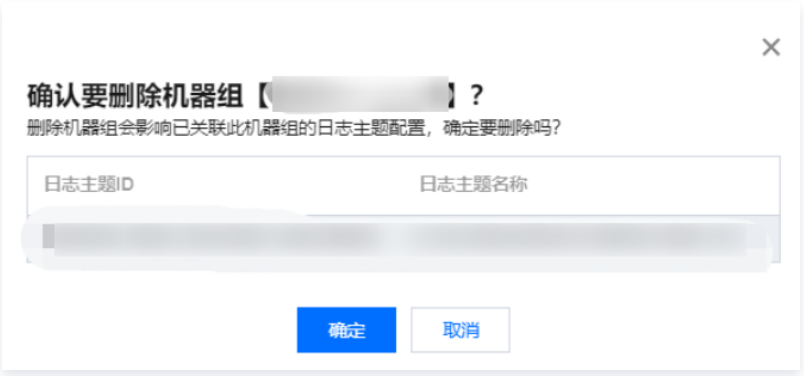
删除机器组

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**机器组管理**。
3. 在机器组管理页面，选择需要删除的机器组，单击**更多 > 删除**。



4. 在弹出的提示窗口，单击**确认**，完成机器组删除。

> **注意：**
> 机器组一旦删除，所关联的日志主题将无法继续采集日志。



闲置资源冻结

最近更新时间：2025-06-24 15:47:31

为了避免用户暂时停用日志服务或未及时清理日志主题期间，资源闲置产生额外的主题分区费用。日志服务支持闲置资源自动冻结功能，您可以根据业务场景，选择是否需要开启闲置资源冻结开关。
当您开启了闲置资源冻结开关后，如果日志服务连续15天日志存储量为0时，视为日志主题分区资源闲置，将自动冻结所有主题分区资源，并从第16天起停止主题分区计费。冻结后，当有新数据写入时，将自动恢复主题分区，并重新开始计费，不影响正常使用。

操作步骤

1. 登录 日志服务控制台 。
2. 在**概览**页面，点击用量统计栏顶部的**闲置资源冻结**，进入配置页面。
3. 单击**自动冻结**开关，开启或关闭该功能。（CLS 已默认开启闲置资源冻结）



4. 单击**确定**，完成闲置资源冻结开关的开启或关闭。

相关问题

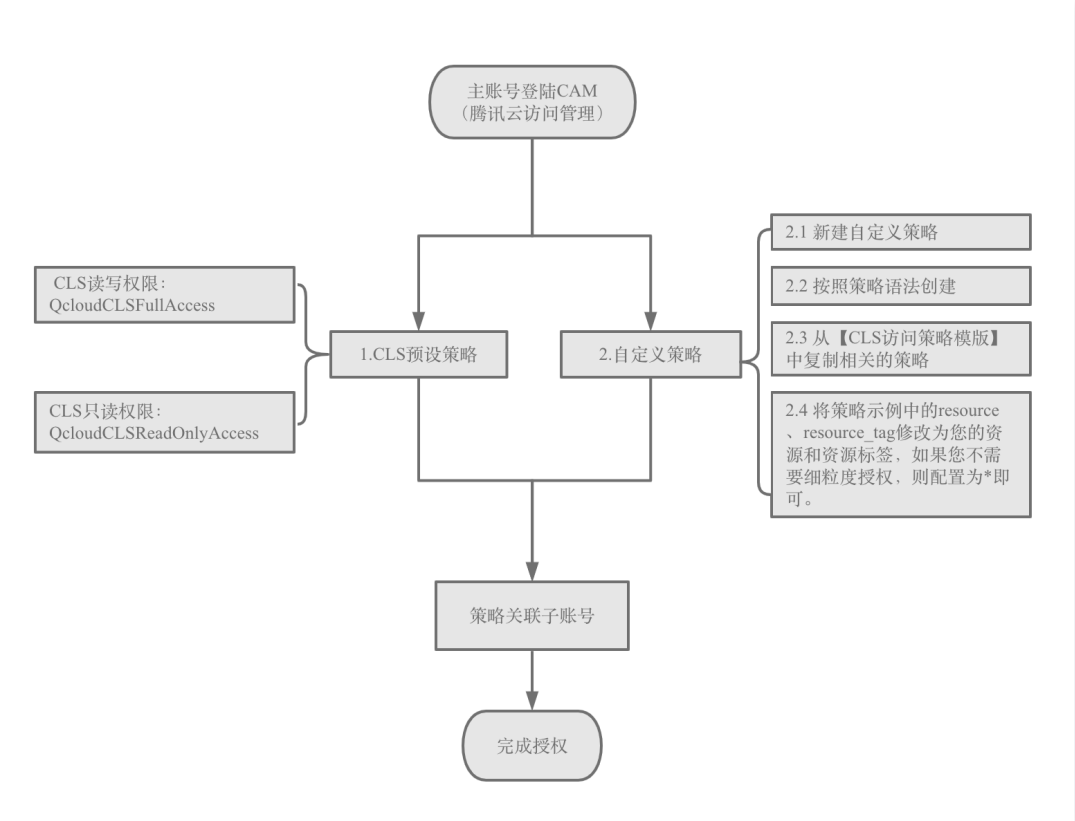
- 什么是主题分区?
- 主题分区如何计费?

权限管理
子账号授权

最近更新时间：2024-10-18 10:20:11

简介

日志服务（Cloud Log Service，CLS）通过访问管理（Cloud Access Management,CAM）对子账号进行授权。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，CAM 策略的详细信息及使用方式请参见 CAM 策略 文档。



预设权限策略

日志服务预设两条权限策略，可满足最基本的权限管理需求。

- CLS读写权限（QcloudCLSFullAccess）：具备日志服务所有功能及所有资源的权限，例如创建日志主题、修改索引配置、删除日志主题、检索日志、上传日志等。
- CLS只读权限（QcloudCLSReadOnlyAccess）：仅具备数据查看权限，不能执行新建、编辑或删除类型的操作。

配置方式参见 授权管理 文档。

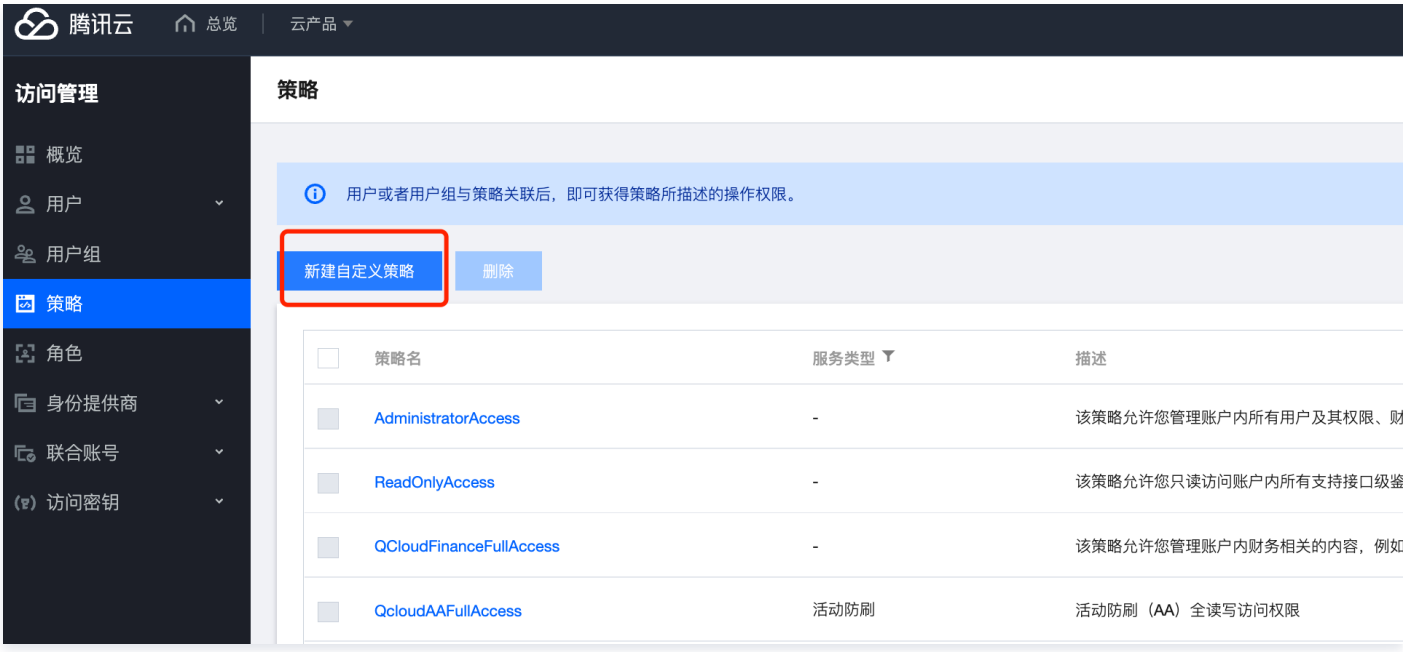
自定义权限策略

预设策略无法满足权限管控需求时，可通过自定义权限策略实现更细粒度的权限划分，例如仅允许某个用户查看特定日志主题的数据。
自定义权限策略主要由两部分组成：

- 操作（Action）：用户可进行的操作，例如：检索日志、修改索引配置、上传日志、创建告警策略等。
- 资源（Resource）：可操作的资源范围，例如：特定的日志主题、仪表盘、数据加工任务等。

日志服务支持授权的资源类型及相关接口参见 可授权的资源类型，配置方式参见 创建自定义策略 。
自定义权限策略配置较为复杂，实际使用过程中可参见 CLS 访问策略模板，这些示例能够满足大多数的权限管理需求，也可以基于这些策略示例再进行个性化调整。详细操作方式如下：

1. 主账号（或具备 CAM 管理权限的用户）在 策略 页面，单击**新建自定义策略**。



2. 在弹出的窗口中，单击**按策略语法创建**。
3. 在选择策略模板页面，选择**空白模板**，单击**下一步**。
4. 在编辑策略页面，设置策略名称，输入策略内容，策略内容可从 CLS 访问策略模板 中进行复制。例如：需要给子账号赋予使用Loglistener采集数据的权限，拷贝如下图所示的策略：

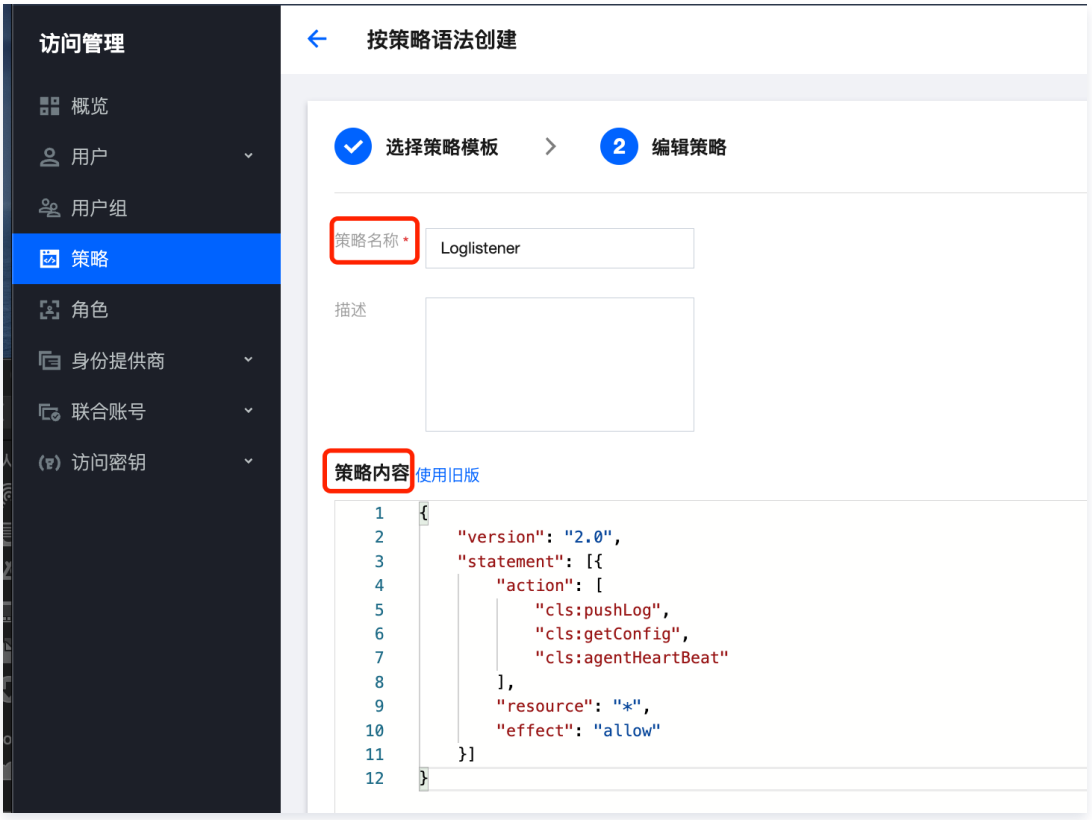
版权所有：腾讯云计算（北京）有限责任公司

数据采集相关

使用 Loglistener 采集数据

用户可以使用采集 Agent Loglistener 采集数据，且具备日志上传的能力（本示例展示机器安装 Loglistener 上传日志的最小权限）。

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:pushLog",
            "cls:getConfig",
            "cls:agentHeartBeat"
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```



5. 单击**完成**，保存该策略。创建完成自定义策略后，可通过 授权管理 将策略关联至用户/用户组/角色，使用用户/用户组/角色获得对应的操作权限。

可授权的资源类型

最近更新时间：2025-06-26 18:56:51

简介

日志服务（Cloud Log Service，CLS）包含多种资源类型，部分接口支持按资源为用户配置权限，例如 对指定日志主题具备管理权限 。
在访问管理（Cloud Access Management，CAM）中可授权的资源类型如下表，其中“按标签授权”是指该类型资源是否支持通过 标签 的方式指定用户具备操作权限的资源范围。

资源类型	授权策略中的资源描述方法	按标签授权
日志集	`qcs::cls:$region:$account:logset/*` `qcs::cls:$region:$account:logset/$logsetId`	支持
日志主题	`qcs::cls:$region:$account:topic/*` `qcs::cls:$region:$account:topic/$topicId`	支持
机器组	`qcs::cvm:$region:$account:machinegroup/*` `qcs::cvm:$region:$account:machinegroup/$machinegroupId`	支持
采集配置	`qcs::cls:$region:$account:config/*` `qcs::cls:$region:$account:config/$configId`	不支持
仪表盘（无地域资源）	`qcs::cls::$account:dashboard/*` `qcs::cls::$account:dashboard/$dashboardId`	支持
告警策略	`qcs::cls:$region:$account:alarm/*` `qcs::cls:$region:$account:alarm/$alarmId`	支持
通知渠道组	`qcs::cls:$region:$account:alarmNotice/*` `qcs::cls:$region:$account:alarmNotice/$alarmNoticeId`	支持
数据加工任务	`qcs::cls:$region:uin/$account:datatransform/*` `qcs::cls:$region:uin/$account:datatransform/$TaskId`	不支持
投递任务（COS）	`qcs::cls:$region:$account:shipper/*` `qcs::cls:$region:$account:shipper/$shipperId`	不支持
DataSight 独立控制台（无地域资源）	`qcs::cls::$account:datasight/*` `qcs::cls::$account:datasight/datasightId`	支持

其他资源类型（已废弃，仅老版本 API 使用）	仪表盘内的单个图表：
	`qcs::cls:$region:$account:chart/*`
	`qcs::cls:$region:$account:chart/$chartId`
		不支持

其中，`$region` 、`$account` 等变量参数需修改为您实际的参数信息，仪表盘为无地域资源，不需要填写region参数。
日志服务支持的所有接口及其对应的资源描述方法，参见 支持 CAM 的业务接口 > 存储 > 存储数据服务 > 日志服务 。其中，“授权粒度”为“资源级”的接口支持以上述资源类型按资源方式为用户配置权限，“授权粒度”为“操作级”的接口在 CAM 权限策略中对应的资源范围必须为 `*` 。

实践建议

由于日志服务中不同类型的资源之间具备关联关系，例如日志集包含日志主题，日志主题需应用采集配置至机器组；如果直接按照资源 ID 的方式在 CAM 权限策略中为用户配置权限存在较大的管理难度，容易导致用户在部分接口下出现无权限错误。因此，建议按照如下方式配置 CAM 权限策略：

- 针对支持按标签授权的资源类型及对应接口，为相关的资源绑定标签，然后通过标签的方式指定用户具备操作权限的资源范围。例如同时为日志主题、日志集及相关的仪表盘绑定标签，然后 对指定标签的日志主题赋予管理权限（含日志集），对指定标签的仪表盘赋予管理权限 。这两条策略可以使用户同时具备这三类资源相关接口的操作权限。
- 针对不支持按标签授权的资源类型及对应接口，为简化管理，可直接以 `*` 作为 CAM 权限策略中的资源范围，即所有资源。为避免普通用户误操作，可针对普通用户配置只读权限，针对管理用户配置管理权限，例如 管理权限：对所有数据加工任务具备管理权限 和 只读权限：对所有数据加工任务具备只读权限 。

> **说明：**
> 更多使用场景，请参见 CLS 访问策略模板 。

CLS 访问策略模板

最近更新时间：2025-06-11 18:56:42

使用自定义权限策略时，可按照使用场景采用如下的权限策略模板：

模块	使用场景
整体（最佳实践）	使用标签对主题、机器组、仪表盘进行分类，并按标签配置权限 • 对指定标签的资源具备管理权限 • 对指定标签的资源具备只读权限
数据采集	• 使用 LogListener 采集服务器数据 • 指标上报 • 使用 LogListener 采集自建 K8s 数据 • 使用 API/SDK 上传数据 • 使用 Kafka 上传数据 • 使用云产品指标订阅上传数据 • 订阅 MySQL Binlog 数据 • 使用 Kafka 订阅采集数据 • FluentBit 日志上传 • Logstash 日志上传 • 管理采集配置及机器组
主题管理及检索分析	使用控制台查看/管理主题及检索分析 • 管理权限：对所有主题具备管理权限 • 管理权限：对指定主题具备管理权限 • 管理权限：对指定标签的主题具备管理权限 • 只读权限：对所有主题具备只读权限 • 只读权限：对指定主题具备只读权限 • 只读权限：对指定标签的主题具备只读权限 使用 API 检索分析 • 只读权限：对所有主题具备检索分析只读权限 • 只读权限：对指定主题具备检索分析只读权限 • 只读权限：对指定标签的主题具备检索分析只读权限
仪表盘	• 管理权限：对所有仪表盘具备管理权限 • 管理权限：对指定标签的仪表盘具备管理权限 • 管理权限：对指定资源的仪表盘具备管理权限 • 只读权限：对所有仪表盘具备只读权限 • 只读权限：对指定标签的仪表盘具备只读权限 • 只读权限：对指定资源的仪表盘具备只读权限
监控告警	• 管理权限：对所有告警策略具备管理权限 • 管理权限：对指定标签的告警策略具备管理权限 • 管理权限：对指定资源的告警策略具备管理权限 • 只读权限：对所有告警策略具备只读权限 • 只读权限：对指定标签的告警策略具备只读权限 • 只读权限：对指定资源的告警策略具备只读权限
数据处理	数据加工 • 管理权限：对所有数据加工任务具备管理权限 • 只读权限：对所有数据加工任务具备只读权限 定时 SQL 分析相关 • 管理权限：对所有日志主题具备定时 SQL 分析的权限 • 管理权限：对指定标签日志主题具备定时 SQL 分析的权限

数据投递/消费	投递 CKafka • 管理权限：对所有日志主题具备投递 CKafka 管理权限 • 管理权限：对指定标签日志主题具备投递 CKafka 管理权限 • 只读权限：对所有日志主题具备投递 CKafka 只读权限 • 只读权限：对指定标签日志主题具备投递 CKafka 只读权限 投递 COS • 管理权限：对所有日志主题具备投递 COS 管理权限 • 管理权限：对指定标签日志主题具备投递 COS 管理权限 • 只读权限：对所有日志主题具备投递 COS 只读权限 • 只读权限：对指定标签日志主题具备投递 COS 只读权限 投递 DLC • 管理权限：对所有日志主题具备投递 DLC 管理权限 • 管理权限：对指定标签日志主题具备投递 DLC 管理权限 • 只读权限：对所有日志主题具备投递 DLC 只读权限 • 只读权限：对指定标签日志主题具备投递 DLC 只读权限 投递 SCF • 管理权限：对所有日志主题具备投递 SCF 管理权限 • 管理权限：对指定标签日志主题具备投递 SCF 管理权限 • 只读权限：对所有日志主题具备投递 SCF 只读权限 • 只读权限：对指定标签日志主题具备投递 SCF 只读权限 Kafka 协议消费 • 管理权限：对所有日志主题具备 Kafka 协议消费权限 • 管理权限：对指定标签日志主题具备 Kafka 协议消费权限 • 管理权限：对指定资源具备 Kafka 协议消费权限 • Kafka 协议消费权限最小权限（非控制台，调用 API） 指标投递 • 管理权限：对所有指标主题具备投递管理权限 • 管理权限：对指定标签的指标主题具备投递管理权限 自定义消费 • 管理权限：对所有日志主题具备自定义消费管理权限
DataSight 独立控制台	管理 DataSight • 管理权限：对所有 DataSight 独立控制台具备管理权限 • 管理权限：对指定 DataSight 独立控制台具备管理权限 • 管理权限：对指定标签 DataSight 独立控制台具备管理权限 • 只读权限：对所有 DataSight 独立控制台具备只读权限 • 只读权限：对指定 DataSight 独立控制台具备只读权限 • 只读权限：对指定标签 DataSight 独立控制台具备只读权限
开发者	通过 Grafana 使用 CLS • 通过 Grafana 展示所有主题的数据 • 通过 Grafana 展示具备指定标签的主题的数据

整体（最佳实践）

使用标签对主题、机器组、仪表盘进行分类，并按标签配置权限。创建资源时需为资源指定标签，用户仅对具备指定标签的资源具备管理或只读权限，通过这种方式能够方便的批量管理日志服务中多种类型的资源。

对指定标签的资源具备管理权限

> ⚠ **注意：**
> 使用这个策略时请清空其中的注释说明。

```
{
    "statement": [{
            "action": [ //必要的相关产品只读权限
```

```
                "monitor:GetMonitorData",
                "monitor:DescribeBaseMetrics",
                "cam:ListGroups",
                "cam:GetGroup",
                "cam:DescribeSubAccountContacts",
                "cam:ListAttachedRolePolicies",
                "cam:GetRole",
                "vpc:DescribeSubnetEx",//创建内网访问的DataSight时需要
                "vpc:DescribeVpcEx",//创建内网访问的DataSight时需要
                "tag:TagResources",
                "tag:DescribeResourceTagsByResourceIds",
                "tag:GetTags",
                "tag:GetTagKeys",
                "tag:GetTagValues",
                "kms:GetServiceStatus"
            ],
            "effect": "allow",
            "resource": "*"
        },
        {
            "action": [ //限制用户创建仪表盘、日志集、主题、告警策略、通知渠道组、机器组和DataSight时，必须绑定指定的标签，例如testCAM:test1。创建其它类型资源暂不支持限定标签。
                "cls:CreateDashboard",
                "cls:CreateLogset",
                "cls:CreateTopic",
                "cls:CreateAlarm",
                "cls:CreateAlarmNotice",
                "cls:CreateMachineGroup",
                "cls:CreateConsole"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:request_tag": [
                        "testCAM&test1"
                    ]
                }
            },
            "effect": "allow",
            "resource": "*"
        },
        {
            "action": [ //资源具备指定标签时，用户即拥有所有相关接口的权限（需接口支持按标签限制权限）。
                "cls:*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "testCAM&test1"
                    ]
                }
            },
            "effect": "allow",
            "resource": "*"
        },
        {
            "action": [ //部分接口不支持按标签限制权限，不能限制资源范围。以下接口主要为读操作，另有少部分辅助功能接口为写操作，均不影响产品核心数据安全性。
                "cls:CheckAlarmChannel",
                "cls:CheckAlarmRule",
                "cls:CheckDomainRepeat",
                "cls:CheckFunction",
```

```
                "cls:CheckRechargeKafkaServer",
                "cls:DescribeClsPrePayDetails",
                "cls:DescribeClsPrePayInfos",
                "cls:DescribeConfigMachineGroups",
                "cls:DescribeConfigs",
                "cls:DescribeAgentConfigs",
                "cls:DescribeTopicExtendConfig",
                "cls:DescribeDataTransformFailLogInfo",
                "cls:DescribeDataTransformInfo",
                "cls:DescribeDataTransformPreviewDataInfo",
                "cls:DescribeDataTransformPreviewInfo",
                "cls:DescribeDataTransformProcessInfo",
                "cls:DescribeDemonstrations",
                "cls:DescribeExceptionResources",
                "cls:DescribeExternalDataSourcePreview",
                "cls:DescribeFunctions",
                "cls:DescribeResources",
                "cls:DescribeShipperPreview",
                "cls:DescribeScheduledSqlProcessInfo",
                "cls:DescribeConfigurationTemplates",
                "cls:DescribeFolders",
                "cls:GetClsService",
                "cls:GetConfigurationTemplateApplyLog",
                "cls:PreviewKafkaRecharge",
                "cls:agentHeartBeat",
                "cls:CreateDemonstrations",
                "cls:DeleteDemonstrations",
                "cls:DescribeNoticeContents",
                "cls:DescribeWebCallbacks"
            ],
            "effect": "allow",
            "resource": "*"
        },
        {
            "action": [ //部分接口不支持按标签限制权限，不能限制资源范围。以下接口涉及核心功能的写操作，建议仅按需向少部分用户授权，可删除不需要授权的接口。
                "cls:RealtimeProducer", //使用 Kafka 上传数据
                "cls:CreateConfigurationTemplate", //配置模版相关功能接口
                "cls:ModifyConfigurationTemplate",
                "cls:DeleteConfigurationTemplate",
                "cls:CreateFolder",//文件夹相关功能接口
                "cls:ModifyFolder",
                "cls:DeleteFolder",
                "cls:ModifyResourceAndFolderRelation",
                "cls:CreateDataTransform",//数据加工相关功能接口
                "cls:ModifyDataTransform",
                "cls:DeleteDataTransform",
                "cls:RetryShipperTask",//投递COS相关功能接口
                "cls:ModifyDashboardSubscribeAck",//仪表盘订阅相关功能接口
                "cls:DeleteDashboardSubscribe",
                "cls:ModifyConfigExtra",//采集配置相关功能接口
                "cls:DeleteConfigExtra",
                "cls:RemoveMachine",//机器组相关功能接口
                "cls:UpgradeAgentNormal",
                "cls:CreateNoticeContent",//告警通知内容模版相关功能接口
                "cls:DeleteNoticeContent",
                "cls:ModifyNoticeContent",
                "cls:CreateWebCallback",//告警集成配置相关功能接口
                "cls:ModifyWebCallback",
                "cls:DeleteWebCallback"
            ],
```

```
            "effect": "allow",
            "resource": "*"
        }
    ],
    "version": "2.0"
}
```

对指定标签的资源具备只读权限

> **注意：**
> 使用这个策略时请清空其中的注释说明。

```
{
    "statement": [{
            "action": [ //必要的相关产品只读权限
                "monitor:GetMonitorData",
                "monitor:DescribeBaseMetrics",
                "cam:ListGroups",
                "cam:GetGroup",
                "cam:DescribeSubAccountContacts",
                "cam:ListAttachedRolePolicies",
                "tag:DescribeResourceTagsByResourceIds",
                "tag:GetTags",
                "tag:GetTagKeys",
                "tag:GetTagValues"
            ],
            "effect": "allow",
            "resource": "*"
        },
        {
            "action": [ //资源具备指定标签时，用户即拥有相关只读接口的权限
                "cls:DescribeConsumer",
                "cls:DescribeConsumerPreview",
                "cls:DescribeCosRecharges",
                "cls:DescribeDashboardSubscribes",
                "cls:DescribeDashboards",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribeKafkaConsume",
                "cls:DescribeKafkaConsumer",
                "cls:DescribeKafkaRecharges",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeLatestUserLog",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLogHistogram",
                "cls:DescribeMachineGroupConfigs",
                "cls:DescribeMachines",
                "cls:DescribePartitions",
                "cls:DescribeScheduledSqlInfo",
                "cls:DescribeScheduledSqlProcessInfo",
                "cls:DescribeShipperPreview",
                "cls:DescribeTopics",
                "cls:EstimateRebuildIndexTask",
                "cls:GetAlarm",
                "cls:GetAlarmLog",
                "cls:GetMetricLabelValues",
                "cls:GetMetricSeries",
```

```
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
                "cls:MetricsQuery",
                "cls:MetricsQueryExemplars",
                "cls:MetricsQueryRange",
                "cls:MetricsSeries",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:SearchCosRechargeInfo",
                "cls:SearchDashboardSubscribe",
                "cls:SearchLog",
                "cls:DescribeAlarmNotices",
                "cls:DescribeAlarms",
                "cls:DescribeAlertRecordHistory",
                "cls:DescribeExternalDataSources",
                "cls:DescribeLogsets",
                "cls:DescribeMachineGroups",
                "cls:DescribeConsoles",
                "cls:DescribeShipperTasks",
                "cls:DescribeShippers",
                "cls:DescribeRebuildIndexTasks"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "testCAM&test1"
                    ]
                }
            },
            "effect": "allow",
            "resource": "*"
        },
        {
            "action": [ //部分接口不支持按标签限制权限，不能限制资源范围。以下接口主要为读操作，另有少部分辅助功能接口为写操作，均不影响产品核心数据安全性。
                "cls:CheckAlarmChannel",
                "cls:CheckAlarmRule",
                "cls:CheckDomainRepeat",
                "cls:CheckFunction",
                "cls:CheckRechargeKafkaServer",
                "cls:DescribeClsPrePayDetails",
                "cls:DescribeClsPrePayInfos",
                "cls:DescribeConfigMachineGroups",
                "cls:DescribeConfigs",
                "cls:DescribeAgentConfigs",
                "cls:DescribeTopicExtendConfig",
                "cls:DescribeDataTransformFailLogInfo",
                "cls:DescribeDataTransformInfo",
                "cls:DescribeDataTransformPreviewDataInfo",
                "cls:DescribeDataTransformPreviewInfo",
                "cls:DescribeDataTransformProcessInfo",
                "cls:DescribeDemonstrations",
                "cls:DescribeExceptionResources",
                "cls:DescribeExternalDataSourcePreview",
                "cls:DescribeFunctions",
                "cls:DescribeResources",
                "cls:DescribeShipperPreview",
                "cls:DescribeScheduledSqlProcessInfo",
                "cls:DescribeConfigurationTemplates",
                "cls:DescribeFolders",
                "cls:GetClsService",
```

```
                "cls:GetConfigurationTemplateApplyLog",
                "cls:PreviewKafkaRecharge",
                "cls:CreateDemonstrations",
                "cls:DeleteDemonstrations",
                "cls:CreateExport",
                "cls:DeleteExport"
                "cls:DescribeNoticeContents",
                "cls:DescribeWebCallbacks"
            ],
            "effect": "allow",
            "resource": "*"
        }
    ],
    "version": "2.0"
}
```

数据采集相关

使用 Loglistener 采集数据

用户可以使用 Agent Loglistener 采集数据，且具备日志上传的能力（本示例展示机器安装 Loglistener 上传日志的最小权限）。

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:pushLog",
            "cls:getConfig",
            "cls:agentHeartBeat"
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

> **说明**
> 如果您使用的 Loglistener 为2.6.5以前的版本，则需要加上 "cls:listLogset" 权限。

指标上报

用户可使用 Prometheus Remote Write 协议上报指标数据，也可使用兼容该协议的各类采集器(如 vmagent 及 telegraf)采集指标并上报至指标主题。（本示例展示指标上报的最小权限）。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:MetricsRemoteWrite"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

使用自建 k8s 上传数据

用户可以使用 Logagent 采集自建 k8s 环境的日志数据，且具备上传的能力（本示例展示自建 k8s 上传日志的最小权限）。

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "cls:pushLog",
        "cls:agentHeartBeat",
        "cls:getConfig",
        "cls:CreateConfig",
        "cls:DeleteConfig",
        "cls:ModifyConfig",
        "cls:DescribeConfigs",
        "cls:DescribeMachineGroupConfigs",
        "cls:DeleteConfigFromMachineGroup",
        "cls:ApplyConfigToMachineGroup",
        "cls:DescribeConfigMachineGroups",
        "cls:ModifyTopic",
        "cls:DeleteTopic",
        "cls:CreateTopic",
        "cls:DescribeTopics",
        "cls:CreateLogset",
        "cls:DeleteLogset",
        "cls:DescribeLogsets",
        "cls:CreateIndex",
        "cls:ModifyIndex",
        "cls:CreateMachineGroup",
        "cls:DeleteMachineGroup",
        "cls:DescribeMachineGroups",
        "cls:ModifyMachineGroup",
        "cls:CreateConfigExtra",
        "cls:DeleteConfigExtra",
        "cls:ModifyConfigExtra"
      ],
      "resource": "*",
      "effect": "allow"
    }
  ]
}
```

使用 API/SDK 上传数据

用户可以通过 API/SDK 上传数据到 CLS（本示例展示使用 API/SDK 上传数据的最小权限）。

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:pushLog",
            "cls:UploadLog",
            "cls:MetricsRemoteWrite"
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

使用 Kafka 上传数据

用户可以通过 Kafka 协议上传日志到 CLS（本示例展示使用 Kafka 协议上传日志的最小权限）。

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:RealtimeProducer"
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

使用云产品指标订阅上传数据

用户可以通过云产品指标订阅上传指标到 CLS（本示例展示控制台配置云产品指标订阅时所需要的最小权限）。

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:CreateMetricSubscribe",
            "cls:DescribeMetricCorrectDimension",
            "cls:DescribeMetricSubscribePreview",
            "monitor:DescribeBaseMetrics",
            "monitor:DescribeProductList"
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

订阅 MySQL Binlog 日志

用户可以订阅 MySQL Binlog 日志至 CLS（本示例展示控制台配置 MySQL Binlog 订阅任务时所需要的最小权限）。

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:CreateBinlogSubscribe",
            "cls:DescribeBinlogSubscribes",
            "cls:ModifyBinlogSubscribe",
            "cls:DescribeBinlogSubscribeConnectivity",
            "cls:DescribeBinlogSubscribePreview",
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

使用 Kafka 订阅采集数据

用户可以订阅 Kafka 集群中的数据至 CLS（本示例展示控制台配置 Kafka 订阅任务时所需要的最小权限）。

```
{
    "version": "2.0",
```

```
    "statement": [{
        "action": [
            "cls:PreviewKafkaRecharge",
            "cls:CreateKafkaRecharge",
            "cls:ModifyKafkaRecharge",
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

FluentBit 日志上传

用户可以通过 Fluent-bit Go 插件将 FluentBit 中的数据上传到 CLS（本示例展示使用 Fluent-bit Go 插件上传数据的最小权限）。

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:pushLog",
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

Logstash 日志上传

用户可以通过 Logstash 插件将 Logstash 中的数据上传到 CLS（本示例展示使用 Logstash 插件上传数据的最小权限）。

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:pushLog",
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

管理采集配置及机器组

包括创建/修改/删除采集配置及创建/修改/删除机器组。

- Config 相关接口对应采集配置相关资源。
- MachineGroup 相关接口对应机器组相关资源。
- ConfigExtra 相关的三个接口权限用于管理自建 k8s 上传日志相关的集群配置信息，如不使用自建 k8s 上传日志相关功能可以忽略。

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:DescribeLogsets",
            "cls:DescribeTopics",
            "cls:CreateConfig",
            "cls:CreateConfig",
            "cls:DeleteConfig",
            "cls:DescribeConfigs",
            "cls:ModifyConfig",
```

```
            "cls:CreateConfigExtra",
            "cls:DeleteConfigExtra",
            "cls:ModifyConfigExtra",
            "cls:CreateMachineGroup",
            "cls:DeleteMachineGroup",
            "cls:DescribeMachineGroups",
            "cls:DeleteConfigFromMachineGroup",
            "cls:ApplyConfigToMachineGroup",
            "cls:ModifyMachineGroup"
        ],
        "resource": "*",
        "effect": "allow"
    }
    ]
}
```

主题管理及检索分析相关

使用控制台查看/管理主题及检索分析

管理权限：对所有主题具备管理权限

用户可以对所有的主题进行检索及管理，包括创建主题、删除主题和修改索引配置等，不包括采集配置、日志投递和日志加工等。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateLogset",
                "cls:CreateTopic",
                "cls:CreateExport",
                "cls:CreateIndex",
                "cls:DeleteLogset",
                "cls:DeleteTopic",
                "cls:DeleteExport",
                "cls:DeleteIndex",
                "cls:ModifyLogset",
                "cls:ModifyTopic",
                "cls:ModifyIndex",
                "cls:MergePartition",
                "cls:SplitPartition",
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                "cls:CreateRebuildIndexTask",
                "cls:EstimateRebuildIndexTask",
                "cls:CancelRebuildIndexTask",
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
```

```
                "cls:MetricsQuery",
                "cls:MetricsQueryRange",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

管理权限：对指定主题具备管理权限

用户能够对指定的主题进行检索及管理，包括修改主题、删除主题和修改索引配置等，不包括创建主题、采集配置、日志投递和日志加工等。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateExport",
                "cls:CreateIndex",
                "cls:DeleteLogset",
                "cls:DeleteTopic",
                "cls:DeleteExport",
                "cls:DeleteIndex",
                "cls:ModifyLogset",
                "cls:ModifyTopic",
                "cls:ModifyIndex",
                "cls:MergePartition",
                "cls:SplitPartition",
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                "cls:CreateRebuildIndexTask",
                "cls:EstimateRebuildIndexTask",
                "cls:CancelRebuildIndexTask",
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
                "cls:MetricsQuery",
                "cls:MetricsQueryRange",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
```

```
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "qcs::cls:ap-guangzhou:100007***827:logset/1c012db7-2cfd-4418-****-7342c7a42516",
                "qcs::cls:ap-guangzhou:100007***827:topic/380fe1f1-0c7b-4b0d-****-d514959db1bb"
            ]
        }
    ]
}
```

管理权限：对指定标签的主题具备管理权限

用户可以对包含指定标签的主题进行检索及管理，包括修改主题、删除主题和修改索引配置等，不包括创建主题、采集配置、日志投递和日志加工等。为主题绑定标签时，需同时为其所属的日志集绑定标签。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateExport",
                "cls:CreateIndex",
                "cls:DeleteLogset",
                "cls:DeleteTopic",
                "cls:DeleteExport",
                "cls:DeleteIndex",
                "cls:ModifyLogset",
                "cls:ModifyTopic",
                "cls:ModifyIndex",
                "cls:MergePartition",
                "cls:SplitPartition",
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                "cls:CreateRebuildIndexTask",
                "cls:EstimateRebuildIndexTask",
                "cls:CancelRebuildIndexTask",
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
                "cls:MetricsQuery",
                "cls:MetricsQueryRange",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
```

```
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "testCAM&test1"
                    ]
                }
            }
        }
    ]
}
```

只读权限：对所有主题具备只读权限

用户可以对所有的主题进行检索。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
                "cls:MetricsQuery",
                "cls:MetricsQueryRange",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

只读权限：对指定主题具备只读权限

用户可以对指定的主题进行检索。

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

```
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
                "cls:MetricsQuery",
                "cls:MetricsQueryRange",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "qcs::cls:ap-guangzhou:100007***827:logset/1c012db7-2cfd-4418-****-7342c7a42516",
                "qcs::cls:ap-guangzhou:100007***827:topic/380fe1f1-0c7b-4b0d-****-d514959db1bb"
            ]
        }
    ]
}
```

只读权限：对指定标签的主题具备只读权限

用户可以对包含指定标签的主题进行检索。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
                "cls:MetricsQuery",
                "cls:MetricsQueryRange",
```

```
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "testCAM&test1"
                    ]
                }
            }
        }
    ]
}
```

使用 API 检索分析

只读权限：对所有主题具备检索分析只读权限

用户可以通过 API 对所有的主题进行检索分析。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
                "cls:MetricsQuery",
                "cls:MetricsQueryRange",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries",
                "cls:MetricsRemoteRead"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

只读权限：对指定主题具备检索分析只读权限

用户可以通过 API 对指定的主题进行检索分析。

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

```
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
                "cls:MetricsQuery",
                "cls:MetricsQueryRange",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries",
                "cls:MetricsRemoteRead"
            ],
            "resource": [
                "qcs::cls:ap-guangzhou:100007***827:logset/1c012db7-2cfd-4418-****-7342c7a42516",
                "qcs::cls:ap-guangzhou:100007***827:topic/380fe1f1-0c7b-4b0d-****-d514959db1bb"
            ]
        }
    ]
}
```

只读权限：对指定标签的主题具备检索分析只读权限

用户可以通过 API 对包含指定标签的主题进行检索分析。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:MetricsLabelValues",
                "cls:MetricsLabels",
                "cls:MetricsQuery",
                "cls:MetricsQueryRange",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries",
                "cls:MetricsRemoteRead"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "testCAM&test1"
                    ]
                }
            }
        }
    ]
```

```
}
```

仪表盘相关

管理权限：对所有仪表盘具备管理权限

用户可以管理**所有**的仪表盘，包括创建、删除、编辑、查看、订阅所有仪表盘。仪表盘可以使用**所有**主题的数据。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:CreateChart",
                "cls:CreateDashboard",
                "cls:DeleteChart",
                "cls:DeleteDashboard",
                "cls:ModifyChart",
                "cls:ModifyDashboard",
                "cls:DescribeDashboards",
                "cls:CreateFolder",
                "cls:DeleteFolder",
                "cls:DescribeFolders",
                "cls:ModifyFolder",
                "cls:ModifyResourceAndFolderRelation",
                "cls:SearchDashboardSubscribe",
                "cls:CreateDashboardSubscribe",
                "cls:ModifyDashboardSubscribe",
                "cls:DescribeDashboardSubscribes",
                "cls:DeleteDashboardSubscribe",
                "cls:ModifyDashboardSubscribeAck"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": "*"
        }
    ]
}
```

管理权限：对指定标签的仪表盘具备管理权限

用户可以管理**指定标签**的仪表盘，包括创建、删除、编辑、查看、订阅携带指定标签的仪表盘。仪表盘可以使用**指定标签**主题的数据。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:CreateChart",
                "cls:CreateDashboard",
                "cls:DeleteChart",
                "cls:DeleteDashboard",
                "cls:ModifyChart",
                "cls:ModifyDashboard",
                "cls:DescribeDashboards",
                "cls:CreateFolder",
                "cls:DeleteFolder",
                "cls:DescribeFolders",
                "cls:ModifyFolder",
                "cls:ModifyResourceAndFolderRelation",
                "cls:SearchDashboardSubscribe",
                "cls:CreateDashboardSubscribe",
                "cls:ModifyDashboardSubscribe",
                "cls:DescribeDashboardSubscribes",
                "cls:DeleteDashboardSubscribe",
                "cls:ModifyDashboardSubscribeAck"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
```

```
    }
```

管理权限：对指定资源的仪表盘具备管理权限

用户可以管理指定仪表盘，包括创建、删除、编辑、查看、订阅指定的仪表盘资源。仪表盘可以使用指定主题的数据。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:CreateChart",
                "cls:CreateDashboard",
                "cls:DeleteChart",
                "cls:DeleteDashboard",
                "cls:ModifyChart",
                "cls:ModifyDashboard",
                "cls:DescribeDashboards",
                "cls:CreateFolder",
                "cls:DeleteFolder",
                "cls:DescribeFolders",
                "cls:ModifyFolder",
                "cls:ModifyResourceAndFolderRelation",
                "cls:SearchDashboardSubscribe",
                "cls:CreateDashboardSubscribe",
                "cls:ModifyDashboardSubscribe",
                "cls:DescribeDashboardSubscribes",
                "cls:DeleteDashboardSubscribe",
                "cls:ModifyDashboardSubscribeAck"
            ],
            "resource": [
                "qcs::cls::uin/100000***001:dashboard/dashboard-0769a3ba-2514-409d-****-f65b20b23736"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "qcs::cls::uin/100000***001:topic/174ca473-50d0-4fdf-****-2ef681a1e02a"
            ]
        }
    ]
}
```

只读权限：对所有仪表盘具备只读权限

用户可以查看**所有**的仪表盘。仪表盘可以查看**所有**主题的数据。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:DescribeDashboards",
                "cls:DescribeFolders",
                "cls:SearchDashboardSubscribe",
                "cls:DescribeDashboardSubscribes"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对指定标签的仪表盘具备只读权限

用户可以查看携带**指定标签**的仪表盘资源。仪表盘可以查看携带**指定标签**的主题的数据。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:DescribeDashboards",
                "cls:DescribeFolders",
                "cls:SearchDashboardSubscribe",
                "cls:DescribeDashboardSubscribes"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
```

```
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

只读权限：对指定资源的仪表盘具备只读权限

用户可以查看**指定仪表盘**。仪表盘可以查看**指定主题**的数据。

```
{
    "version": "2.0",
    "statement": [
         {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:DescribeDashboards",
                "cls:DescribeFolders",
                "cls:SearchDashboardSubscribe",
                "cls:DescribeDashboardSubscribes"
            ],
            "resource": [
                "qcs::cls::uin/100000***001:dashboard/dashboard-0769a3ba-2514-409d-****-f65b20b23736"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
```

```
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "qcs::cls::uin/100000***001:topic/174ca473-50d0-4fdf-****-2ef681a1e02a"
            ]
        }
    ]
}
```

监控告警相关

管理权限：对所有告警策略具备管理权限

用户可以对所有告警策略进行管理，包括创建告警策略、创建通知渠道组和查看告警策略等。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:SearchLog",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:CreateAlarm",
                "cls:ModifyAlarm",
                "cls:DeleteAlarm",
                "cls:DescribeAlarmNotices",
                "cls:CreateAlarmNotice",
                "cls:ModifyAlarmNotice",
                "cls:DeleteAlarmNotice",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory",
                "cls:CheckAlarmRule",
                "cls:CheckAlarmChannel"
            ],
            "resource": "*"
        }
    ]
}
```

管理权限：对指定标签的告警策略具备管理权限

用户可以对包含执行标签的告警策略进行管理，包括修改告警策略、修改通知渠道组和查看告警策略等。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:SearchLog",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup",
                "cls:CheckAlarmRule",
                "cls:CheckAlarmChannel",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:ModifyAlarm",
                "cls:DeleteAlarm",
                "cls:DescribeAlarmNotices",
                "cls:ModifyAlarmNotice",
                "cls:DeleteAlarmNotice",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

管理权限：对指定资源的告警策略具备管理权限

用户可以对指定的告警策略进行管理，包括修改告警策略、修改通知渠道组和查看告警策略等。

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

```
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:SearchLog",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup",
                "cls:CheckAlarmRule",
                "cls:CheckAlarmChannel",
                "cls:GetMetricLabelValues",
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:ModifyAlarm",
                "cls:DeleteAlarm",
                "cls:DescribeAlarmNotices",
                "cls:ModifyAlarmNotice",
                "cls:DeleteAlarmNotice",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory"
            ],
            "resource": [
                "qcs::cls:ap-guangzhou:100007***827:alarm/alarm-xxx-9bbe-4625-ac29-b5e66bf643cf",
                "qcs::cls:ap-guangzhou:100007***827:alarmNotice/notice-xxx-ec2c-410f-924f-4ee8a7cd028e"
            ]
        }
    ]
}
```

只读权限：对所有告警策略具备只读权限

用户可以查看所有告警策略。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:DescribeAlarmNotices",
```

```
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对指定标签的告警策略具备只读权限

用户可以查看包含指定标签的告警策略。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:DescribeAlarmNotices",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

只读权限：对指定资源的告警策略具备只读权限

用户可以对指定的告警策略进行查看。

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

```
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:DescribeAlarmNotices",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory"
            ],
            "resource": [
                "qcs::cls:ap-guangzhou:100007***827:alarm/alarm-xxx-9bbe-4625-ac29-b5e66bf643cf",
                "qcs::cls:ap-guangzhou:100007***827:alarmNotice/notice-xxx-ec2c-410f-924f-4ee8a7cd028e"
            ]
        }
    ]
}
```

数据处理

数据加工相关

管理权限：对所有数据加工任务具备管理权限

所有日志主题的“数据加工任务”的管理权限。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeDataTransformPreviewDataInfo",
                "cls:DescribeTopics",
                "cls:DescribeIndex",
                "cls:CreateDataTransform"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeFunctions",
                "cls:CheckFunction",
                "cls:DescribeDataTransformFailLogInfo",
                "cls:DescribeDataTransformInfo",
```

```
                "cls:DescribeDataTransformPreviewInfo",
                "cls:DescribeDataTransformProcessInfo",
                "cls:DeleteDataTransform",
                "cls:ModifyDataTransform"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

只读权限：对所有数据加工任务具备只读权限

所有日志主题的“数据加工任务”的只读权限。由于仅是查看，所以不需要对 DSL 函数进行授权。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeDataTransformFailLogInfo",
                "cls:DescribeDataTransformInfo",
                "cls:DescribeDataTransformPreviewDataInfo",
                "cls:DescribeDataTransformPreviewInfo",
                "cls:DescribeDataTransformProcessInfo"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

定时 SQL 分析相关

管理权限：对所有日志主题具备定时 SQL 分析的权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:CreateScheduledSql",
                "cls:SearchLog",
                "cls:DescribeScheduledSqlInfo",
```

```
                "cls:DescribeScheduledSqlProcessInfo",
                "cls:DeleteScheduledSql",
                "cls:ModifyScheduledSql",
                "cls:RetryScheduledSqlTask"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

管理权限：对指定标签日志主题具备定时 SQL 分析的权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:SearchLog",
                "cls:DescribeScheduledSqlProcessInfo",
                "cls:CreateScheduledSql",
                "cls:DeleteScheduledSql",
                "cls:ModifyScheduledSql",
                "cls:RetryScheduledSqlTask"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:DescribeScheduledSqlInfo"
```

```
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

数据投递/消费相关

投递 CKafka

管理权限：对所有日志主题具备投递 CKafka 管理权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:CreateConsumer",
                "cls:ModifyConsumer",
                "cls:DeleteConsumer",
                "cls:DescribeConsumer",
                "cls:DescribeConsumerPreview"

            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList",
                "ckafka:DescribeInstances",
                "ckafka:DescribeTopic",
                "ckafka:DescribeInstanceAttributes",
                "ckafka:CreateToken",
                "ckafka:AuthorizeToken"
            ],
            "resource": "*"
        }
    ]
}
```

管理权限：对指定标签日志主题具备投递 CKafka 管理权限

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

```
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:CreateConsumer",
                "cls:ModifyConsumer",
                "cls:DeleteConsumer",
                "cls:DescribeConsumer",
                "cls:DescribeConsumerPreview"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "age&13",
                        "name&vinson"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList",
                "ckafka:DescribeInstances",
                "ckafka:DescribeTopic",
                "ckafka:DescribeInstanceAttributes",
                "ckafka:CreateToken",
                "ckafka:AuthorizeToken"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对所有日志主题具备投递 CKafka 只读权限

具备所有日志主题投递 CKafka 的只读权限。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:DescribeConsumer",
                "cls:DescribeConsumerPreview"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
```

```
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "ckafka:DescribeInstances",
                "ckafka:DescribeTopic",
                "ckafka:DescribeInstanceAttributes",
                "ckafka:CreateToken",
                "ckafka:AuthorizeToken"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对指定标签日志主题具备投递 CKafka 只读权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:DescribeConsumer",
                "cls:DescribeConsumerPreview"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "ckafka:DescribeInstances",
                "ckafka:DescribeTopic",
                "ckafka:DescribeInstanceAttributes",
                "ckafka:CreateToken",
                "ckafka:AuthorizeToken"
            ],
            "resource": "*"
        }
    ]
}
```

投递 COS

管理权限：对所有日志主题具备投递 COS 管理权限

具备所有日志主题投递 COS 的管理权限。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:DescribeIndex",
                "cls:CreateShipper"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cls:ModifyShipper",
                "cls:DescribeShippers",
                "cls:DeleteShipper",
                "cls:DescribeShipperTasks",
                "cls:RetryShipperTask",
                "cls:DescribeShipperPreview",
                "cos:GetService",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList"
            ],
            "resource": "*"
        }
    ]
}
```

管理权限：对指定标签日志主题具备投递 COS 管理权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:DescribeIndex",
                "cls:CreateShipper"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
```

```
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cls:ModifyShipper",
                "cls:DescribeShippers",
                "cls:DeleteShipper",
                "cls:DescribeShipperTasks",
                "cls:RetryShipperTask",
                "cls:DescribeShipperPreview",
                "cos:GetService",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对所有日志主题具备投递 COS 只读权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets" ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cls:DescribeShippers",
                "cls:DescribeShipperTasks",
                "cls:RetryShipperTask",
                "cls:DescribeShipperPreview",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对指定标签日志主题具备投递 COS 只读权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
```

```
                "cls:DescribeTopics",
                "cls:DescribeLogsets"],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cls:DescribeShippers",
                "cls:DescribeShipperTasks",
                "cls:RetryShipperTask",
                "cls:DescribeShipperPreview",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": "*"
        }
    ]
}
```

投递 DLC

管理权限：对所有日志主题具备投递 DLC 管理权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:CreateDlcDeliver",
                "cls:ModifyDlcDeliver",
                "cls:DescribeDlcDelivers",
                "cls:DeleteDlcDeliver"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "dlc:DescribeDatabases",
                "dlc:DescribeOptimizedTables",
                "dlc:DescribeDatasourceConnection",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList"
```

```
            ],
            "resource": "*"
        }
    ]
}
```

管理权限：对指定标签日志主题具备投递 DLC 管理权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:CreateDlcDeliver",
                "cls:ModifyDlcDeliver",
                "cls:DescribeDlcDelivers",
                "cls:DeleteDlcDeliver"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "dlc:DescribeDatabases",
                "dlc:DescribeOptimizedTables",
                "dlc:DescribeDatasourceConnection",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对所有日志主题具备投递 DLC 只读权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets" ],
```

```
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cls:DescribeDlcDelivers",
                "dlc:DescribeDatabases",
                "dlc:DescribeOptimizedTables",
                "dlc:DescribeDatasourceConnection",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对指定标签日志主题具备投递 DLC 只读权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cls:DescribeDlcDelivers",
                "dlc:DescribeDatabases",
                "dlc:DescribeOptimizedTables",
                "dlc:DescribeDatasourceConnection",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": "*"
        }
    ]
}
```

投递 SCF

管理权限：对所有日志主题具备投递 SCF 管理权限

具备所有日志主题投递 SCF 的管理权限。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:CreateDeliverFunction",
                "cls:DeleteDeliverFunction",
                "cls:ModifyDeliverFunction",
                "cls:GetDeliverFunction",
                "scf:ListFunctions",
                "scf:ListAliases",
                "scf:ListVersionByFunction"
            ],
            "resource": "*"
        }
    ]
}
```

管理权限：对指定标签日志主题具备投递 SCF 管理权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:CreateDeliverFunction",
```

```
                "cls:DeleteDeliverFunction",
                "cls:ModifyDeliverFunction",
                "cls:GetDeliverFunction",
                "scf:ListFunctions",
                "scf:ListAliases",
                "scf:ListVersionByFunction"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对所有日志主题具备投递 SCF 只读权限

具备所有日志主题投递 SCF 的只读权限。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:GetDeliverFunction",
                "scf:ListFunctions",
                "scf:ListAliases",
                "scf:ListVersionByFunction"
            ],
            "resource": "*"
        }
    ]
}
```

只读权限：对指定标签日志主题具备投递 SCF 只读权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
```

```
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:GetDeliverFunction",
                "scf:ListFunctions",
                "scf:ListAliases",
                "scf:ListVersionByFunction"
            ],
            "resource": "*"
        }
    ]
}
```

Kafka 协议消费

管理权限：对所有日志主题具备 Kafka 协议消费权限

具备所有日志主题 Kafka 协议消费权限。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeKafkaConsumer",
                "cls:CloseKafkaConsumer",
                "cls:ModifyKafkaConsumer",
                "cls:OpenKafkaConsumer"
            ],
            "resource": [
                "*"]
     },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

管理权限：对指定标签日志主题具备 Kafka 协议消费权限

具备指定标签日志主题 Kafka 协议消费的管理权限。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeKafkaConsumer",
                "cls:CloseKafkaConsumer",
                "cls:ModifyKafkaConsumer",
                "cls:OpenKafkaConsumer"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

管理权限：对指定资源具备 Kafka 协议消费权限

```
{
    "statement": [
        {
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeKafkaConsumer",
                "cls:CloseKafkaConsumer",
                "cls:ModifyKafkaConsumer",
                "cls:OpenKafkaConsumer"
            ],
            "effect": "allow",
            "resource": [
                "qcs::cls:ap-chengdu:100001127XXX:logset/axxxxxx-772e-4971-ad9a-ddcfcfff691b",
                "qcs::cls:ap-chengdu:100001127XXX:topic/590xxxxxxx-36c4-447b-a84f-172ee7340b22"
            ]
```

```
        },
        {
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ]
        }
    ],
    "version": "2.0"
}
```

Kafka 协议消费权限最小权限（非控制台，调用 API）

```
{
    "version": "2.0",
    "statement": [
        {
            "action": [
                "cls:OpenKafkaConsumer"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ]
        }
    ]
}
```

指标投递

管理权限：对所有指标主题具备投递管理权限

```
{
    "statement": [
        {
            "action": [
                "cls:DescribeRemoteWriteTask",
                "cls:DescribeTopics",
                "cls:CreateRemoteWriteTask",
                "cls:ModifyRemoteWriteTask",
                "cls:DescribeLogsets",
                "cls:DeleteRemoteWriteTask",
                "cls:CheckRemoteWriteTaskConnect"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ]
        }
    ],
    "version": "2.0"
}
```

管理权限：对指定标签的指标主题具备投递管理权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeRemoteWriteTask",
                "cls:DescribeTopics",
                "cls:CreateRemoteWriteTask",
                "cls:ModifyRemoteWriteTask",
                "cls:DescribeLogsets",
                "cls:DeleteRemoteWriteTask",
                "cls:CheckRemoteWriteTaskConnect"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "string_equal": {
                    "qcs:resource_tag": "key:value"
                }
            }
        }
    ]
}
```

自定义消费

管理权限：对所有日志主题具备自定义消费管理权限

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateConsumerGroup",
                "cls:ModifyConsumerGroup",
                "cls:DescribeConsumerGroups",
                "cls:DeleteConsumerGroup",
                "cls:DescribeConsumerOffsets",
                "cls:CommitConsumerOffsets",
                "cls:SendConsumerHeartbeat",
                "cls:pullLog"
            ],
            "resource": [
                "*"]
     }
    ]
}
```

DataSight 管理权限

管理权限：对所有 DataSight 独立控制台具备管理权限

用户可以在腾讯云控制台上创建、修改、查看、删除 DataSight 控制台。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateConsole",
                "cls:DeleteConsole",
                "cls:DescribeConsoles",
                "vpc:DescribeSubnetEx",
                "vpc:DescribeVpcEx",
                "cls:ModifyConsole"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

管理权限：对指定 DataSight 独立控制台具备管理权限

用户可以在腾讯云控制台上创建、修改、查看、删除指定 DataSight 控制台。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateConsole",
                "cls:DeleteConsole",
                "cls:DescribeConsoles",
                "vpc:DescribeSubnetEx",
                "vpc:DescribeVpcEx",
                "cls:ModifyConsole"
            ],
            "resource": [
                "qcs::cls::uin/100******123:datasight/clsconsole-1234abcd"
            ]
        }
    ]
}
```

管理权限：对指定标签的 DataSight 独立控制台具备管理权限

用户可以在腾讯云控制台上创建、修改、查看、删除指定标签的 DataSight 控制台。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateConsole",
                "cls:DeleteConsole",
                "cls:DescribeConsoles",
                "vpc:DescribeSubnetEx",
                "vpc:DescribeVpcEx",
```

```
                "cls:ModifyConsole"
            ],
            "resource": [
                "*"
            ],
             "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

只读权限：对所有 DataSight 独立控制台具备只读权限

用户可以在腾讯云控制台上查看 DataSight 控制台的相关信息。

```
{
    "statement": [
        {
            "action": [
                "cls:DescribeConsoles"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ]
        }
    ],
    "version": "2.0"
}
```

只读权限：对指定 DataSight 独立控制台具备只读权限

用户可以在腾讯云控制台上查看指定 DataSight 控制台的相关信息。

```
{
    "statement": [
        {
            "action": [
                "cls:DescribeConsoles"
            ],
            "effect": "allow",
            "resource": [
                "qcs::cls::uin/100******123:datasight/clsconsole-1234abcd"
            ]
        }
    ],
    "version": "2.0"
}
```

只读权限：对指定标签的 DataSight 独立控制台具备只读权限

用户可以在腾讯云控制台上查看指定标签的 DataSight 控制台的相关信息。

```
{
```

```
    "statement": [
        {
            "action": [
                "cls:DescribeConsoles"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ],
    "version": "2.0"
}
```

开发者相关

通过 Grafana 使用 CLS

通过 Grafana 展示所有主题的数据

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:MetricsLabelValues",
                "cls:MetricsQueryRange",
                "cls:MetricsLabels",
                "cls:MetricsQuery"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

通过 Grafana 展示具备指定标签的主题的数据

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
```

```
                "cls:MetricsLabelValues",
                "cls:MetricsQueryRange",
                "cls:MetricsLabels",
                "cls:MetricsQuery"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

使用临时密钥访问 CLS

最近更新时间：2024-09-09 18:25:11

注意：

- 使用临时密钥授权访问时，请务必根据业务需要，按照最小权限原则进行授权。如果您直接授予所有资源(resource:*)，或者所有操作(action:*)权限，则存在由于权限范围过大导致的数据安全风险。
- 您在申请临时密钥时，如果指定了权限范围，那么申请到的临时密钥也只能在权限范围内进行操作。例如您在申请临时密钥时，指定了可以往日志主题 e621fdb8-16f4-41cf-bc73-xxxxxxxxx1 上写日志的权限范围，那么申请到的密钥不能将日志信息写入到 e621fdb8-16f4-41cf-bc73-xxxxxxxxx2，也不能从 e621fdb8-16f4-41cf-bc73-xxxxxxxxx2 中检索日志。

临时密钥

- 临时密钥（临时访问凭证）是通过 CAM 云 API 提供的接口，获取到权限受限的密钥。
- CLS API 可以使用临时密钥计算签名，用于发起 CLS API 请求。
- CLS API 请求使用临时密钥计算签名时，需要用到获取临时密钥接口返回信息中的三个字段，如下：
  - secretId：临时证书密钥 ID
  - secretKey：临时证书密钥 Key
  - secretToken：临时证书密钥 Token

使用临时密钥的优势

Web、iOS、Android 使用 CLS 时，通过固定密钥计算签名方式不能有效地控制权限，同时把永久密钥放到客户端代码中有极大的泄露风险。如若通过临时密钥方式，则可以方便、有效地解决权限控制问题。
例如，在申请临时密钥过程中，可以通过参见 子账户授权策略，限制操作和资源，将权限限制在指定的范围内。

获取临时密钥

获取临时密钥，可以直接请求 STS 云 API 的方式获取。

使用临时密钥访问 CLS

以 CLS Java SDK 为例，使用临时密钥访问 CLS 示例如下：

说明：
运行如下示例前，请前往 Github 项目 获取 Java SDK 安装包。

```
package com.tencentcloudapi.cls;

import com.tencentcloudapi.cls.producer.errors.ProducerException;
import org.junit.Test;

public class AsyncProducerClientTest {

    @Test
    public void testAsyncProducerClient() throws ProducerException, InterruptedException {
        String endpoint = "ap-guangzhou.cls.tencentcs.com";
        // API密钥 secretId，必填
        String secretId = "";
        // API密钥 secretKey，必填
        String secretKey = "";
        // API token，必填
        String secretToken = "";
        // 日志主题ID，必填
        String topicId = "";

        final AsyncProducerConfig config = new AsyncProducerConfig(endpoint, secretId, secretKey,
```

```
            NetworkUtils.getLocalMachineIP(), secretToken);

        // 构建一个客户端实例
        final AsyncProducerClient client = new AsyncProducerClient(config);

        for (int i = 0; i < 10000; ++i) {
            List<LogItem> logItems = new ArrayList<>();
            int ts = (int) (System.currentTimeMillis() / 1000);
            LogItem logItem = new LogItem(ts);
            logItem.PushBack(new LogContent("__CONTENT__", "你好，我来自深圳|hello world"));
            logItem.PushBack(new LogContent("city", "guangzhou"));
            logItem.PushBack(new LogContent("logNo", Integer.toString(i)));
            logItem.PushBack(new LogContent("__PKG_LOGID__",
(String.valueOf(System.currentTimeMillis()))));
            logItems.add(logItem);
            client.putLogs(topicId, logItems, result -> System.out.println(result.toString()));
        }
        client.close();
    }
}
```

日志采集
采集概述

最近更新时间：2024-12-04 20:03:22

概述

日志服务（Cloud Log Service，CLS）提供 LogListener， API 和 SDK 采集方式， 方便用户在多种数据源场景下采集日志并导入到日志服务。

功能特性

- **日志源接入**

  不同的日志源可以选择不同的日志接入方式，详情参见以下列表：

日志源	接入方式
Linux/Unix	LogListener 采集 / Kafka 协议上传 / API 上传日志
Windows	Beats 采集 / Kafka 协议上传 / API 上传日志/ 采集 Windows 事件日志
iOS/Android/HarmonyOS/ Web 端/小程序	SDK 上传日志/ 使用匿名写入采集日志
Nginx	采集 Nginx 访问日志
MySql BinLog	采集 MySql BinLog
Syslog	采集 Syslog
自建 K8s 集群	采集自建 K8S 集群日志
Logstash/FileBeat	Kafka 协议上传
Java 日志	Logback Appender 上传日志/Log4j Appender上传日志
云产品日志	云产品日志接入

- **采集方式**

  日志服务提供多种采集方式：

采集方式	描述
API 方式采集	通过调用日志服务 API 上传结构化日志至日志服务，详情请参见 上传日志接口
SDK 方式采集	通过使用 SDK 上传结构化日志至日志服务，详情请参见 SDK 采集
kafka 协议方式采集	通过使用 Kafka 协议上传日志到 CLS，详情参见 kafka 协议上传日志
LogListener 客户端采集	LogListener 是日志服务提供的日志采集客户端，通过控制台简单配置可快速接入日志服务，详情请参见 LogListener 使用流程

采集方式对比：

类别名称	LogListener 采集	API/SDK 方式采集
修改代码	对应用程序是无侵入式，无需修改代码	需修改应用程序代码才能上报日志
断点续传	支持断点续传日志	自行代码实现
失败重传	自带重试机制	自行代码实现
本地缓存	支持本地缓存，高峰期间保障数据完整	自行代码实现
资源占用	占用内存、CPU 等资源	无额外资源占用

版权所有：腾讯云计算（北京）有限责任公司

- **日志结构化解析**

  日志的结构化指您的日志数据将以 key-value 的形式存储在 CLS 平台上。结构化后的日志数据可以根据指定的键值进行日志检索、日志分析及日志投递。
  日志服务允许直接上报结构化的数据，详见以下说明：
  例如，一条本地原始日志为：

```
10.20.20.10;[Tue Jan 22 14:49:45 CST 2019 +0800];GET /online/sample
HTTP/1.1;127.0.0.1;200;647;35;http://127.0.0.1/
```

  指定日志的解析方式为分隔符方式，以 `;` 分号作为分隔符，可以将该条日志解析为多个字段组，每个字段组按 key-value 键值对的方式进行组织，为每个 key 定义一个键名称，如下所示：

```
IP: 10.20.20.10
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
request: GET /online/sample HTTP/1.1
host: 127.0.0.1
status: 200
length: 647
bytes: 35
referer: http://127.0.0.1/
```

  LogListener 提供多种解析方式，如下表所示：

单行全文格式

单行全文日志是指一行日志内容为一条完整的日志。日志服务在采集的时候，将使用换行符 \n 来作为一条日志的结束符。为了统一结构化管理，每条日志都会存在一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条日志原始数据为：

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

采集到日志服务的数据为：

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

多行全文格式

多行全文日志是指一条完整的日志数据可能跨占多行（例如 Java stacktrace）。在这种情况下，以换行符 \n 为日志的结束标识符就显得有些不合理，为了能让日志系统明确区分开每条日志，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。
多行全文也会设置一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条多行日志原始数据为：

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
    at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
    at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

首行正则表达式为如下：

```
\d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2},\d{3}\s.+
```

采集到日志服务的数据为：

```
__CONTENT__:2019-12-15 17:13:06,043 [main] ERROR
com.test.logging.FooFactory:\njava.lang.NullPointerException\n    at
com.test.logging.FooFactory.createFoo(FooFactory.java:15)\n    at
com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

单行-完全正则格式

单行完全正则格式通常用来处理结构化的日志，指将一条完整日志按正则方式提取多个 key-value 的日志解析模式。
假设一条日志原始数据为：

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703
"http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum"
"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0"  0.354 0.354
```

配置的正则表达式为如下：

```
(\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"
([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
```

采集到日志服务的数据为：

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

多行-完全正则格式

假设您的一条日志原始数据为：

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

行首正则表达式为：

```
\[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*
```

配置的自定义正则表达式为：

```
\[(\d+-\d+-\w+:\d+:\d+,\d+)\]\s\[(\w+)\]\s(.*)
```

系统根据 `()` 捕获组提取对应的 key-value 后，您可以自定义每组的 key 名称如下所示：

```
time: 2018-10-01T10:30:01,000`
level: INFO`
msg: java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

JSON 格式

假设您的一条 JSON 日志原始数据为：

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34
+0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost":"unix:/tmp/php-
cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","request":"POST
/event/dispatch HTTP/1.1","xff":"-","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0
(Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

分隔符

假设您的一条日志原始数据为：

```
10.20.20.10 - ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/1.1 ::: 127.0.0.1
::: 200 ::: 647 ::: 35 ::: http://127.0.0.1/
```

当日志解析的分隔符指定为 `:::` ，该条日志会被分割成八个字段，并为这八个字段定义唯一的 key，如下所示：

```
IP: 10.20.20.10 -
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
```

```
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

组合解析

假设您的一条日志的原始数据为：

```
1571394459, http://127.0.0.1/my/course/4|10.135.46.111|200, status:DEAD,
```

自定义插件内容如下：

```
{
 "processors": [
   {
     "type": "processor_split_delimiter",
     "detail": {
       "Delimiter": ",",
       "ExtractKeys": [ "time", "msg1","msg2"]
     },
     "processors": [
       {
         "type": "processor_timeformat",
         "detail": {
           "KeepSource": true,
           "TimeFormat": "%s",
           "SourceKey": "time"
          }
       },
       {
         "type": "processor_split_delimiter",
         "detail": {
           "KeepSource": false,
           "Delimiter": "|",
           "SourceKey": "msg1",
           "ExtractKeys": [ "submsg1","submsg2","submsg3"]
          },
          "processors": []
       },
       {
         "type": "processor_split_key_value",
         "detail": {
           "KeepSource": false,
           "Delimiter": ": ",
           "SourceKey": "msg2"
         }
       }
     ]
   }
 ]
}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
time: 1571394459
submsg1: http://127.0.0.1/my/course/4
```

```
submsg2: 10.135.46.111
submsg3: 200
status: DEAD
```

Nginx 日志模版

假设您的一条日志的原始数据为：

```
111.121.0.1 - - [06/Aug/2019:12:12:19 +0800] "GET /nginx-logo.png HTTP/1.1" 200 368
"http://119.x.x.x/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/75.0.3770.142 Safari/537.36" "-"
```

Nginx 日志配置内容如下：

```
log_format main '$remote_addr - $remote_user [$time_local] '
 '"$request" $status $body_bytes_sent '
 '"$http_referer" "$http_user_agent"';
```

经过 Nginx 日志配置处理后，生成如下正则表达式：

```
(\S+)\s*-\s*(\S+)\s*\[(\d+\S+\d+:\d+:\d+:\d+)\s+\S+\]\s*\"(\S+)\s+(\S+)\s+\S+\"\s*(\S+)\s*(\S+)\s*\"
([^"]*)\"\s*\"([^"]*)\".*
```

采集到日志服务的数据为：

```
remote_addr: 111.121.0.1
remote_user: -
time_local: 06/Aug/2019:12:12:19
request_method: GET
request_uri: /nginx-logo.png
status: 200
body_bytes_sent: 368
http_referer: http://119.x.x.x/
http_user_agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/75.0.3770.142 Safari/537.36
```

费用说明

日志采集会产生相关日志写入流量费用，更多信息，请您参见 计费概述 。

规格限制

日志采集的规格限制，请您参见：

- LogListener 限制说明
- Kafka 数据订阅
- API/SDK 日志上传限制说明
- COS 数据导入规格限制

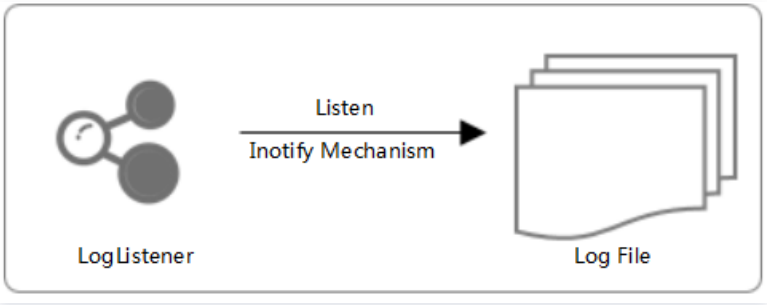
LogListener 采集器采集机制

最近更新时间：2024-07-11 16:43:41

LogListener 是腾讯云日志服务（Cloud Log Service，CLS）所提供的日志采集客户端，它将按照预设的采集策略实时上报日志数据，本文将详细阐述 LogListener 的工作机制。

机制原理

日志服务的 LogListener 成功部署之后，会对所关联的日志文件进行实时监听，主要通过文件系统修改的通知机制 Inotify 来感知目标日志文件的变化，这里的变化不仅是文件内容的变化，对于 Linux 系统而言，也包括文件 inode 发生改变。当 LogListener 感知到日志文件发生变化，就会主动采集上报新写入的日志，并记录当前位置。即使系统重启，也会从记录的位置继续采集日志。



示例说明

为了更直观地说明日志服务 LogListener 采集策略，举例进行说明：

```
2018-01-01 10:00:01 start LogListener
2018-01-01 10:00:02 echo log_1 >> cls.log
2018-01-01 10:00:03 echo log_2 >> cls.log
2018-01-01 10:00:04 echo log_3 >> cls.log
2018-01-01 10:00:05 echo log_4 >> cls.log
......
```

在上述场景中，LogListener 将采集 log_1、log_2、log_3…… 到日志服务当中，并自动监听上报目标文件的所有日志。注意，这里 LogListener 会监控到文件的 inode，若用 vim 修改日志文件 cls.log 时，由于 vim 机制会修改 inode，所以日志系统会认为是一个全新的日志文件，将会采集上报整个文件的内容。

说明

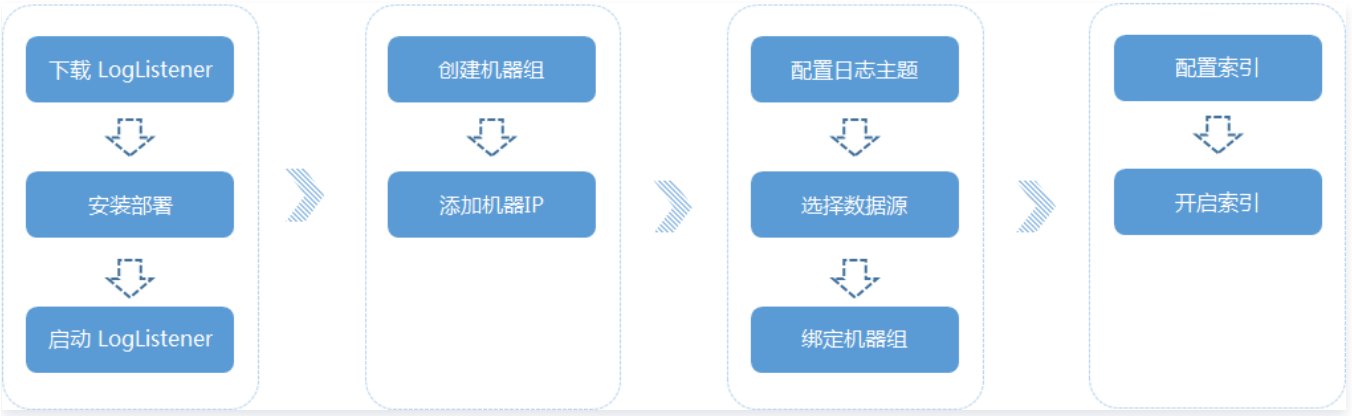
- 机器重启后，会自动拉起 LogListener。
- LogListener 进程挂掉重启后，会根据所记录的偏移位置继续上报日志。
- 目前一个日志文件仅能上报到一个日志主题。若有多个日志主题关联到同一个日志文件，配置信息会覆盖，因此该日志文件实际只会上报到最后一个主题中。

LogListener 使用流程

最近更新时间：2024-07-11 16:43:41

概述

LogListener 是日志服务提供的日志采集客户端，通过安装部署 LogListener，可以方便快速地接入日志服务，无需修改应用程序运行逻辑，是一种对应用服务无侵入式的采集方式。
使用 LogListener 采集日志的流程如下图所示：



流程说明

1. 下载安装最新版本 LogListener 。LogListener 成功安装后将自动启动进程，并与日志服务后端保持心跳连接。
2. 前往 日志服务控制台，创建 机器组，并关联机器。
3. 前往日志主题的采集配置页，绑定机器组，并输入日志路径确定数据源。详细操作示例请参见 采集文本日志 。
4. 前往日志主题的索引配置页，配置全文或键值索引，并开启索引。详细操作示例请参见 索引配置 。

至此，LogListener 将按照日志主题的采集配置监听符合规则的日志文件，用户可以通过日志检索查看采集上来的日志数据。

版权所有：腾讯云计算（北京）有限责任公司

LogListener 安装和部署
LogListener 安装指南（Linux 版）

最近更新时间：2025-05-30 10:38:52

LogListener 是腾讯云日志服务（Cloud Log Service，CLS）所提供的专用日志采集器，将它安装部署到服务器上，可快速采集日志到日志服务。

安装环境

本文将介绍 Linux 操作系统环境下 LogListener 的安装，Windows 操作系统请参见 LogListener 安装指南（Windows版）。

> **说明：**
> 如果其他版本环境若安装异常，请 提交工单 联系我们。

LogListener 版本	处理器架构	操作系统类别	确定可安装环境
v2.x.x	x64/ARM	TencentOS Server	TencentOS Server 3.1、TencentOS Server 2.4
		CentOS（64位）	CentOS_6.8_64位、CentOS_6.9_64位、CentOS_7.2_64位、CentOS_7.3_64位、CentOS_7.4_64位、CentOS_7.5_64位、CentOS_7.6_64位、CentOS_8.0_64位
		Ubuntu（64位）	Ubuntu Server_14.04.1_LTS_64位、Ubuntu Server_16.04.1_LTS_64位、Ubuntu Server_18.04.1_LTS_64位、Ubuntu Server_20.04.1_LTS_64位、Ubuntu Server_22.04.1_LTS_64位、
		Debian（64位）	Debian_8.2_64位、Debian_9.0_64位
		openSUSE（64位）	openSUSE_42.3_64位

支持功能

LogListener 详细版本功能变更信息请参见 LogListener 版本变更。

操作步骤

步骤1：下载安装 LogListener

以安装路径 `/usr/local/` 为例：下载 LogListener 安装包并解压，解压路径为 `/usr/local/` ，解压完成后进入 LogListener 目录 `/usr/local/loglistener/tools` ，执行安装命令 。

x64架构

> **说明：**
> 从2.8.3版本开始，Loglistener 安装包不再添加版本号后缀，使用 loglistener-linux-x64 默认安装 LogListener 最新版本。
> 如果需要安装特定版本可以在后面指定版本号，如将 loglistener-linux-x64替换为 loglistener-linux-x64-2.8.0，将会指定安装2.8.0版本。

- **公网环境**

```
wget http://mirrors.tencent.com/install/cls/loglistener-linux-x64.tar.gz && tar zxvf loglistener-linux-x64.tar.gz  -C /usr/local/ && cd /usr/local/loglistener/tools && ./loglistener.sh install
```

- **内网环境**

```
wget http://mirrors.tencentyun.com/install/cls/loglistener-linux-x64.tar.gz && tar zxvf loglistener-linux-x64.tar.gz  -C /usr/local/ && cd /usr/local/loglistener/tools && ./loglistener.sh
```

```
install
```

ARM 架构

- **公网环境**

```
wget http://mirrors.tencent.com/install/cls/loglistener-linux-ARM-2.9.0.tar.gz && tar zxvf
loglistener-linux-ARM-2.9.0.tar.gz  -C /usr/local/ && cd /usr/local/loglistener/tools &&
./loglistener.sh install
```

- **内网环境**

```
wget http://mirrors.tencentyun.com/install/cls/loglistener-linux-ARM-2.9.0.tar.gz && tar zxvf
loglistener-linux-ARM-2.9.0.tar.gz  -C /usr/local/ && cd /usr/local/loglistener/tools &&
./loglistener.sh install
```

步骤2：初始化 LogListener

以安装路径 `/usr/local/` 为例，在 `/usr/local/loglistener/tools` 路径下，以 root 权限执行 LogListener 初始化命令，初始化命令如下：

```
./loglistener.sh init -secretid AKID****************************** -secretkey
****************************** -region ap-xxxxxx
```

说明：

- 初始化命令中 **-secretid、-secretkey、-region** 为必填的参数，更多其他参数请见如下 参数说明 。
- 若主账号已授权协作者日志服务的读写权限，建议使用协作者密钥。
- -region 为您所使用的日志服务区域，而非您的业务机器所处的区域。当您所使用的日志服务区域与您的业务机器所处的区域不一致时，请额外配置参数-network 为 internet，即代表外网访问。
- 云服务器与日志集同地域的情况下，建议使用内网方式访问服务域名；云服务器与日志集在不同地域的情况下，建议使用外网方式访问服务域名。
- 关于日志采集权限详情，请参见 LogListener 日志采集权限 。

参数说明

参数名	是否必填	类型描述
secretid	是	云 API 密钥 的一部分，SecretId 用于标识 API 调用者身份。请确保云 API 密钥关联的账号具有相应的 LogListener 日志采集权限 。
secretkey	是	云 API 密钥 的一部分，SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥。请确保云 API 密钥关联的账号具有相应的 LogListener 日志采集权限 。
encryption	否	是否对云API密钥进行加密存储。若要对密钥进行加密，则入 true；若不需要对密钥进行加密，则入 false。详情请参见 密钥加密存储 。
network	否	表示 LogListener 通过哪种方式访问服务域名，取值：intra 内网访问（默认），internet 外网访问。 • 内网访问：适用于腾讯云服务器，且服务器所在地域与机器组一致。 • 外网访问：适用于非腾讯云服务器，或服务器所在地域与机器组不一致。

region	若配置了 domain，该参数无需配置，否则必填	region 表示日志服务所在的 地域，此处填写域名简称，例如 ap-beijing、ap-guangzhou 等。 **注意：** 当您所使用的日志服务区域与您的业务机器所处的区域不一致时，请配置参数 network 为 internet，即代表外网访问。
domain	若配置了 region，该参数无需配置，否则必填	表示日志服务所在地域的 域名。例如 ap-beijing.cls.tencentyun.com、ap-guangzhou.cls.tencentyun.com 等。 **注意：** 当您所使用的日志服务区域与您的业务机器所处的区域不一致时，请配置外网域名。例如 ap-beijing.cls.tencentcs.com。
ip	否	机器的 IP 地址，机器组可通过所配 IP 地址与该机器关联，详情请参见 机器组。若不填写，LogListener 会自动获取本机的 IP 地址。
label	否	机器标识，填写后该机器将与同样带有所填机器标识的机器组关联，详情请参见 机器组。多个标识按逗号分隔。 **注意：** 若配置了机器标识， 该机器将只能通过机器标识与机器组关联， 无法通过 IP 进行关联；若不配置， 机器组只能通过 IP 关联该机器。

初始化成功返回样例

```
[root@VM_0_12_centos tools]# ./loglistener.sh init -secretid AKIDPEtPyKabfW8Z3Us
pdz83[illegible] -secretkey whHwQfjdLnzzCE1jIf0[illegible] -region ap-shangha
i -network internet
[OK] check parameter region ap-shanghai ok
[OK] check dependencies ok
Connect to ap-shanghai.cls.tencentcs.com (211.[illegible]) ...
[OK] check network connection ok
[OK] check authentication ok
config parameters change, data will be cleaned, do you want to continue? (yes/no
): yes
[RESULT] loglistener init success, group ip is 172.[illegible]
[root@VM_0_12_centos tools]#
```

步骤3：启动 LogListener

通过 systemd 执行

适用于 Loglistener 2.8.3以上并且操作系统具有 systemd。

```
systemctl start loglistenerd
```

执行 `/etc/init.d/loglistenerd check` 查看是否启动成功：

```
[root@VM_0_12_centos tools]# /etc/init.d/loglistenerd check
[OK] loglistener is running ok
[OK] check loglistener hearbeat ok
group ip:172
host:ap-shanghai.cls.tencentcs.com
port:80
gethostbyname ip:211
[OK] check loglistener config ok
{"logconf":[],"needupdate":false}
[root@VM_0_12_centos tools]#
```

不通过 systemd 执行

适用于 LogListener 2.8.3以下或者 LogListener 2.8.3以上但是操作系统没有 systemd。

```
/etc/init.d/loglistenerd start
```

启动成功返回样例：

```
[root@VM_30_69_centos ~]# /etc/init.d/loglistenerd start
[OK] loglistener is running, ip is 10.
[OK] start loglistener success
```

步骤4：加入机器组

完成 LogListener 安装并启动后，需在 日志服务控制台 创建或选择已有 机器组，并将服务器加入机器组中。您可通过以下两种方式将服务器加入机器组：

- 通过关联机器 IP 地址创建机器组
- 通过机器标识创建机器组

步骤5：采集日志

将服务器加入机器组后，即可 配置采集服务器上的文本日志 。

LogListener 常用操作

> **说明：**
> 本文档示例的操作命令说明仅适用于 LogListener-2.2.4及以上版本，低版本操作命令请参见 低版本 LogListener 操作指南 。

查看 LogListener 版本

```
/etc/init.d/loglistenerd -v
```

查看 LogListener 帮助文档

```
/etc/init.d/loglistenerd -h
```

停止 LogListener

通过 systemd 执行

适用于 Loglistener 2.8.3以上并且操作系统具有 systemd。

```
systemctl stop loglistenerd
```

不通过 systemd 执行

适用于 LogListener 2.8.3以下或者 LogListener 2.8.3以上但是操作系统没有 systemd。

```
/etc/init.d/loglistenerd stop
```

重启 LogListener

通过 systemd 执行

适用于 Loglistener 2.8.3以上并且操作系统具有 systemd。

```
systemctl restart loglistenerd
```

执行 `/etc/init.d/loglistenerd check` 查看是否重启成功：

```
[root@VM_0_12_centos tools]# /etc/init.d/loglistenerd check
[OK] loglistener is running ok
[OK] check loglistener hearbeat ok
group ip:172
host:ap-shanghai.cls.tencentcs.com
port:80
gethostbyname ip:211
[OK] check loglistener config ok
{"logconf":[],"needupdate":false}
[root@VM_0_12_centos tools]#
```

不通过 systemd 执行

适用于 LogListener 2.8.3以下或者 LogListener 2.8.3以上但是操作系统没有 systemd。

```
/etc/init.d/loglistenerd restart
```

重启成功返回样例：

```
[root@VM_30_69_centos ~]# /etc/init.d/loglistenerd start
[OK] loglistener is running, ip is 10.
[OK] start loglistener success
```

查看 LogListener 进程状态

```
/etc/init.d/loglistenerd status
```

LogListener 正常情况会运行两个进程：

```
[root@VM-0-2-centos tools]# /etc/init.d/loglistenerd status
root      4375     1  0 12:27 pts/0    00:00:00 bin loglistenerm -d   守护进程
root      4379  4375  0 12:27 pts/0    00:00:00 bin loglistener --conf=etc/loglistener.conf   采集进程
```

检查 LogListener 心跳及配置

```
/etc/init.d/loglistenerd check
```

```
[root@VM_0_12_centos tools]# /etc/init.d/loglistenerd check
[OK] loglistener is running ok
[OK] check loglistener hearbeat ok
group ip:172
host:ap-shanghai.cls.tencentcs.com
port:80
gethostbyname ip:211
[OK] check loglistener config ok
{"logconf":[],"needupdate":false}
[root@VM_0_12_centos tools]#
```

修改 LogListener 配置

请参见 LogListener 配置指南 。

卸载 LogListener

1. 使用 停止命令 停止运行旧版本的 LogListener。
2. 以安装路径 `/usr/local/` 为例，在 `/usr/local/loglistener/tools` 路径下，以管理员权限执行卸载命令：

```
./loglistener.sh uninstall
```

手动更新 LogListener

复用断点文件（不会重复采集日志）

1. 使用 停止命令 停止运行旧版本的 LogListener。
2. 以安装目录 `/usr/local/loglistener` 为例，前往安装目录下，备份旧版本中的断点文件目录 `./data` 。例如：将旧版的断点文件备份至 `/tmp/loglistener-backup` 目录下。

```
cp -r ./data /tmp/loglistener-backup/
```

3. 使用 卸载命令 卸载旧版本的 LogListener。
4. 下载并安装 最新版本的 LogListener，初始化 新版本 LogListener。
5. 以安装目录 `/usr/local/loglistener` ，复制所备份的断点文件目录（步骤2）到新版本 LogListener 目录下。

```
cp -r /tmp/loglistener-backup/data ./
```

6. 使用 启动命令 启动运行新版本 LogListener。

不复用断点文件（可能会重复采集日志）

1. 使用 停止命令 停止运行旧版本的 LogListener。
2. 使用 卸载命令 卸载旧版本的 LogListener。
3. 下载并安装最新版本 的 LogListener，初始化 新版本 LogListener。
4. 使用 启动命令 启动运行新版本 LogListener。

LogListener 安装指南（Windows 版）

最近更新时间：2025-05-30 10:38:52

LogListener 是腾讯云日志服务（Cloud Log Service，CLS）所提供的专用日志采集器，将它安装部署到服务器上，可快速采集日志到日志服务。

安装环境

LogListener 仅支持64位 Windows Server 操作系统环境，并适配主流 Windows Server 操作系统版本，若需要其他版本环境或安装异常，请 提交工单 联系我们。

LogListener 版本	操作系统类别	确定可安装环境
v2.8.9及以上	Windows Server	Windows Server 2012R2、Windows Server 2016、Windows Server 2019、Windows Server 2022

支持功能

> **注意：**
> 仅2.9.7及以上版本 LogListener 支持采集 Windows 操作系统环境下的文本日志。

- 采集文本日志
- 采集 Windows 事件日志

安装启动 LogListener

1. 下载 LogListener 安装包

文本日志采集

> **注意：**
> 该安装包仅适用于 采集文本日志 。

通过以下链接，下载并解压缩 LogListener 安装包：

- 内网下载链接
- 外网下载链接

Windows 事件日志采集

> **注意：**
> 该安装包仅适用于 Windows 事件日志 采集。

通过以下链接，下载并解压缩 LogListener 安装包：

- 内网下载链接
- 外网下载链接

2. 安装 LogListener

以管理员身份运行 Windows PowerShell，在解压缩后的路径下，执行以下命令安装 LogListener。

```
 .\loglistener_installer.exe install --secret_id AKID****************************** --secret_key
whHwQfjdLnzzCE1jIf09xxxxxxxxxxx --region ap-xxxxxx
```

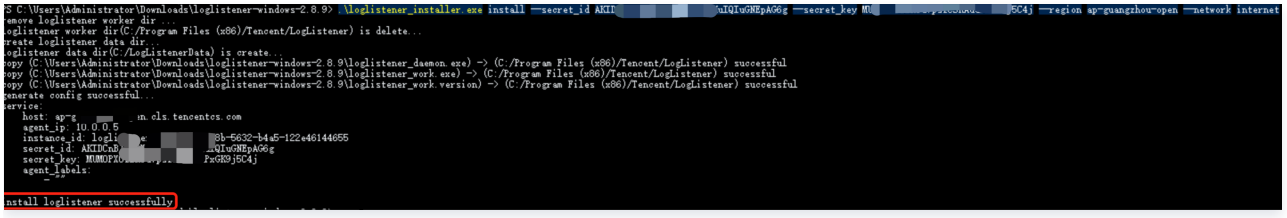
说明：

- 初始化命令中 --secret_id、--secret_key、--region 为必填的参数，更多其他参数请见如下 参数说明 。
- 若主账号已授权协作者日志服务的读写权限，建议使用协作者密钥。
- --region 为您所使用的日志服务区域，而非您的业务机器所处的区域。
- 云服务器与日志集同地域的情况下，建议使用内网方式访问服务域名；云服务器与日志集在不同地域的情况下，建议使用外网方式访问服务域名。
- 关于日志采集权限详情，可参见 LogListener 日志采集权限 文档。
- 执行安装指令后， 将默认在 `C:\Program Files (x86)\Tencent\LogListener` 目录下安装 LogListener。

参数说明

参数名	是否必填	类型描述
secret_id	是	云 API 密钥 的一部分，SecretId 用于标识 API 调用者身份。请确保云 API 密钥关联的账号具有相应的 LogListener 日志采集权限 。
secret_key	是	云 API 密钥 的一部分，SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥。请确保云 API 密钥关联的账号具有相应的 LogListener 日志采集权限 。
encryption	否	是否对云 API 密钥进行加密存储。若要对密钥进行加密，则入 true；若不需要对密钥进行加密，则入 false。详情请参见 密钥加密存储 。（仅文本日志采集的 loglistener 2.9.7.13及以上版本支持）
network	否	表示 LogListener 通过哪种方式访问 CLS 服务，取值：intra 内网访问，internet 外网访问。 • 内网访问：适用于腾讯云服务器，且服务器所在地域与机器组一致。 • 外网访问：适用于非腾讯云服务器，或服务器所在地域与机器组不一致。
region	若配置了endpoint，该参数无需配置，否则必填	region 表示日志服务所在的 地域 ，此处填写域名简称，例如 ap-beijing、ap-guangzhou 等。
endpoint	若配置了 region，该参数无需配置，否则必填	endpoint 表示日志服务所在地域的 域名 。例如 ap-beijing.cls.tencentyun.com、ap-guangzhou.cls.tencentyun.com 等。
ip	否	机器的 IP 地址，机器组可通过所配 IP 地址与该机器关联，详情可参见 机器组 。若不填写，LogListener 会自动获取本机的 IP 地址。
label	否	机器标识，填写后该机器将与同样带有所填机器标识的机器组关联，详情可参见 机器组 。多个标识按逗号分隔。 **注意：** 若配置了机器标识， 该机器将只能通过机器标识与机器组关联， 无法通过 IP 进行关联；若不配置， 机器组只能通过 IP 关联该机器。
check_net	否	是否在安装过程中执行网络连通性校验，默认取值 true。若入参 true，当网络异常时将安装失败。若不需要对网络环境进行校验，则入 false。（Windows 事件采集不支持）

安装成功返回样例



3. 加入机器组

注意：

> 安装了 LogListener Windows 版的服务器只可以加入**系统环境**为 Windows 的机器组。

完成 LogListener 安装并启动后，需在 日志服务控制台 创建或选择已有 机器组，并将服务器加入机器组中。
您可通过以下两种方式将服务器加入机器组：

- 通过关联机器 IP 地址创建机器组
- 通过机器标识创建机器组

4. 采集日志

将服务器加入机器组后， 即可配置 采集 Windows 事件日志 或 采集文本日志 。

LogListener 常用操作命令

查看 LogListener 版本

此处以安装路径 `C:\Program Files (x86)\Tencent\LogListener` 为例，以管理员身份运行 Windows PowerShell，在安装路径下，执行以下命令查看 LogListener 版本：

```
.\loglistener_work.exe -v
```

停止 LogListener

> ⚠ **注意：**
> 执行停止 LogListener 命令时，请确保 LogListener 安装路径下的所有文件均处于关闭状态，否则将无法停止 LogListener。

此处以安装路径 `C:\Program Files (x86)\Tencent\LogListener` 为例，以管理员身份运行 Windows PowerShell，在安装路径下，执行以下命令停止 LogListener：

```
.\loglistener_daemon.exe -action stop
```

重启 LogListener

> ⚠ **注意：**
> 执行重启 LogListener 命令时，请确保 LogListener 安装路径下的所有文件均处于关闭状态，否则将无法重启 LogListener。

此处以安装路径 `C:\Program Files (x86)\Tencent\LogListener` 为例，以管理员身份运行 Windows PowerShell，在安装路径下，执行以下命令重启 LogListener：

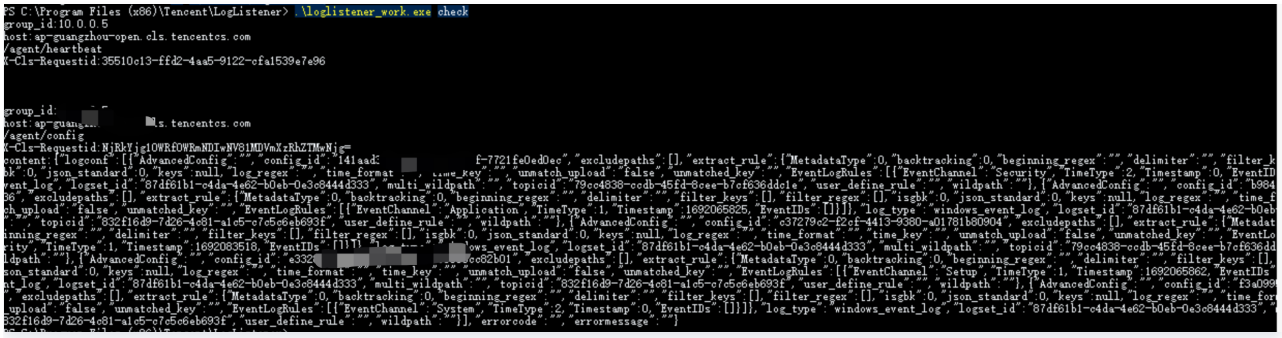
```
.\loglistener_daemon.exe -action restart
```

检查 LogListener 心跳及配置

此处以安装路径 `C:\Program Files (x86)\Tencent\LogListener` 为例，以管理员身份运行 Windows PowerShell，在安装路径下，执行以下命令检查 LogListener 心跳及配置：

```
.\loglistener_work.exe check
```

返回样例：



卸载 LogListener

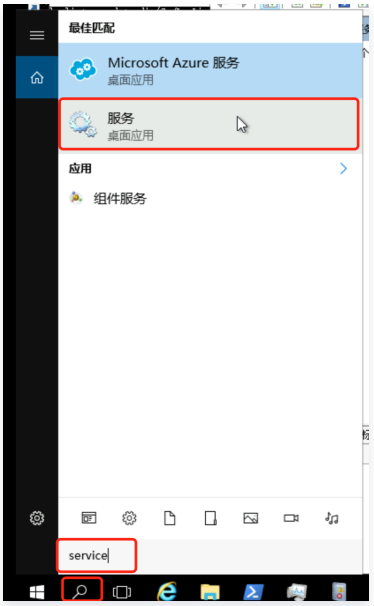
> **注意：**
> - 执行卸载 LogListener 命令时，请确保 LogListener 安装路径下的所有文件均处于关闭状态，否则将无法卸载 LogListener。
> - 卸载工具位于 LogListener 的安装包中，而非在 LogListener 的安装路径下。

1. 使用 停止命令 停止 LogListener。
2. 以管理员身份运行 Windows PowerShell，在安装包路径下，执行以下命令卸载 LogListener：

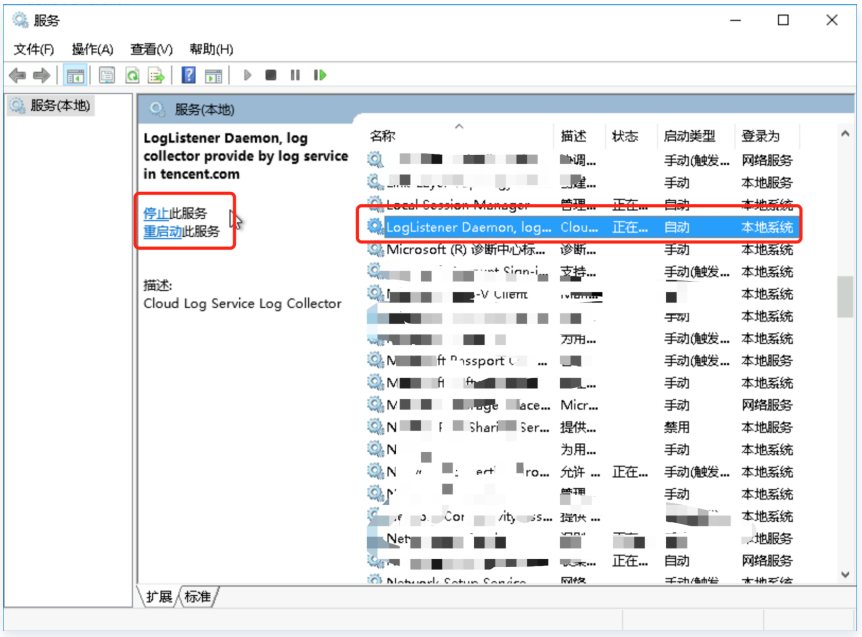
```
.\loglistener_installer.exe uninstall
```

通过服务列表操作 LogListener

1. 单击左下角搜索，并搜索 service。



2. 在服务列表中，找到并单击 **LogListener Deamon**。您可以单击左侧的操作按钮，停止或重启 LogListener。



配置 LogListener

请参见 LogListener 配置指南 。

CVM 与 Lighthouse 批量部署 LogListener

最近更新时间：2024-08-20 16:27:21

概述

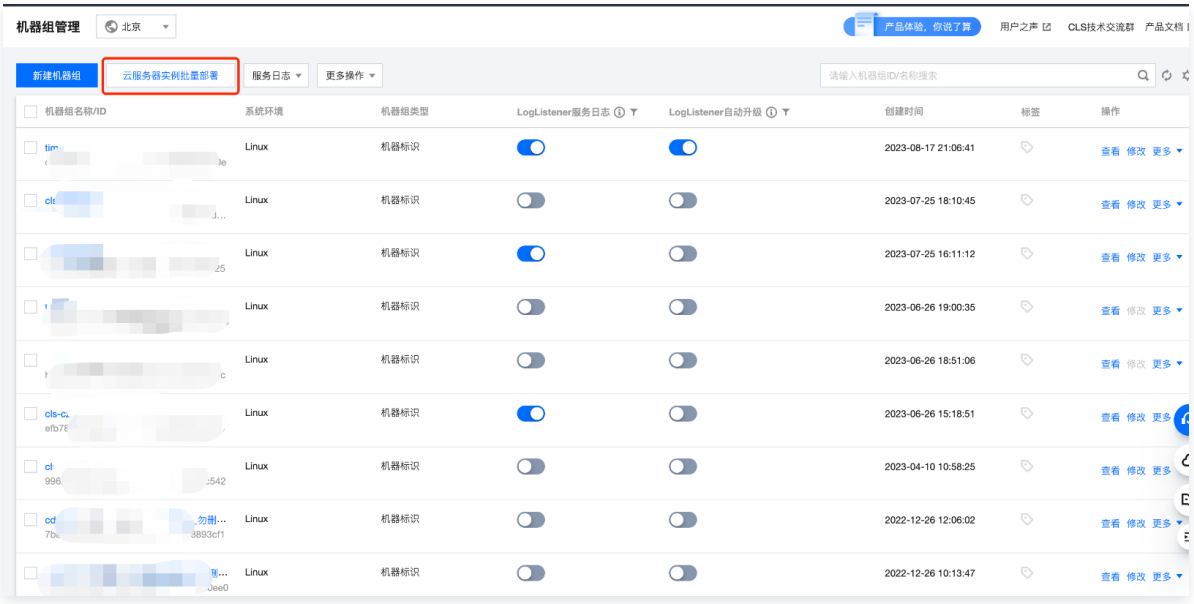
日志服务（Cloud Log Service，CLS）支持使用 LogListener 采集腾讯云云服务器（Cloud Virtual Machine，CVM）与轻量应用服务器（TencentCloud Lighthouse）实例上的日志，在进行日志采集前，需要在 CVM 或 Lighthouse上部署安装 LogListener。为了更快捷的支持大量的 CVM 或 Lighthouse 部署安装 LogListener，现支持用户在控制台选择 CVM 或 Lighthouse 实例，接口批量下发部署 LogListener 任务，自动完成 LogListener 的安装部署（包括 accesskey、ID 配置、地域配置）。

前提条件

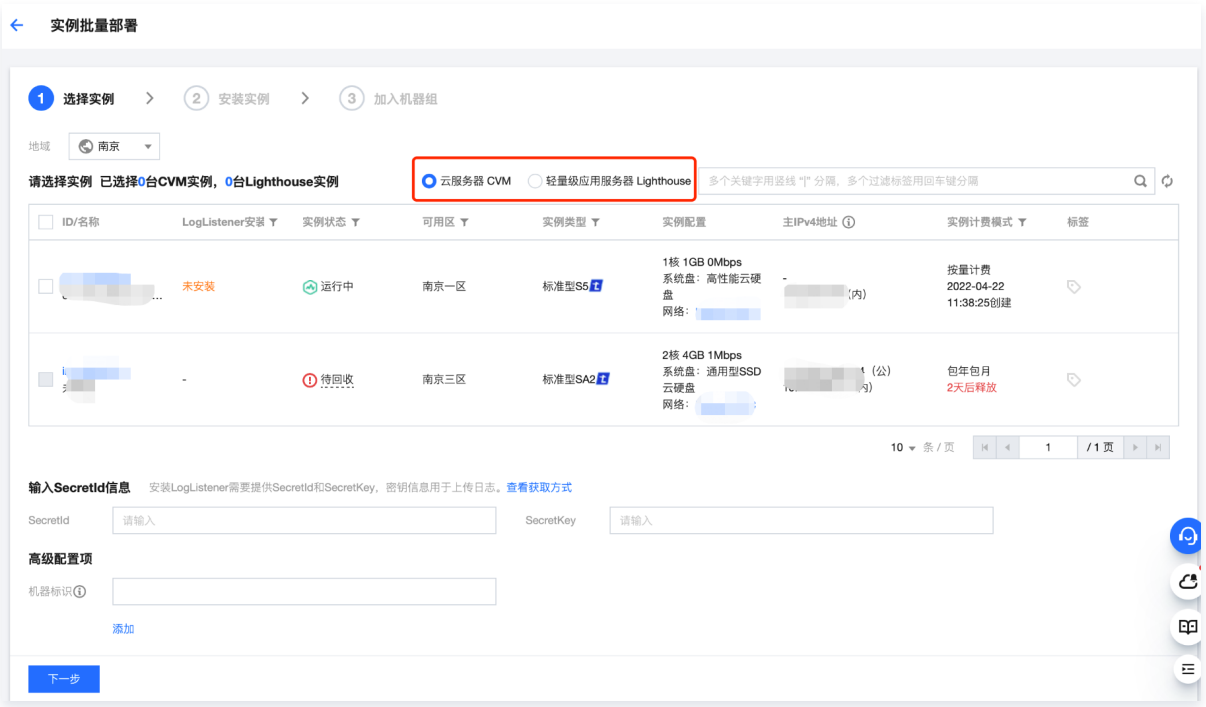
CVM 或 Lighthouse 已 安装腾讯云自动化助手（TencentCloud Automation Tools，TAT）。

操作步骤

1. 登录 日志服务控制台。
2. 在左侧导航栏中，单击**机器组管理**，进入机器组管理页面。
3. 在页面上方单击**云服务器实例批量部署**。



4. 在实例批量部署页面， 选择服务器类型：CVM 或 Lighthouse。（支持同时为 CVM 和 Lighthouse 服务器批量部署 LogListener）



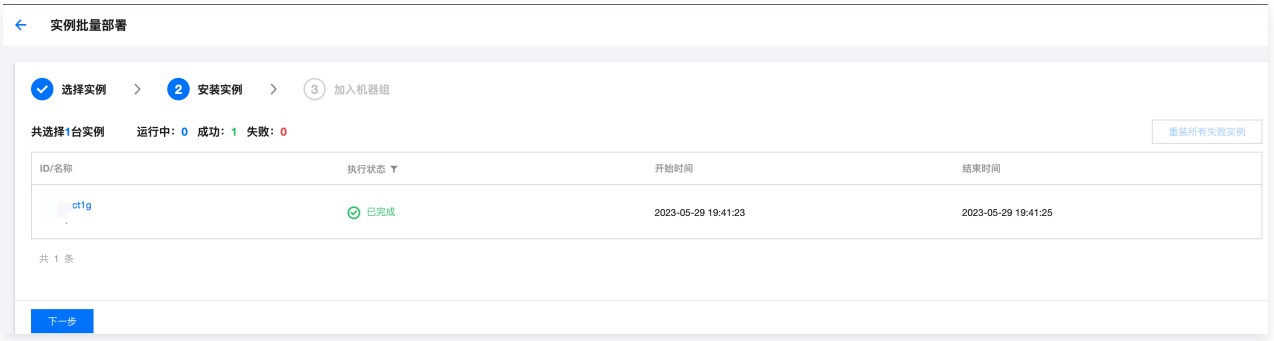
5. 进一步勾选需要部署 LogListener 的 CVM 或 Lighthouse 实例，输入 SecretId 信息（SecretId 和 SecretKey），并根据需求自定义设置高级配置项 – 机器标识。

> **说明**
> - 安装 LogListener 需要提供 SecretId 和 SecretKey，密钥信息用于上传日志，查看获取方式 。
> - 请确保填写密钥信息对应的账号拥有日志上传的权限。 如何配置权限请参见 账号授权 。
> - 如果您的机器 IP 地址经常变动，建议配置机器标识，详情请查看 通过配置机器标识创建机器组 。
> - 已安装 LogListener 机器不支持选择。

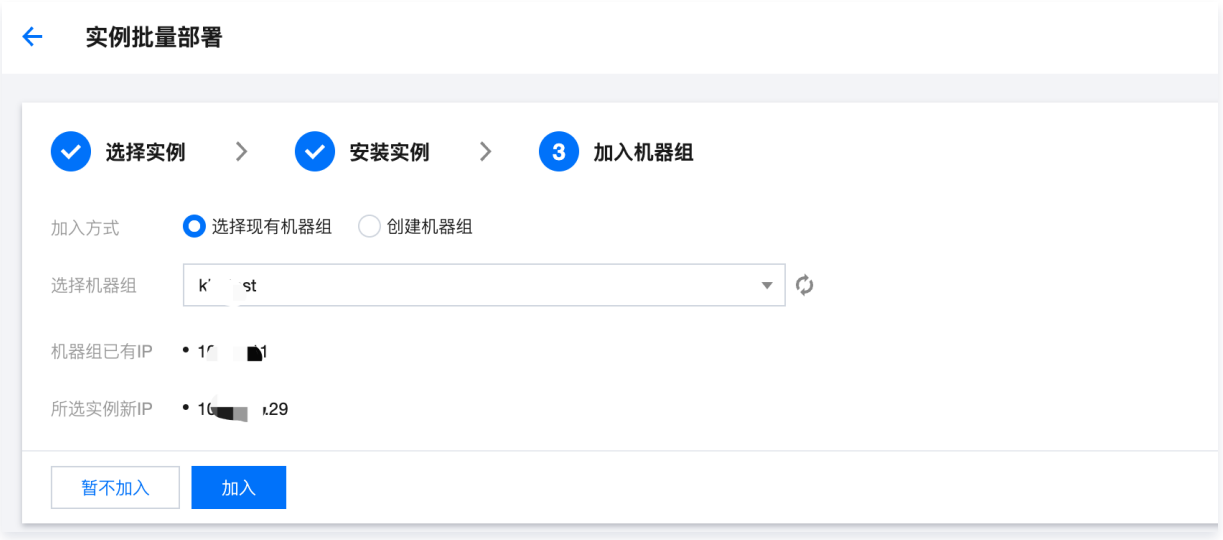


6. 单击**下一步**。
7. 在安装实例页面，待**执行状态**变为**已完成**，即安装完成后，单击**下一步**。

> **说明**
> 如果安装失败，可以将鼠标移动到安装实例的**执行状态**查看失败原因。



8. 在导入机器组页面，根据实际需求，选择现有机器组或者创建机器组，单击**加入**，即可完成批量安装部署。



> **说明**
> 批量安装的 LogListener 为2.6.0及以上版本。

9. 将机器加入机器组后， 即可 配置采集服务器上的文本日志 。

自建 K8s 集群安装 LogListener

最近更新时间：2025-06-27 15:38:52

本文将介绍如何在自建 Kubernetes 集群上进行安装、配置、升级、卸载与排障 LogListener 组件。

前提条件

- 已创建 Kubernetes 1.10及以上版本集群。
- 已安装 kubectl，并且 kubectl 可正常访问集群。
- 已配置自建 K8s 采集所需的云 API 权限，详情请参见 使用自建 K8s 上传数据 文档。
- 已获取日志主题所在地域（region）。详细 CLS 地域列表请参见 可用地域 文档。
- 已获取访问 CLS 侧鉴权所需的 API 密钥 ID（secretid）以及 API 密钥 Key（secretkey），可前往 API 密钥管理 查看。

操作步骤

安装 LogListener

> ⚠ 依赖：
> - 标准的 Kubernetes 集群，不支持 microk8s 或者 k3s 等非标准部署的 Kubernetes 集群。
> - 需要 Helm 版本为3.1以上版本。
> - Linux 环境，不支持 macOS。

1. 安装 Helm。
   安装详情可参见 安装 Helm。
2. 安装 LogListener。
   通过以下命令，将会以 daemonset 的方式在每个 Kubernetes 的非 Master 节点运行一个采集器。若需将采集器运行在 Master 节点中，请参见 Master 节点部署 LogListener 。

```
wget https://mirrors.tencent.com/install/cls/k8s/addon-charts/tencentcloud-cls-k8s.sh
```

```
# 仅支持Linux 环境， 不支持 MacOS!
chmod +x tencentcloud-cls-k8s.sh
```

```
./tencentcloud-cls-k8s.sh  install --region ap-guangzhou --secretid xxx  --secretkey xxx
```

参数说明

参数名	是否必填	类型描述
secretid	是	云 API 密钥 的一部分，SecretId 用于标识 API 调用者身份。请确保云API密钥关联的账号具有相应的 自建 K8s 日志采集上传权限
secretkey	是	云 API 密钥 的一部分，SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥。请确保云 API 密钥关联的账号具有相应的 自建 K8s 日志采集上传权限
region	是	目标 CLS 服务地域，地域详情请参见 可用地域，例如：ap-guangzhou
network	否	表示 LogListener 通过哪种方式访问服务域名，取值：intra 内网访问（默认），internet 外网访问 • 内网访问：适用于集群所在服务器为腾讯云服务器，且服务器所在地域与目标访问 CLS 地域一致 • 外网访问：适用于集群所在服务器为非腾讯云服务器，或服务器所在地域与目标访问 CLS 地域不一致
docker_root	否	集群 Docker 的根目录，默认是 `/var/lib/docker`，如果集群不是这个默认目录，需要指定具体的 Docker 的根目录
cluster_id	否	K8s 集群安装 LogListener 后将自动在 CLS 创建一个相应的机器组，您可通过配置 cluster_id 来指定该机器组的名称，如果不指定，在安装期间会生成一个默认 ID（最好指定一个 cluster_id，默认生成的 ID 的可读性

		较差） ⚠ 注意： 指定的 cluster_id 不可以与已有机器组名称相同。
local_region	否	K8s 集群所在地域，默认赋值与 region 参数一致，若 K8s 集群所在地域与 region 不一致，可自定义修改
api_network	否	云 API 使用内网还是外网，取值：intra 内网访问，internet 外网访问（默认）。仅在专线接入场景下需关注
api_region	否	云 API 的地域。仅在专线接入场景下需关注，默认赋值与 region 参数一致
namespace	否	指定安装日志采集组件的 namespace。若不指定，默认为 kube-system（一般不需要特别指定）
label	否	支持自定义日志采集组件所在 Pod 的 Pod Label。 例如：假设希望为日志采集组件所在 Pod 配置 service=a, app=b 的 Pod Label，此处参数定义为 --label label.service=a,label.app=b
on_master	否	是否在 Master 节点部署。若不指定默认为 false，即不在 Master 节点部署采集组件

示例：广州外网部署

```
./tencentcloud-cls-k8s.sh  install --region ap-guangzhou --secretid xxx  --secretkey xxx --network internet
```

3. **验证安装。**

3.1 查看安装 Helm 包。安装成功后，查看 Helm 安装 tencent-cloud-cls-log。

```
helm list -n kube-system
```

3.2 查看组件。使用以下命令查看组件是否都启动正常，正常情况下，会启动一个 cls-provisioner 的 pod，并在每台宿主机上各启动一个 tke-log-agent 的采集 pod。

```
kubectl get pods -o wide -n kube-system | grep tke-log-agent
```

```
kubectl get pods -o wide -n kube-system | grep cls-provisioner
```

配置日志采集

完成 LogListener 安装后，您即可通过以下两种方式中的任意一种为集群中的日志配置采集。

- 控制台交互式配置自建 K8s 日志采集，详情请参见 使用控制台配置自建 K8s 集群日志采集 。
- 通过编写 CRD YAML 文件配置 K8s 日志采集，详情请参见 使用 CRD 配置自建 K8s 集群日志采集 。

其他操作

配置 LogListener

若要修改 LogListener 的配置（例如最大内存占用），可参见以下步骤修改：

1. 执行以下 kubectl 命令修改 tke-log-agent 环境变量。

```
kubectl edit ds tke-log-agent -n kube-system
```

2. 通过新增环境变量定义 LogListener 配置。

```
spec:
  containers:
  - args:
    - --cred-sources=norm
    - --loglistener-config=/usr/local/loglistener/etc/loglistener.conf
    command:
    - /log-agent
    env:
    - name: CRED_SOURCES
      value: norm
    - name: CLS_HOST
      value: ap-chongqing.cls.tencentyun.com
    - name: TKE_CLUSTER_ID
      value: cls-cz76mmrw
    - name: CLS_ENDPOINT
    - name: LOCAL_REGION
      value: ap-chongqing
    - name: MAX_CONNECTION
      value: "10"
    - name: CHECKPOINT_WINDOW_SIZE
      value: "1024"
    - name: MAX_FILE_BREAKPOINTS
      value: "102400"
    - name: MAX_SENDRATE
      value: ""
    - name: MAX_FILE
      value: "15000"
    - name: MAX_DIR
      value: "5000"
    - name: MAX_HTTPS_CONNECTION
      value: "100"
    - name: CONCURRENCY_TASKS
      value: "256"
    - name: PROCESS_TASKS_EVERY_LOOP
      value: "4"
    - name: CONCURRENCY_TASKS
      value: "256"
    - name: CPU_USAGE_THRES
      value: ""
    - name: MAX_MEM
      value: ""
    - name: POD_NAME
      valueFrom:
        fieldRef:
          apiVersion: v1
          fieldPath: metadata.name
```

3. **参数说明：**

变量名	变量描述
MAX_CONNECTION	最大连接数，默认10
CHECKPOINT_WINDOW_SIZE	单个文件的 checkpoint 环长度，默认1024
MAX_FILE_BREAKPOINTS	位点文件大小，N*2k，N 默认8k
MAX_SENDRATE	最大发送速率，Bytes/s，默认不限制
MAX_FILE	最大监控文件数量，默认15000
MAX_DIR	最大监控目录数量，默认5000
MAX_HTTPS_CONNECTION	Https 最大连接数，默认100
CONCURRENCY_TASKS	LogListener 任务池，默认256 。该配置仅在 LogListener v3.0.0及以上版本支持。 **说明：** LogListener v3.0.0及以上版本为白名单供应，需 提交申请 使用。
PROCESS_TASKS_EVERY_LOOP	单次循环处理任务数，默认4
CPU_USAGE_THRES	LogListener 最大 CPU 利用率，默认与 LogListener 容器的 CPU Limit 保持一致
MAX_MEM	最大内存占用上限，默认与 LogListener 容器的内存 Limit 保持一致。单位：Bytes
MAX_SEND_RETRY_TIME	日志上传请求失败并重试时，最大超时时间（s），默认为10800

升级 LogListener

执行以下命令升级

```
./tencentcloud-cls-k8s.sh  upgrade  --<options>
```

参数说明：

参数	参数说明
upgrade_metrics	升级的时候是否安装 metric 采集组件，默认为 true，升级安装
namespace	日志采集组件安装的 namespace，默认安装在 kube-system，一般不需要特殊指定

查看/更新 LogListener 密钥

自建 Kubernetes 安装部署 LogListener 后，会将用户指定的 secretid 和 secretkey 保存在 Kubernetes 集群的 configmap 中。
您可通过执行以下命令查看密钥：

```
kubectl get configmap -n kube-system cls-credential
```

若要修改密钥，直接编辑修改 configmap 中的密钥和过期时间即可，组件会自动更新密钥。

卸载 LogListener

使用下面命令可以卸载已经安装的 tencent-cloud-cls-log helm 包， 其中 `-n` 为 LogListener 所在的命名空间，通常为 `kube-system` 。

```
helm uninstall tencent-cloud-cls-log -n kube-system
```

> ⚠ **注意：**
> 如果想要完全删除 tencent-cloud-cls-log 包，需执行以下命令删除相关 secret 配置，其中 `-n` 为 LogListener 所在的命名空间，通常为 `kube-system` 。
>
> ```
> kubectl delete secret -n kube-system cls-k8s
> ```

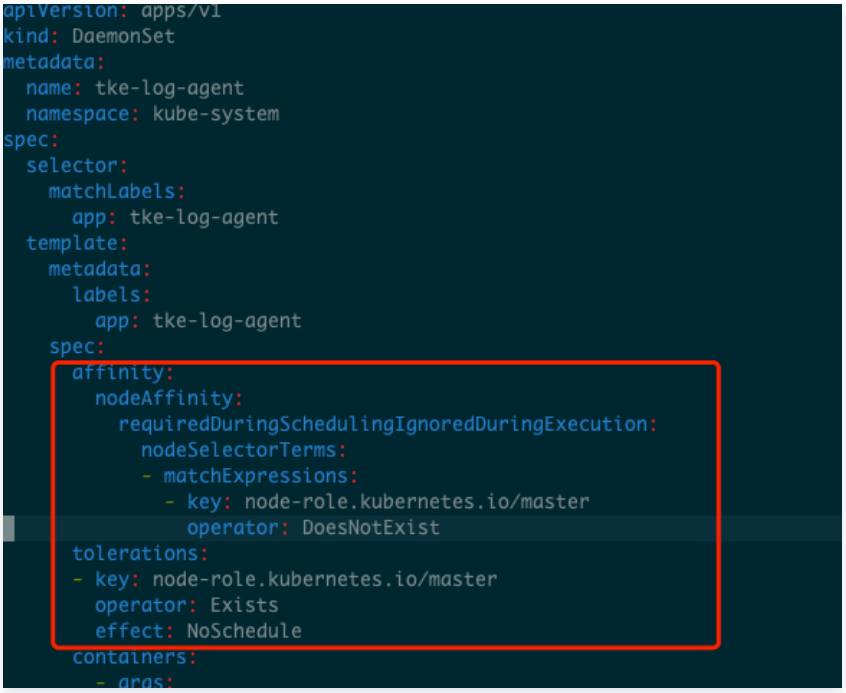
Master 节点部署 LogListener

默认情况下，采集器会以 daemonset 的方式在 Kubernetes 的每个非 Master 节点上，若需要在 Master 节点上运行采集器，需参见以下步骤，调整 tke-log-agent 的 daemonset 声明文件。

1. 执行以下 kubectl 命令修改 tke-log-agent 声明文件。

   ```
   kubectl edit ds tke-log-agent -n kube-system
   ```

2. 删除以下图中的配置项：affinity、tolerations。



```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: tke-log-agent
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: tke-log-agent
  template:
    metadata:
      labels:
        app: tke-log-agent
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: node-role.kubernetes.io/master
                operator: DoesNotExist
      tolerations:
      - key: node-role.kubernetes.io/master
        operator: Exists
        effect: NoSchedule
      containers:
        - args:
```

排障 LogListener

可参见 自建 K8s 日志采集排查指南 文档。

LogListener 升级指南

最近更新时间：2025-05-23 15:16:52

概述

为了更好地提供日志服务（Cloud Log Service，CLS）的日志采集服务，现支持 LogListener 自动升级、控制台一键升级和脚本半自动升级功能。LogListener 采集端程序版本迭代更新之后，用户无需自主下载新版本安装包进行相关操作的手动升级，只需在控制台预设时间段指定机器组进行 LogListener 自动升级，或对目标机器实行一键手动升级即可。

说明：

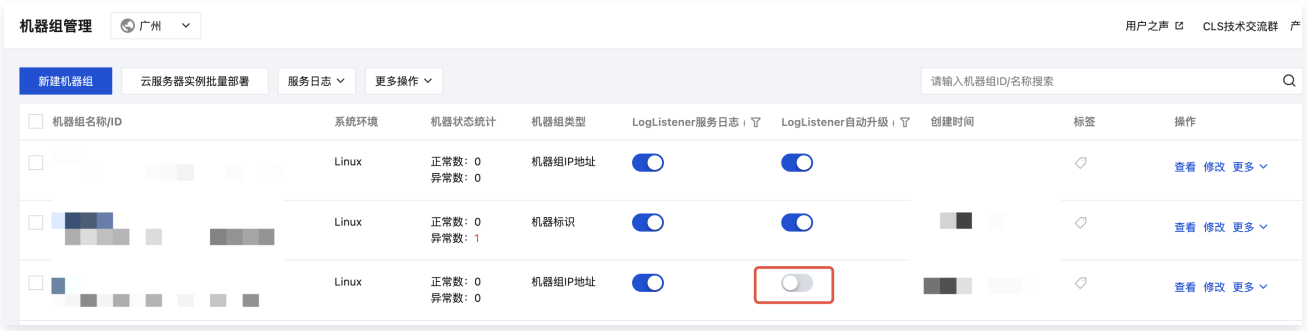
- LogListener 自动升级功能仅在 LogListener 2.5.0及以上版本开始支持，为了更好的使用体验，建议先通过 半自动升级 将 LogListener 升级至2.5.0以上。
- 脚本半自动升级功能需 Python2.7支持，若采集机器上安装的不是该版本，将无法使用此方式进行升级。建议升级 Python 版本，或使用 手动升级 LogListener 。

- 采集端程序 LogListener 控制台升级，可以使用 控制台自动升级 和 控制台一键升级 。
- Agent 升级异常或者因为版本限制无法使用自动升级时，可以使用 半自动升级 或 手动升级 。

操作步骤

控制台自动升级

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**机器组管理**，进入机器组管理页面。
3. 找到需要自动升级的目标机器组，单击 。



4. 在弹出的窗口中，开启 LogListener 自动升级开关，指定升级时间段（默认为当前时间至后两小时，如23:25～01:25）。



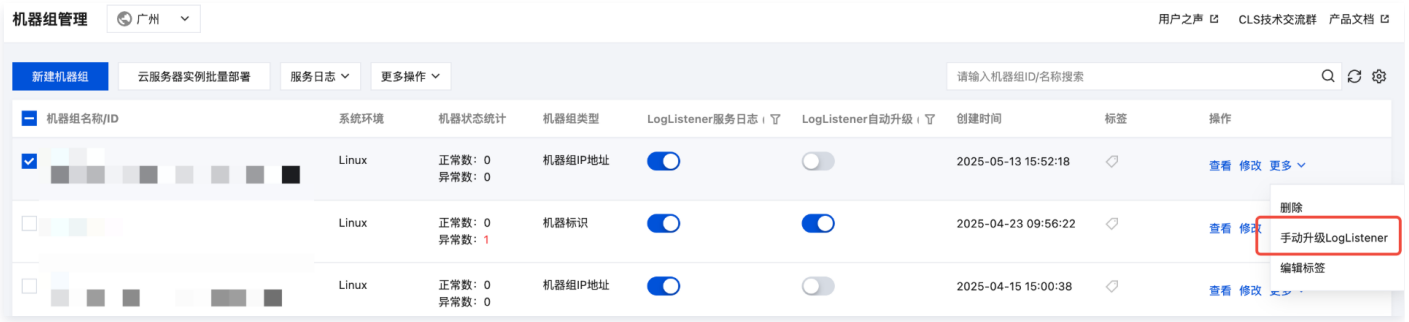
5. 单击**确定**，目标机器组的**自动升级**栏变为 ，即表示开启自动升级 LogListener 成功。

说明：

- 自动升级的时间段可选择任意时间段，系统会在用户指定的时间段每天进行检查。若满足升级条件，则进行自动升级；若不满足升级条件，则不进行操作。
- 如需对多个机器组进行 LogListener 自动升级，可以勾选多个目标机器组，然后单击**更多操作**中的 **Loglistener 自动升级**进行批量升级。

控制台一键升级

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**机器组管理**，进入机器组管理页面。
3. 找到需要升级的目标机器组，单击操作栏的**更多**，在下拉框中找到并单击**手动升级 LogListener**。



4. 在弹出的窗口中，勾选**升级状态**为**可升级**的目标机器，单击**手动升级**。



系统默认升级至最新版本，当**升级状态**为**已是最新版本**时，即表示升级成功。

说明：

- 当升级状态显示**不支持更新**时，表示不支持在控制台一键更新 LogListener，需要您自主下载新版本安装包进行相关的手动升级操作，详情请参见 LogListener 手动升级 。
- 当升级状态显示**心跳异常**时，请 检查机器组状态 。

半自动升级

LogListener 2.9.1及以上版本

以 LogListener 安装路径 `/usr/local/` 为例，进入 `/usr/local/loglistener/tools` 目录中，并执行以下命令：

```
./update auto_upgrade --root-dir /usr/local/loglistener --url
http://mirrors.tencentyun.com/install/cls/loglistener-linux-x64.tar.gz
```

参数描述：

参数	描述
--root-dir	指定新版本 Loglistener 安装目录。此处默认为 `/usr/local/loglistener`
--url	需要升级的 LogListener 的安装包地址。此处默认为最新 LogListener 的安装包地址。若需指定版本，可在 loglistener-linux-x64后增加“-版本号”。例如：2.8.7版本的 LogListener 的安装包地址为 http://mirrors.tencent.com/install/cls/loglistener-linux-x64-2.8.7.tar.gz **⚠ 注意：** 如果输入的版本不存在，则下载失败；如果输入的版本低于机器上已安装的当前版本，则更新不生效。

LogListener 2.9.1以下版本

1. 执行如下命令，下载升级脚本。

```
wget http://mirrors.tencent.com/install/cls/update
```

2. 执行以下命令，添加权限。

```
chmod +x update
```

3. 执行以下命令，执行 LogListener 升级脚本。

```
./update auto_upgrade --root-dir /usr/local/loglistener --url
http://mirrors.tencentyun.com/install/cls/loglistener-linux-x64.tar.gz
```

参数描述：

参数	描述
--root-dir	指定新版本 Loglistener 安装目录。此处默认为 `/usr/local/loglistener`
--url	需要升级的 LogListener 的安装包地址。此处默认为最新 LogListener 的安装包地址。若需指定版本，可在 loglistener-linux-x64后增加“-版本号”。例如：2.8.7版本的 LogListener 的安装包地址为 http://mirrors.tencent.com/install/cls/test/loglistener-linux-x64-2.8.7.tar.gz **⚠ 注意：** 如果输入的版本不存在，则下载失败；如果输入的版本低于机器上已安装的当前版本，则更新不生效。

LogListener 配置指南

最近更新时间：2025-02-28 14:46:22

Linux

以安装路径 `/usr/local/` 为例， 您可在 `/usr/local/loglistener/etc` 路径下，运行以下命令查看并配置 LogListener 配置文件。

```
vim loglistener.conf
```

参数说明

参数名	参数描述	默认值	取值范围
proxy_host	目标 CLS 服务访问域名。	• 若 LogListener 初始化时指定了地域(-region)以及内外网访问方式(-network)，此处默认为指定地域对应的 CLS 内网或外网域名。 • 若 LogListener 初始化未指定地域(-region)而指定了域名(-domain)，此处默认为域名。	请参见 可用地域 配置。
secret_id	API 密钥 ID。	LogListener 初始化时指定的密钥 ID。	请前往 API 密钥管理 获取密钥 ID。
secret_key	API 密钥 Key。	LogListener 初始化时指定的密钥 ID。	请前往 API 密钥管理 获取密钥 Key。
encryption	访问 CLS 服务并鉴权时，是否需要对 secret_id 与 secret_key 进行解密。若要修改密钥的存储方式请参考 密钥加密存储 。	LogListener 初始化时若指定 encryption（是否加密存储密钥信息）为 true，此处则为 true；反之为 false。	true, false
group_ip	IP 地址，机器组可通过所配 IP 地址与该机器关联，详情可参见 机器组 。	默认自动获取网卡IP，若 LogListener 初始化时指定了 IP，则为初始化时指定 IP。	-
group_label	机器标识。配置后，机器组可通过所配机器标识与该机器关联，详情可参见 机器组 。支持配置多个，由逗号分隔。 **⚠ 注意：** 配置了机器标识的机器，仅可通过机器标识关联机器组，不可通过 IP 关联。	-	-
max_connection	同一时刻，最大的日志上传请求并行数量。	10	大于0的整数
max_mem	最大内存占用（Bytes）。	2097152000	大于0的整数
max_dir	最大监听目录数。该配置仅在 LogListener v2.8.8及以上版本支持。	5000	大于0的整数
max_file	最大监听文件数。	15000	大于0的整数
cpu_usage_thres	单核最大 CPU 占用（%）。0代表无限制。	0	0-100
max_send_rate	最大上传速率（Bytes/s）。0代表无限制。	0	大于0的整数

版权所有：腾讯云计算（北京）有限责任公司

max_depth	最大采集目录深度。当采集路径中包含**这种 wildcard 时，该配置可控制针对**的下探深度。0代表不下探。	10	大于0的整数
request_compression	是否压缩上传日志。	true	true、false
replace_special_characters	当采集的日志中出现时出现特殊字符，是否以空格替换。该配置仅在 LogListener v2.8.8及以上版本支持。	false	true、false
memory_tight_mode	是否开启内存节省模式。	false	true、false
multi_process	是否开启多线程模式。该配置仅在 LogListener v3.0.0及以上版本支持。LogListener v3.0.0及以上版本为白名单供应，需 提交申请 使用。	false	true、false
polling_mode	是否开启 Polling 模式。该配置仅在 LogListener v3.0.0及以上版本支持。LogListener v3.0.0及以上版本为白名单供应，需 提交申请 使用。	true	true、false
max_send_retrys	日志上传请求失败时，最大重试次数。0代表不断尝试。	0	大于0的整数
max_send_retry_time	日志上传请求失败并重试时，最大超时时间（s）。	10800	大于0的整数
send_retry_period	日志上传请求失败时，最大重试时间间隔（s）。	3	大于0的整数

完成配置修改后，需执行以下命令重启 LogListener，使配置变更生效。

通过 systemd 执行

适用于 Loglistener 2.8.3以上并且操作系统具有 systemd。

```
systemctl restart loglistenerd
```

执行 `/etc/init.d/loglistenerd check` **查看是否重启成功：**

```
[root@VM_0_12_centos tools]# /etc/init.d/loglistenerd check
[OK] loglistener is running ok
[OK] check loglistener hearbeat ok
group ip:172 [redacted]
host:ap-shanghai.cls.tencentcs.com
port:80
gethostbyname ip:211 [redacted]
[OK] check loglistener config ok
{"logconf":[],"needupdate":false}
[root@VM_0_12_centos tools]#
```

不通过 systemd 执行

适用于 LogListener 2.8.3以下或者 LogListener 2.8.3以上但是操作系统没有 systemd。

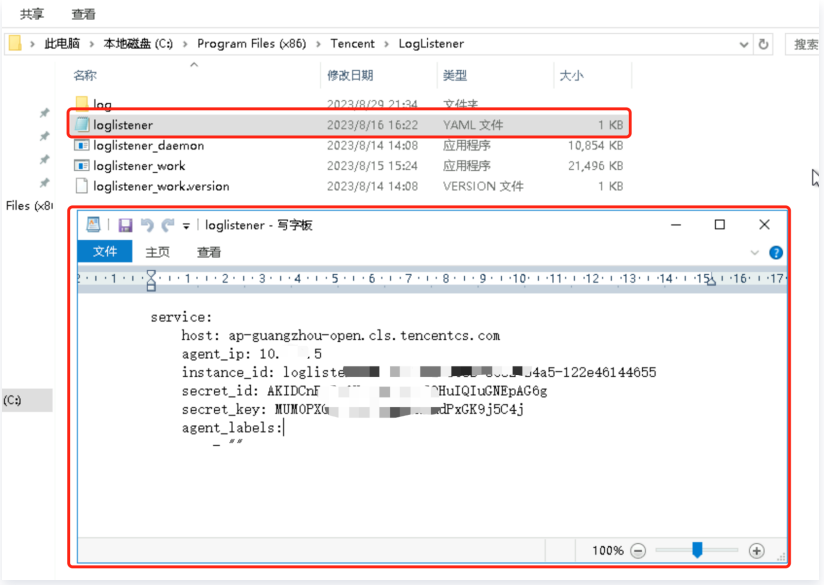
```
/etc/init.d/loglistenerd restart
```

执行 `/etc/init.d/loglistenerd check` **查看是否重启成功：**

```
[root@VM_0_12_centos tools]# /etc/init.d/loglistenerd check
[OK] loglistener is running ok
[OK] check loglistener hearbeat ok
group ip:172
host:ap-shanghai.cls.tencentcs.com
port:80
gethostbyname ip:211
[OK] check loglistener config ok
{"logconf":[],"needupdate":false}
[root@VM_0_12_centos tools]#
```

Windows

以安装路径 `C:\Program Files (x86)\Tencent\LogListener` 为例，在安装目录下找到 loglistener.yaml 并用任意编辑器打开：



参数说明

参数名	参数描述	默认值	取值范围
host	目标 CLS 服务访问域名。	• 若 LogListener 初始化时指定了地域(--region)以及内外网访问方式（--network），此处默认为指定地域对应的 CLS 内网或外网域名。 • 若 LogListener 初始化未指定地域（--region），而指定了域名（--endpoint），此处默认为域名。	请参见 可用地域 配置。
agent_ip	IP 地址，机器组可通过所配 IP 地址与该机器关联，详情可参见 机器组 。	默认自动获取网卡 IP，若 LogListener 初始化时指定了 IP，则为初始化时指定 IP。	-
secret_id	API 密钥 ID。	LogListener 初始化时指定的密钥 ID。	请参见 API 密钥管理 获取密钥 ID。
secret_key	API 密钥 Key。	LogListener 初始化时指定的密钥 ID。	请参见 API 密钥管理 获取密钥 Key。
agent_labels	机器标识。配置后，机器组可通过所配机器标识与该机器关联，详情可参见 机器组 。支持配置多个，并用逗号分隔。	-	-

	注意： 配置了机器标识的机器， 仅可通过机器标识关联机器组，不可通过 IP 关联。		

完成配置修改后，需执行以下命令重启 LogListener，使配置变更生效：
以管理员身份运行 Windows PowerShell，在安装路径 `C:\Program Files (x86)\Tencent\LogListener` 下，执行以下命令重启 LogListener：

```
.\loglistener_daemon.exe -action restart
```

LogListener 服务日志

最近更新时间：2025-06-27 11:06:31

概述

LogListener 服务日志功能支持记录 LogListener 端运行状态和采集监控的日志数据并配置可视化视图，提供重要指标数据，便于用户观测了解 LogListener 的运行状态和日志采集统计情况。

默认配置

默认配置项	配置内容
日志主题	当在某一地域下首次为机器组开启 LogListener 服务日志时，会在该地域自动为您创建一个 cls_service_logging 日志集，并将当前以及未来该地域下所有开启 LogListener 服务日志的机器组所产生的 LogListener 服务日志保存到以下3个日志主题中： • loglistener_status：对应 LogListener 的心跳状态日志。 • loglistener_alarm：对应 LogListener 的采集指标/错误类型监控日志。 • loglistener_business ：对应 LogListener 的采集操作日志，每条日志对应一次请求。
日志存储时间	默认保存7天，不支持修改存储时间。
索引	默认为采集到的所有日志数据开启键值索引。不支持修改索引配置。
仪表盘	默认生成 CLS 日志采集状态监控预置仪表盘。

说明：

- LogListener 服务日志专属用于 LogListener 采集监控产生的日志，不支持写入其他数据。
- LogListener 服务日志功能产生的日志数据不产生费用。
- cls_service_logging 为统一的 LogListener 服务日志的日志集。

前提条件

当前仅支持 Linux 机器组开启服务日志，并且机器组需要关联采集配置才能上报服务日志数据，如果机器组未关联任何采集配置，则不会上报任何服务日志。

应用场景

- **查看 LogListener 状态**
  开通 LogListener 服务日志功能后，您可以查看 LogListener 运行状态和采集统计情况。用户可以通过 service_log_dashboard 仪表盘，查看活跃 LogListener 数、LogListener 状态分布等统计指标。
- **采集端监控配置**
  开通 Loglistener 服务日志功能后，您可以按指标/错误类型，配置采集端监控指标，例如：
  - 根据 MEM、CPU、采集速度、采集延时等指标进行监控。
  - 根据 LogListener 解析错误次数的维度进行监控。
- **文件级监控**
  开通 Loglistener 服务日志功能后，您可以查看文件及目录的监控日志，例如：
  - 某个 IP 上所有文件的采集统计文件。
  - 某个 IP 上某个路径下的采集日志量情况，如 app1 应用日志位于/var/log/app1/，统计这个路径下的日志采集情况。
  - 某个 topic 的采集统计情况。

前提条件

机器组仅在 LogListener 2.5.4 及以上版本支持采集监控服务日志，您可前往升级至 最新版本 。

操作步骤

开通服务日志

1. 登录 日志服务 控制台。
2. 在左侧导航栏中单击**机器组管理**，进入机器组列表页。

3. 在机器组列表页，选择目标机器组，单击 ，即可开启 LogListener 服务日志。



关闭服务日志

1. 登录 日志服务 控制台。
2. 在左侧导航栏中单击**机器组管理**，进入机器组列表页。
3. 在机器组列表页，选择目标机器组，单击 ，即可关闭 LogListener 服务日志。

> **说明：**
> 关闭服务日志功能后，日志集 cls_service_logging 中保存的日志数据不会自动删除，如果您需要删除这部分日志数据，可以手动删除保存服务日志的日志集。

服务日志仪表盘

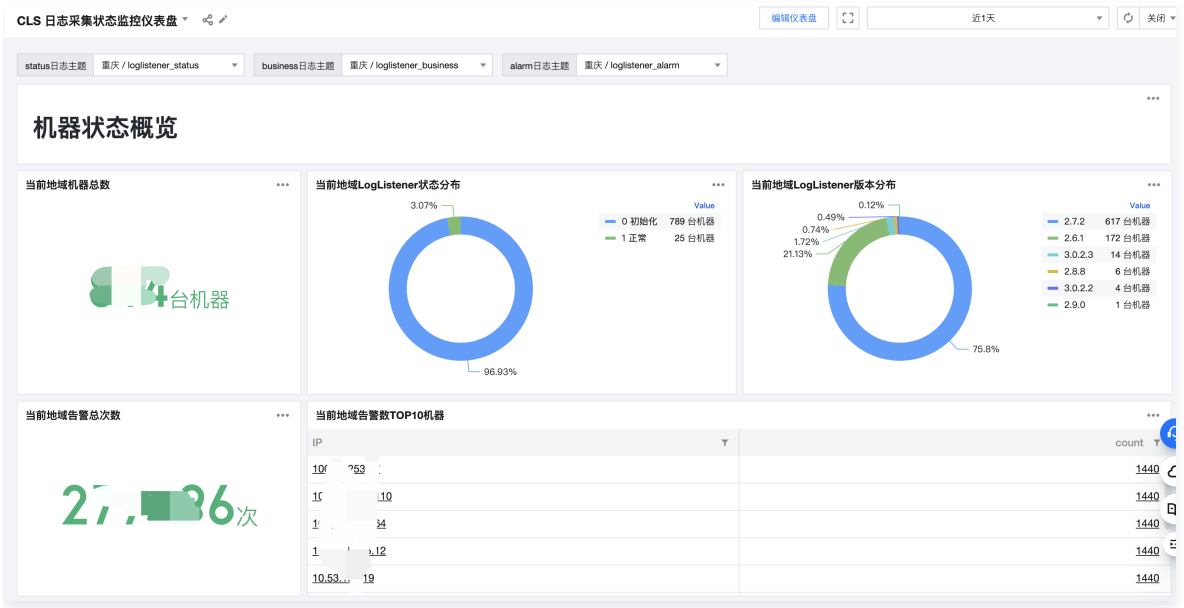
开启 LogListener 服务日志后，日志服务会默认生成 CLS 日志采集状态监控预置仪表盘，展示 LogListener 采集监控统计。

CLS 日志采集状态监控仪表盘

您可以在日志服务控制台的 仪表盘 页面，**预置仪表盘**标签下，找到名称为 **CLS 日志采集状态监控仪表盘**的仪表盘。



单击仪表盘的名称，查看 LogListener 采集相关的统计信息，包括 LogListener 状态展示、LogListener 解析失败率、LogListener 发送成功率等指标信息。



日志字段描述

LogListener 状态日志（Loglistener_status）

日志主题 Loglistener_status 的字段具体说明如下：

字段	描述
InstanceId	LogListener 唯一标识值，可以在 LogListener 安装目录的 `etc/loglistener.conf` 中查看具体值，或者在机器组详情中查看
IP	机器组 IP
Label	机器标识数组
Version	版本号
MemoryUsed	组件内存使用情况
MemMax	Agent 在该机器上设置的内存使用阈值
CpuUsage	组件 CPU 使用率
Status	LogListener 运行状态，取值说明为如下： • 1：心跳正常 • 10：鉴权失败 • 11：日志主题不存在 • 13：日志上传请求超频 • 20：CPU 利用率超限 • 21：目标采集文件数超限 • 22：目标采集目录数超限 • 23：内存占用超限
TotalSendLogSize	发送日志量大小
SendSuccessLogSize	发送成功日志量大小
SendFailureLogSize	发送失败日志量大小
SendTimeoutLogSize	发送超时日志量大小
TotalParseLogCount	解析总日志条数
ParseFailureLogCount	解析失败日志条数
TotalSendLogCount	总发送日志条数
SendSuccessLogCount	发送成功日志条数
SendFailureLogCount	发送失败日志条数
SendTimeoutLogCount	发送超时日志条数
TotalSendReqs	总发送请求数
SendSuccessReqs	发送成功请求数
SendFailureReqs	发送失败请求数
SendTimeoutReqs	发送超时请求数
TotalFinishRsps	收到的全部 .rsp 文件
TotalSuccessFromStart	LogListener 启动到现在总的成功数
AvgReqSize	平均请求包大小
SendAvgCost	平均发送耗时
AvailConnNum	可用连接数

版权所有：腾讯云计算（北京）有限责任公司

QueueSize	排队请求大小

LogListener 告警日志（Loglistener_alarm）

日志主题 Loglistener_alarm 的字段具体说明如下：

监控指标分类	描述
InstanceId	LogListener 唯一标识值
Label	机器标识数组
IP	机器组 IP
Version	LogListener 版本
AlarmType.count	告警类型统计
AlarmType.example	告警类型样例

AlarmType：

alarm type	type ID	描述
UnknownError	0	初始化 alarm 类型
UnknownError	1	解析失败
CredInvalid	2	认证失败
SendFailure	3	发送失败
RunException	4	LogListener 运行异常
MemLimited	5	触发 mem limited 限制
FileProcException	6	文件处理异常
FilePosGetError	7	获取 file pos 失败
HostIpException	8	host IP 线程异常
StatException	9	获取进程相关信息异常
UpdateException	10	cls update 功能异常
DoSendError	11	dosend 失败
FileAddError	12	文件新增失败
FileMetaError	13	元数据文件新增失败
FileOpenError	14	open file 失败
FileReadError	15	read file 失败
FileStatError	16	stat file 失败
GetTimeError	17	getTimeFromLogContent 失败
HandleEventError	18	handle file event 异常
HandleFileCreateError	19	handleFileCreateEvent() 异常
LineParseError	20	log item 解析失败
Lz4CompressError	21	压缩失败

ReadEventException	22	readEvent 失败
ReadFileBugOn	23	触发 bugon
ReadFileException	24	procReadyFile() 异常
ReadFileInodeChange	25	file inode 发生变化
ReadFileTruncate	26	Readfile 截断
WildCardPathException	27	addWildcardPathInotify() 异常

LogListener 采集日志（Loglistener_business）

日志主题 Loglistener_business 的字段具体说明如下：

字段	描述
InstanceId	LogListener 唯一标识值
Label	机器标识数组
IP	机器组 IP
Version	LogListener 版本
TopicId	文件采集到的目标 topic
FileName	文件路径名
RealPath	文件实际路径
FileInode	文件 inode
FileSize	文件大小
LastReadTime	上次读取文件时间
ParseFailLines	时间窗口，解析失败日志条数
ParseFailSize	时间窗口，解析失败日志大小
ParseSuccessLines	时间窗口，解析成功日志条数
ParseSuccessSize	时间窗口，解析成功日志大小
ReadOffset	读取文件的偏移量，单位字节
TruncateSize	时间窗口内，truncate 的文件大小
ReadAvgDelay	时间窗口内，读取平均时延
TimeFormatFailuresLines	时间窗口内，时间戳匹配错误次数
SendSuccessSize	时间窗口内，发送成功日志大小
SendSuccessCount	时间窗口内，发送成功日志条数
SendFailureSize	时间窗口内，发送失败日志大小
SendFailureCount	时间窗口内，发送失败日志条数
SendTimeoutSize	时间窗口内，发送超时日志大小
SendTimeoutCount	时间窗口内，发送超时日志条数
DroppedLogSize	时间窗口内，丢掉日志大小
DroppedLogCount	时间窗口内，丢掉日志条数

ProcessBlock	标记一个统计周期内，当前文件是否触发过采集阻塞（一个文件的滑动窗口10分钟未移动过，即为触发）

LogListener 采集配置导入

最近更新时间：2025-06-27 15:03:21

本文介绍通过 LogListener 采集服务器快速导入其他日志主题的采集配置规则。

操作场景

LogListener 采集配置是指 LogListener 采集服务器在进行日志采集前所设置的采集路径、使用限制、采集模式等采集规则。采集配置规则导入功能支持用户新增或修改采集配置时导入已有日志主题的采集配置，快速配置 LogListener 采集规则，免去多日志主题配置重复的繁琐操作，提高采集配置效率。

说明：

- 默认情况下，一个文件只能被一个 LogListener 配置采集。
- 如果一个文件需对应多个采集配置，请给源文件添加一个软链接，并将其加到另一组采集配置中。
- LogListener 2.3.9及以上版本才可以添加多个采集路径。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，进入日志主题页面。
3. 选择已有的日志主题，单击该日志主题 ID/名称，进入日志主题信息页面。
4. 选择**采集配置**页签，进入**采集配置**页面。
5. 在 **LogListener 采集配置**列表右上角，单击**新增**，进入数据源选择页面。



6. 在数据源页面，选择您要采集的数据源，并单击进入采集流程。



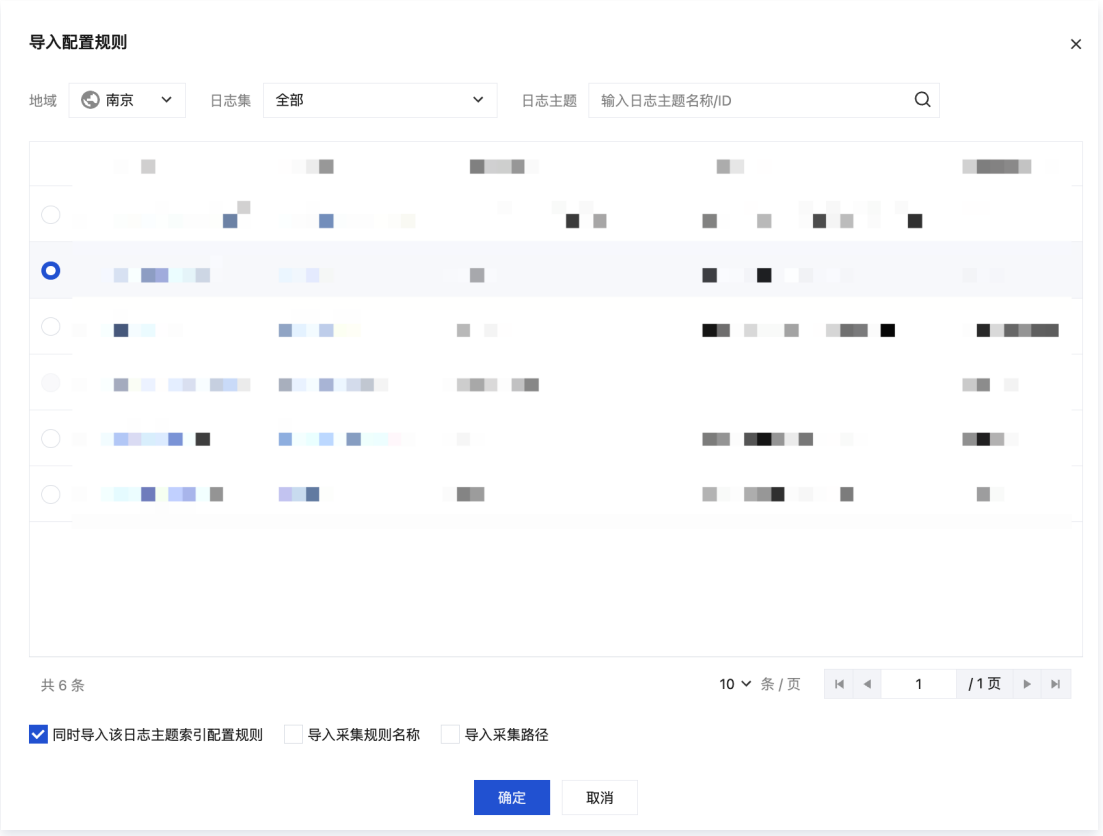
注意：

Windows 事件日志、Logback Appender 上传、Log4j Appender 上传与 SYSLOG 采集不支持导入采集配置。

7. 在采集流程中，完成机器组管理配置后，在采集配置环节中单击左上角**导入配置规则**。



8. 在弹出浮窗显示配置规则列表窗口，选择目标日志主题的配置规则，单击**确定**即可导入当前日志主题的采集配置中。



说明：

- 列表默认展示当前地域所有日志主题支持跨地域日志主题配置规则导入。
- 未配置采集路径的日志主题不可导入当前日志主题采集配置规则中。

采集模式

LogListener 支持通过 单行全文格式、多行全文格式、单行完全正则格式、多行完全正则格式、JSON 格式、分隔符格式 、组合解析 采集文本日志。

LogListener 版本变更

最近更新时间：2024-10-15 18:50:51

本文档为您介绍日志服务 LogListener 的版本更新记录。

说明：
- 采集 __HOSTNAME__ 功能，在 Loglistener 2.7.4版本中开始支持。
- 组合解析功能，在 LogListener 2.6.4版本中开始支持。
- 全量/增量采集策略功能，在 LogListener 2.6.2版本中开始支持。
- CVM 批量部署功能，在 LogListener 2.6.0版本中开始支持。
- 多行-完全正则采集模式，在 LogListener 2.4.5版本中开始支持。
- LogListener 自动升级功能，在 LogListener 2.5.0版本中开始支持。
- 解析失败日志上传功能，在 LogListener 2.5.2版本中开始支持。
- 为了更好的使用体验，建议 前往安装/升级至最新版本 。

版本号	变更类型	描述	相关文档
V2.9.5	体验优化	• 修复整形时间戳无法解析问题 • 修复 CentOS 6.8无法安装问题	-
2.9.1	体验优化	• 支持 instanceID 动态生成 • JSON 解析性能优化 • 修复连续升级异常场景 • 修复升级后无法正常上报心跳问题	-
V2.9.0	体验优化	解决 LogListener 过启动时采集异常的问题	-
V2.8.9	新功能	• LogListener 支持 Windows 操作系统 • 支持采集 Windows 事件日志	采集 Windows 事件日志
V2.8.8	新功能	• 自定义元数据 • 支持多行模式下的解析失败合并上传功能 • 支持在 CentOS 9及以上的系统上进行安装	自定义元数据
	体验优化	• 优化文件路径模式的黑名单处理逻辑 • 修复启动异常后无限重启的问题 • 修复单行全文时配置了正则规则，导致的重启问题 • 修复关闭 LogListener 可能会卡死的问题	-
v2.8.6	体验优化	• 修复部分场景下，自动升级异常 • 修复 JSON 解析时 string 类型数据失真的问题	-
v2.8.5	新功能	支持采集配置级别目录深度、file timeout 高级配置选项	-
	体验优化	• 修复部分场景下，采集容器标准输出出现的内存泄漏问题 • 修复多行采集模式下，有可能造成的 crash 问题	-
v2.8.4	新功能	支持微秒时间戳解析	-
	体验优化	• 优化容器标准输出 16k 截断问题 • 修复某些日志轮转场景下的内存泄漏问题 • 修复 hostname 元数据乱码问题 • 修复标准 JSON 解析时，整型数据被转义成 text 型的问题	-
v2.8.3	体验优化	• 变更默认安装方式，更换为 systemd 管理	-

		• 修复部分 agent 状态无法自动恢复的问题 • 修复某些场景下，存量文件无法采集完的问题 • 优化文件轮转处理逻辑，避免重采 • 修复移除软链接时偶发的 crash 问题	
v2.8.2	体验优化	• 修复多软链接+不同通配路径场景下，日志重采问题	-
v2.8.1	体验优化	• 修复 JSON 解析异常 • 修复多行模式下的过滤器异常和单 KV 超过 1M 造成的上传异常	-
v2.8.0	新功能	Loglistener 支持采集 GBK 编码模式的日志文本	-
v2.7.9	体验优化	• 增加 loglistener 文件锁校验，默认只能启动一个 agent 实例 • 优化 containerd stdout 空行处理异常 • 优化文件句柄泄露导致的磁盘满、业务异常问题 • 优化多行日志行数过多时，后半部分内容被解析失败的问题	-
v2.7.8	体验优化	优化容器场景下 metadata 文件生成延迟，造成日志无 TAG 元数据的问题	-
v2.7.7	体验优化	优化 DNS 解析异常恢复后，采集程序网络连接无法恢复的问题	-
v2.7.6	体验优化	优化 hostname 提取时的换行符处理	-
v2.7.5	体验优化	优化真实文件及其同目录软链接同时采集（不同采集配置）时，文件轮转情况下的处理异常	-
v2.7.4	新功能	• Loglistener 会默认采集机器的主机名作为默认字段上报，以 __HOSTNAME__ 作为 key 展现，例如 __HOSTNAME__：VM-108-centos • 组合解析增加 meta_processor，支持自定义元数据解析（路径）	-
	体验优化	• 修复在文件删除场景的漏采问题 • 修复由于文件尾无换行符引起的文件大小判断出错，导致的文件重采的问题	-
v2.7.3	新功能	单 agent 实例支持同时多 endpoint 上传日志	-
v2.7.2	体验优化	修复优化轮转文件在移除时无法清理掉对应的配置缓存，造成内存泄漏的问题	-
v2.7.1	体验优化	修复大量打印 processor 为空日志的问题	-
v2.7.0	体验优化	修复空字符串在封装 PB 时，有可能引发异常，导致采集阻塞的问题	-
v2.6.9	体验优化	修复多行解析失败场景下，无效日志超量打印的问题	-
v2.6.8	体验优化	• 优化增加 loglistener 采集规格限制，超限启动保护机制 • 修复 Ubuntu 开机启动不生效问题 • 优化黑名单功能，节省内存使用 • 修复组合解析模式，且 root processor 为正则解析插件时的处理异常 • 优化部分日志打印	-
v2.6.7	新功能	支持单 agent 下多租户采集能力	-
v2.6.6	体验优化	修复软链接场景下，对于写入量很小的文件，可能发生漏采/延迟采集的问题	-
v2.6.5	新功能	日志时间支持时区信息解析	-
	体验优化	• 修复高级数据处理空指针异常 • 修复当多个文件同时轮转时异常问题	-
v2.6.4	新功能	Loglistener 支持使用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	组合解析格式
	体验优化	• 优化日志解析格式 pipeline • 修复对毫秒时间戳（%F）格式解析的问题	-

v2.6.3	体验优化	• 修复 checkpoint 文件损坏时，loglistener 无法启动的问题 • 修复特殊场景下，黑名单对新文件不生效的问题	-
v2.6.2	新功能	支持增量采集功能	-
	体验优化	• 修复文件在从扫描到处理之间被移除场景下的采集忽略问题 • 修复自动升级异常覆盖的问题	-
v2.6.1	体验优化	• 修复某些场景下，日志轮转时可能发生回溯采集的问题 • 调整采集端上传日志超时时间，避免因为 timeout 导致数据重复	-
v2.6.0	新功能	• 支持用户在控制台选择 CVM 实例，接口批量下发部署 LogListener 任务，自动完成 LogListener 的安装部署 • 支持 secret ID/KEY 密文存储	CVM 批量部署 LogListener
	体验优化	• 优化 loglistener install/stop 逻辑 • 优化 upload 失败场景下的重试策略 • 增加对老版本 glibc 库造成的 dead lock 的检测修复工具 • 采集性能优化	-
v2.5.9	体验优化	优化资源限制策略	-
v2.5.8	体验优化	• 修复当移除一个目录软链接时，影响到其它指向相同目标的目录软链接的采集的问题 • 修复当移除一个目录软链接并再次创建相同软链接后，目录下文件无法采集的问题	-
v2.5.7	体验优化	• 修复当 filesize 大于2G时，会存在重复采集的问题（新引入） • 修复当文件数量特别多的时候，文件发生 rename 有可能会导致程序卡住的问题。 • 修复文件采集监控中，某字段无法更新的问题	-
v2.5.6	体验优化	修复特殊使用场景下，触发采集程序异常，停止工作的问题	-
v2.5.5	体验优化	• 优化文件采集元数据 checkpoint ，保证重启不丢数据 • 支持资源限制可配置超限处理，内存、CPU、带宽	-
v2.5.4	新功能	支持 LogListener 服务日志功能，该功能记录 LogListener 端运行状态和采集监控的日志数据并配置可视化视图，提供重要指标数据	LogListener 服务日志
	体验优化	增强内存资源限制处理，当内存超限持续一段时间后，LogListener 自动加载	-
v2.5.3	性能优化	修复内存问题引发 LogListener 工作异常	-
v2.5.2	新功能	支持所有解析失败的日志，均以 LogParseFailure 作为键名称（Key），原始日志内容作为值（Value）进行上传	-
	体验优化	优化黑名单功能，黑名单 FILE 模式支持通配符过滤	-
v2.5.1	体验优化	优化当采集文件找不到断点元数据时的处理	-
v2.5.0	新功能	• 支持用户在控制台预设时间段指定机器组进行 agent 自动升级，也可对目标机器实行手动升级 • 支持在 Ubuntu 系统下，LogListener 自启动	LogListener 升级指南
v2.4.6	体验优化	• 变更采集配置时，清理相关配置 cache 的数据残留 • 修复处理软链接的 IN_DELETE 事件时，影响其他指向此 realpath 文件的软链接文件采集的问题 • 优化同一源文件同时使用文件软链接和目录软连接进行采集功能	-
v2.4.5	新功能	LogListener 采集配置规则新增多行-完全正则提取模式	完全正则（多行）
v2.4.4	体验优化	优化 msec 功能导致的日志采集使用日志时间不准确的问题	-
v2.4.3	新功能	支持自动检测日志格式（logFormat）	-

v2.4.2	体验优化	优化腾讯云容器场景下拉取配置时缓存淘汰问题	-
v2.4.1	新功能	支持毫秒采集日志数据	-
	体验优化	优化用户日志中无换行符数据引发的工作异常	-
v2.4.0	新功能	LogListener 支持进程实例级别监控	-
v2.3.9	新功能	支持采集路径配置黑名单	-
	性能优化	优化 boost 版本库过低导致的内存泄漏	-
v2.3.8	新功能	采集配置支持多路径	-
v2.3.6	性能优化	• 修复无效键值 key invalid 导致的停止采集问题 • 修复请求失败返回502导致的内存泄漏问题	-
v2.3.5	新功能	支持日志上下文检索功能	-
	性能优化	• 修复在静态配置模式下，在上传日志时返回鉴权失败时后续不再采集的问题 • 修复在动态配置模式下，内存超过阈值后，不再读取动态配置的问题 • 修复在日志滚动时，如果生产日志速度过大，偶现重复采集的问题 • 修复在日志上传重试多次失败时，导致的内存泄漏的问题	-
v2.3.1	体验优化	• 内存限制优化 • 达到内存限制时，超过3s的请求判定为超时	-
v2.2.6	新功能	支持分离配置内外网域名	-
	性能优化	修复 getip 引发的 LogListener 工作异常	-
v2.2.5	新功能	支持腾讯云织云环境部署	-
	体验优化	修复 getip 导致 core 的问题	-
v2.2.4	体验优化	安装和初始化改为：tools/loglistener.sh 的子命令 install 和 init 启动改成: `/etc/init.d/loglistenerd start|stop|restart`	-
v2.2.3	体验优化	日志轮转 rename+create 不丢日志	-
v2.2.2	体验优化	日志大小超过512KB自动截断	-
更早版本	-	• 2.2.2版本的 LogListener 支持完全正则采集 • 2.1.4版本的 LogListener 支持多行全文格式 • 2.1.1版本的 LogListener 支持日志结构化	-

版权所有：腾讯云计算（北京）有限责任公司

密钥加密存储

最近更新时间：2025-04-03 17:39:52

操作场景

LogListener 访问 CLS 服务端需要通过密钥鉴权，LogListener 初始化后，默认会将密钥明文写入并存储于其配置文件中。若不希望明文存储（明文存储可能存在泄露风险，请谨慎操作），本文档将介绍如何在 LogListener 初始化时指定是否加密存储密钥，以及初始化后如何修改密钥的存储方式。

操作步骤

初始化时指定加密存储密钥

参考 LogListener 安装指南，在初始化 LogListener 时，添加 `-encryption` 参数。若要对密钥进行加密，则入　true；若不需要对密钥进行加密，则入 false。

初始化后修改密钥存储方式

修改为加密存储

1. 前往 LogListener 安装目录 。
2. 在 LogListener 安装目录下，执行 `./bin/encrypt_tool -e {密钥 ID}`， 获取加密后的密钥 ID。
3. 在 LogListener 安装目录下，执行 `./bin/encrypt_tool -e {密钥 KEY}`， 获取加密后的密钥 KEY。
4. 在 LogListener 安装目录下，执行 `vim ./etc/loglistener.conf`，将 conf 文件中的 secret_id 和 secret_key 替换为步骤1、2中获取的加密密钥 ID 和加密密钥 KEY，同时将 encryption 设为 true。



5. 执行以下命令重启 LogListener。

通过 systemd 执行

适用于 LogListener2.8.3以上并且操作系统具有 systemd。

```
systemctl restart loglistenerd
```

不通过 systemd 执行

适用于 LogListener2.8.3以下或者 LogListener2.8.3以上但是操作系统没有 systemd。

```
/etc/init.d/loglistenerd restart
```

修改为明文存储

1. 前往 LogListener 安装目录。
2. 在 LogListener 安装目录下，执行 `vim ./etc/loglistener.conf`，将 conf 文件中的 secret_id 和 secret_key 替换为明文密钥 ID 和密钥 KEY，同时将 encryption 设为 false。

3. 执行以下命令重启 LogListener。

通过 systemd 执行

适用于 LogListener2.8.3以上并且操作系统具有 systemd。

```
systemctl restart loglistenerd
```

不通过 systemd 执行

适用于 LogListener2.8.3以下或者 LogListener2.8.3以上但是操作系统没有 systemd。

```
/etc/init.d/loglistenerd restart
```

采集文本日志
使用单行全文提取模式采集日志

最近更新时间：2025-07-03 17:35:32

操作场景

单行全文提取模式指导您在采集的时候，使用换行符 `\n` 来作为一条日志的结束符，并将每一条日志存储在一个默认的键值 `__CONTENT__` 中。日志数据本身不进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。本文为您介绍如何使用单行全文提取模式采集日志。

前提条件

- 目标文件所在服务器已安装 LogListener。详情请参见：
  - LogListener 安装指南（Linux 版）
  - LogListener 安装指南（Windows 版）
- LogListener Linux 版本为2.1.1及以上或 LogListener Windows 版本为2.9.7及以上。

效果预览

假设您的一条日志原始数据为：

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

日志最终被日志服务处理为：

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

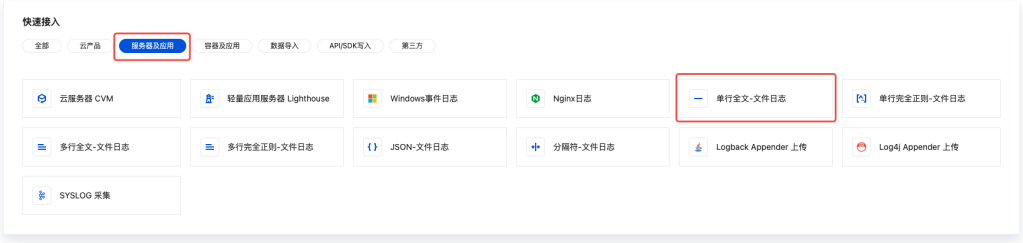
操作步骤

步骤1：创建/选择日志主题

创建新的日志主题

如果您想创建新的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**概览**，进入概览页面。
3. 在**快速接入** > **服务器及应用**中，找到并单击**单行全文-文件日志**，进入 LogListener 采集配置流程。



4. 在**创建日志主题**页面，根据实际需求，输入日志主题名称，配置日志存储时间，选择日志集等信息，单击**下一步** 。



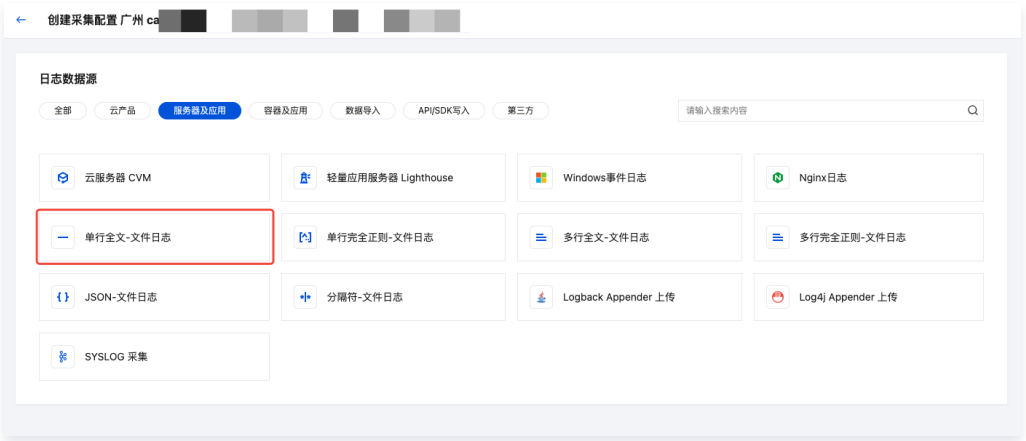
使用现有的日志主题

如果您想选择现有的日志主题，可执行如下操作：

1. **登录** 日志服务控制台 **。**
2. **在左侧导航栏中，单击日志主题，选择需要投递的日志主题，单击日志主题名称，进入日志主题管理页面。**
3. **选择采集配置页签，在 LogListener 采集配置栏下单击新增，进入采集配置流程。**



4. **在日志数据源选择页面中，选择服务器及应用，找到并单击单行全文-文件日志，进入机器组管理流程。**



步骤2：管理机器组（以从概览页创建/选择日志主题为例）

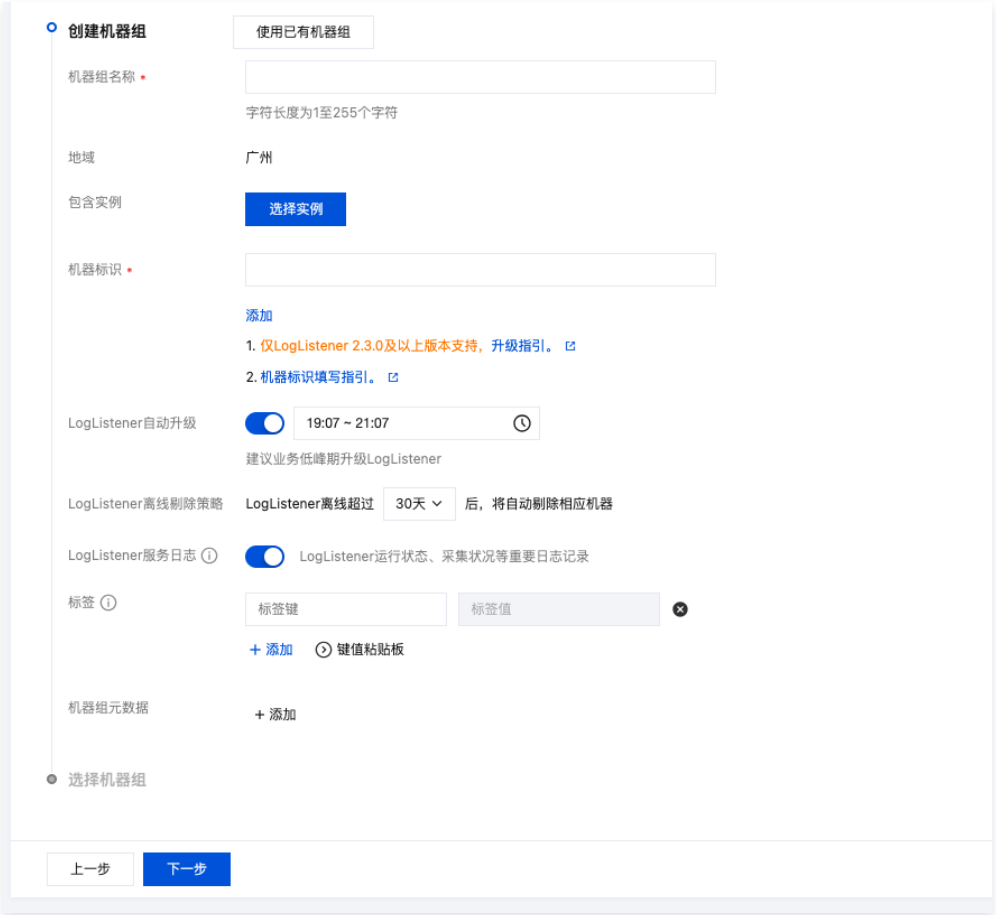
- 若您需要采集的目标服务器未安装 LogListener，可参见：
  - LogListener 安装指南（Linux 版）
  - LogListener 安装指南（Windows 版）
- 在机器组管理界面，您可以创建机器组，也可以选择使用已有机器组，



创建新的机器组

如果您想创建新的机器组，可执行如下操作：

1. 填写机器组名称，通过机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 机器组 。单击**下一步**。



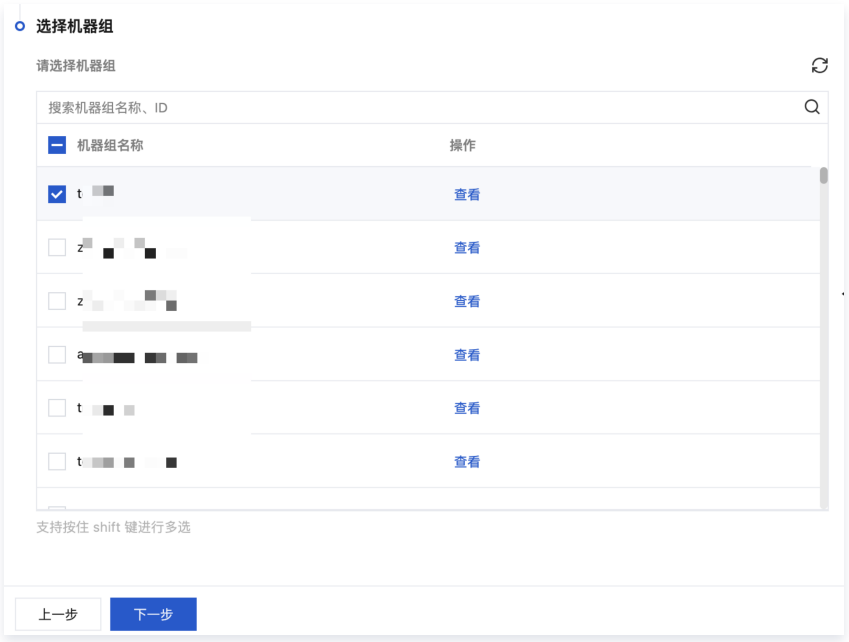
2. 完成创建后，在 Tab 选项中选择您创建的机器组的系统环境，并在列表中找到您创建的机器组。



选择已有的机器组

如果您想选择已有的机器组，可执行如下操作：

1. 单击**使用已有机器组**。
2. 在列表中勾选您的目标机器组，并单击**下一步**。



步骤3：采集配置

配置日志文件采集路径



在采集配置页面，填写采集规则名称，并根据日志采集路径格式填写采集路径。日志采集路径格式参考如下：

> ⚠ **注意：**
> Linux 日志路径必须以 `/` 开头。Windows 文件路径必须以盘符开头，如 `C:\` 。

- Linux 系统中的日志路径：`/[目录前缀表达式]/**/[文件名表达式]`，例如：`/data/log/**/*.log` 。
- Windows 系统中的日志路径：`[盘符]:\[目录前缀表达式]\**\[文件名表达式]`，例如：`C:\Program Files\Tencent\...\*.log` 。

填写日志采集路径后，LogListener 会按照[目录前缀表达式]匹配所有符合规则的公共前缀路径，并监听这些目录（包含子层目录）下所有符合[文件名表达式]规则的日志文件。其参数详细说明如下：

字段	说明
目录前缀	日志文件前缀目录结构，仅支持通配符 * 和 ？ • * 表示匹配多个任意字符 • ？表示匹配单个任意字符 不支持填写逗号
**	表示当前目录以及所有子目录
文件名	日志文件名，仅支持通配符 * 和 ？ • * 表示匹配多个任意字符 • ？表示匹配单个任意字符 不支持填写逗号

常用的配置模式如下：

- **[公共目录前缀]/**/[公共文件名前缀]***
- **[公共目录前缀]/**/*[公共文件名后缀]**
- **[公共目录前缀]/**/[公共文件名前缀]*[公共文件名后缀]**
- **[公共目录前缀]/**/*[公共字符串]***

填写示例如下：

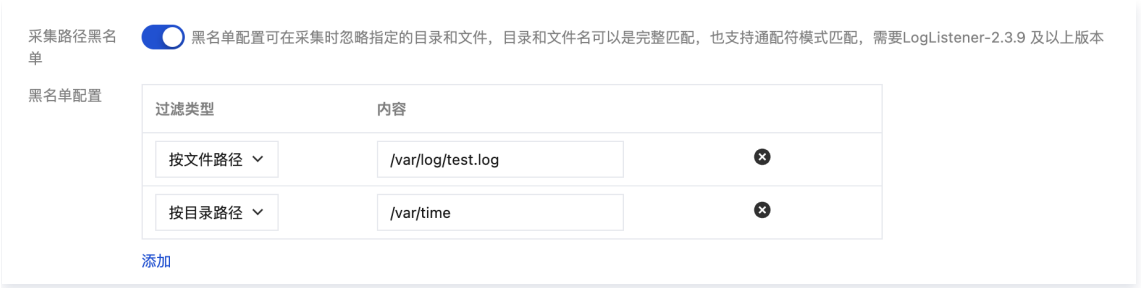
序号	目录前缀表达式	文件名表达式	说明
1	/var/log/nginx	access.log	此例中，日志路径配置为 `/var/log/nginx/**/access.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `access.log` 命名的日志文件
2	/var/log/nginx	*.log	此例中，日志路径配置为 `/var/log/nginx/**/*.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `.log` 结尾的日志文件
3	/var/log/nginx	error*	此例中，日志路径配置为 `/var/log/nginx/**/error*`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `error` 开头命名的日志文件

> ⚠ **注意：**
> - Windows 系统环境暂不支持软链采集。
> - LogListener 2.3.9及以上版本才可以添加多个采集路径。
> - 日志轮转场景下，建议配置采集路径为 `log/*.log`，并将日志轮转后的老文件命名为 `log/*.log.xxxx` 。

- 默认情况下，一个日志文件只能被一个日志主题采集。如果一个文件需要对应多个采集配置，且文件所在系统环境为 Linux，请给源文件添加一个软链接，并将其加到另一组采集配置中。

配置采集路径黑名单

开启采集路径黑名单，可在采集时忽略指定的目录前缀或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



采集黑名单分为两类过滤类型，且可以同时使用：

- 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配*或？，支持**路径模糊匹配。
- 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配*或？，支持**路径模糊匹配。

注意：

- 需要 LogListener 2.3.9及以上版本。
- 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

配置采集策略

- 全量采集：LogListener 采集文件时，从文件的开头开始读。
- 增量采集：LogListener 采集文件时，只采集文件内新增的内容。

配置回溯采集



当采集策略选择为增量采集时，您可以进一步在此设置回溯采集的起始点，指定 LogListener 启动时是否从最新位置往前偏移指定字节数开始采集。

注意：

Windows 系统环境暂不支持。

配置编码模式

- UTF-8：若您的日志文件编码模式为 UTF-8，请选择该选项。
- GBK：若您的日志文件编码模式为 GBK，请选择该选项。

配置单行全文格式

在“采集配置”页面，将“提取模式”设置为**单行全文**。如下图所示：



配置自定义元数据

您可配置自定义元数据，用来区分日志。当前支持以下几种元数据配置。详情请参见 自定义元数据 。

- 机器组元数据：使用机器组元数据。
- 采集路径：通过正则提取采集路径中的值作为元数据。
- 自定义：自定义键值作为元数据。

> **注意：**
> 需要 LogListener-2.8.7及以上版本。

配置过滤器条件

过滤器的目的是根据业务需要添加日志采集过滤规则，以帮助您筛选出有价值的日志数据。
单行全文模式下，默认使用 `__CONTENT__` 作为全文的键（key）名， 并支持以下过滤规则：

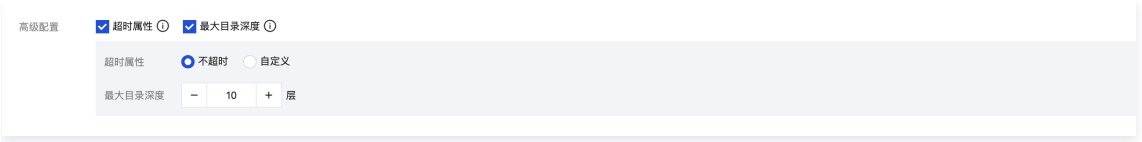
- 等于：仅采集指定字段值匹配指定字符的日志。 支持完全匹配，或正则匹配。
- 不等于：仅采集指定字段值不匹配指定字符的日志。支持完全匹配，或正则匹配。

例如，单行全文日志样例格式为 `Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64` ，您希望采集1月22号这一天的所有日志，则 key 处填写 `__CONTENT__` ，过滤规则配置选择**等于**，value 处填写 `Tue Jan 22.*` 。

> **注意：**
> - Windows 系统环境暂不支持。
> - 过滤规则 “不等于”、“字段存在”以及“字段不存在” 仅在 LogListener 2.9.3及以上版本支持。
> - 多条过滤条件之间关系是"与"逻辑，若同一 key 名配置多条过滤条件，规则会被覆盖。

高级配置

通过**勾选**，选择您需要定义的高级配置：



单行全文模式下，支持配置以下高级配置：

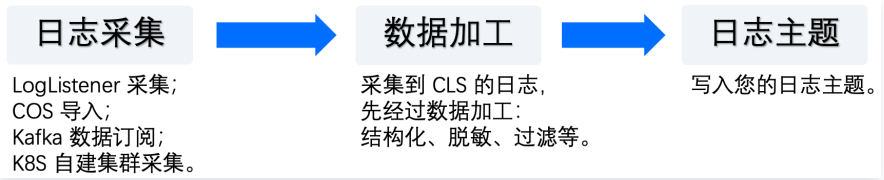
名称	描述	配置项

超时属性	该配置控制日志文件的超时时间。如果一个日志文件在指定时间内没有任何更新，则为超时。超时的日志文件 LogListener 将不再采集。当您的日志文件数量较大时，建议降低超时时间，避免 LogListener 性能浪费	• 不超时：日志文件永不超时 • 自定义：自定义日志文件的超时时间
最大目录深度	该配置控制日志采集的最大目录深度。LogListener 不会采集所在目录层级超过指定最大目录深度的日志文件。当您目标采集路径包含模糊匹配时，建议配置合适的最大目录深度，避免 LogListener 性能浪费。	大于0的整数。0代表不进行子目录的下钻

注意：
Windows 系统环境暂不支持。

数据加工

如您需要对采集到的 CLS 日志进一步处理，如结构化、脱敏、过滤，然后再写入日志主题，您可点击**采集配置**页面下方的**数据加工**，添加数据加工，最后 配置索引 。



说明：
- 数据加工相关操作请参见 创建加工任务 文档中【前置数据加工】TAB 页面；
- 数据加工脚本的编写请参见 数据加工函数总览，或者 加工实战案例；
- 数据加工会产生费用，详情请参见 计费概述 。

步骤4：索引配置

1. 单击**下一步**，进入索引配置页面。
2. 在**索引配置**页面，设置如下信息。配置详情请参见 索引配置 。



注意：
检索必须开启索引配置，否则无法检索。

3. 单击**提交**，进入编辑索引配置确认页面，单击**确认**。



如果您设置的索引配置仅需对新写入的日志生效，单击**确认**。如果您想对历史数据生效配置，单击**确认**后，详情请参见 重建索引，做进一步设置。

4. 操作成功，完成采集配置。

相关操作

检索日志，请参见 检索分析 文档。

使用多行全文提取模式采集日志

最近更新时间：2025-07-03 17:35:32

操作场景

多行全文日志是指一条完整的日志数据跨占多行（例如 Java 程序日志）。在这种情况下，以换行符 `\n` 为日志的结束标识符就显得有些不合理，为了能让日志系统明确区分开每条日志，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。
使用多行全文提取模式采集日志也会设置一个默认的键值 `__CONTENT__` 用于存放多行日志，且日志数据本身不进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。本文为您介绍如何使用多行全文提取模式采集日志。

前提条件

- 目标文件所在服务器已安装 LogListener。详情请参见：
  - LogListener 安装指南（Linux 版）
  - LogListener 安装指南（Windows 版）
- LogListener Linux 版本为2.1.1及以上 或 LogListener Windows 版本为2.9.7及以上。

效果预览

假设您的一条多行日志原始数据为：

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
    at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
    at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

该条日志最终被日志服务结构化处理为：

```
__CONTENT__:2019-12-15 17:13:06,043 [main] ERROR
com.test.logging.FooFactory:\njava.lang.NullPointerException\n    at
com.test.logging.FooFactory.createFoo(FooFactory.java:15)\n    at
com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

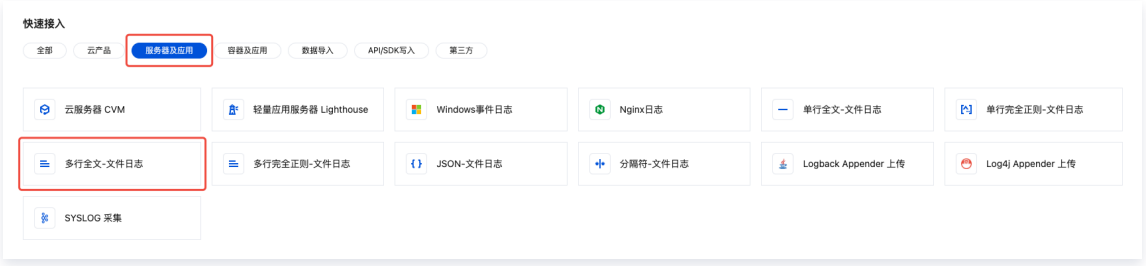
操作步骤

步骤1：创建/选择日志主题

创建新的日志主题

如果您想选择新的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**概览**，进入概览页面。
3. 在快速接入中，选择**服务器及应用**，并单击**多行全文-文件日志**，进入采集配置流程。



4. 在采集配置流程的**创建日志主题**页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步** 。



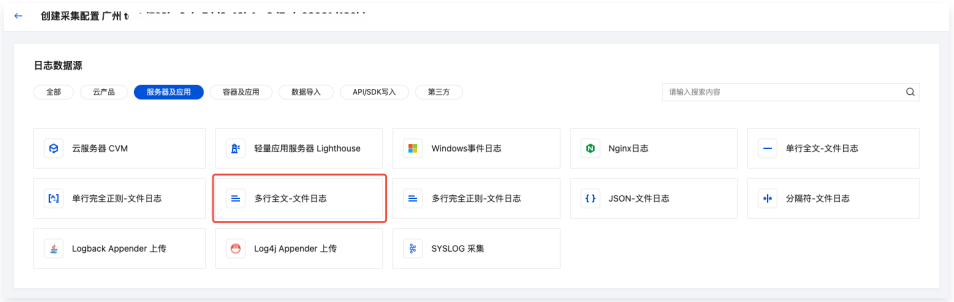
使用现有的日志主题

如果您想选择现有的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，选择需要投递的目标日志主题，单击日志主题名称，进入日志主题管理页面。
3. 选择**采集配置**页签，在 **LogListener 采集配置**栏下单击**新增**，进入日志数据源选择。



4. 在日志数据源选择页面中，选择**服务器及应用**，找到并单击**多行全文-文件日志**，进入采集配置流程。



步骤2：管理机器组多行全文-文件日志（以从概览页创建/选择日志主题为例）

完成日志主题新建或选择之后，单击**下一步**进入采集配置流程 – 机器组管理页面。



若您需要采集的目标服务器未安装 LogListener，可参见：

- LogListener 安装指南（Linux 版）
- LogListener 安装指南（Windows 版）

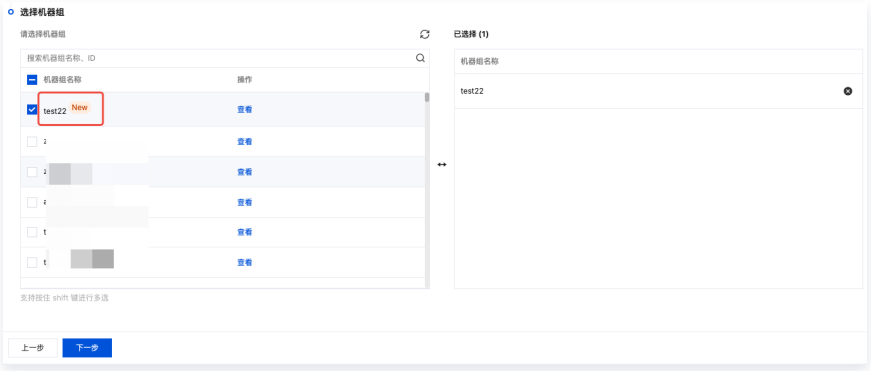
创建新的机器组

如果您想创建新的机器组，可执行如下操作：

1. 填写机器组名称，通过机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 机器组 。单击**下一步**。



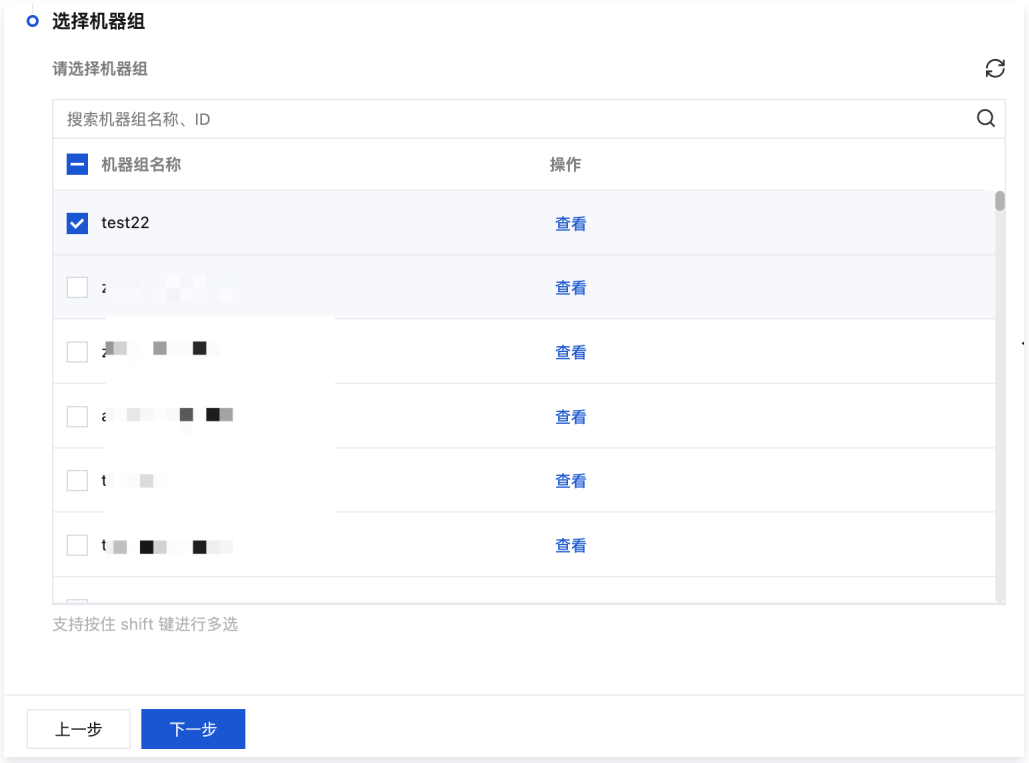
2. 完成创建后，在列表中勾选您创建的机器组。



选择已有的机器组

如果您想选择已有的机器组，可执行如下操作：

1. 单击**使用已有机器组**。
2. 在列表中勾选您的目标机器组，并单击**下一步**。



步骤3：采集配置

配置日志文件采集路径



在**采集配置**页面，填写采集规则名称，并根据日志采集路径格式填写“采集路径”。日志采集路径格式参考如下：

> **注意：**
> Linux 日志路径必须以 `/` 开头。Windows 文件路径必须以盘符开头，如 `C:\` 。

- Linux 系统中的日志路径：`/[目录前缀表达式]/**/[文件名表达式]`，例如：`/data/log/**/*.log` 。
- Windows 系统中的日志路径：`[盘符]:\[目录前缀表达式]\**\[文件名表达式]`，例如：`C:\Program Files\Tencent\...\*.log` 。

填写日志采集路径后，LogListener 会按照**[目录前缀表达式]**匹配所有符合规则的公共前缀路径，并监听这些目录（包含子层目录）下所有符合**[文件名表达式]**规则的日志文件。其参数详细说明如下：

字段	说明
目录前缀	日志文件前缀目录结构，仅支持通配符 * 和 ？ • * 表示匹配多个任意字符 • ？表示匹配单个任意字符

	不支持填写逗号
**	表示当前目录以及所有子目录
文件名	日志文件名，仅支持通配符 * 和 ？ • * 表示匹配多个任意字符 • ？表示匹配单个任意字符 不支持填写逗号

常用的配置模式如下：

- [公共目录前缀]/**/[公共文件名前缀]*
- [公共目录前缀]/**/*[公共文件名后缀]
- [公共目录前缀]/**/[公共文件名前缀]*[公共文件名后缀]
- [公共目录前缀]/**/*[公共字符串]*

填写示例如下：

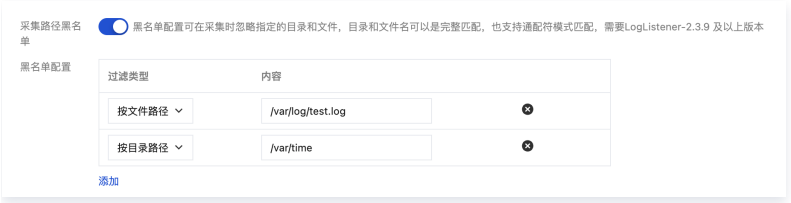
序号	目录前缀表达式	文件名表达式	说明
1	/var/log/nginx	access.log	此例中，日志路径配置为 `/var/log/nginx/**/access.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `access.log` 命名的日志文件
2	/var/log/nginx	*.log	此例中，日志路径配置为 `/var/log/nginx/**/*.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `.log` 结尾的日志文件
3	/var/log/nginx	error*	此例中，日志路径配置为 `/var/log/nginx/**/error*`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `error` 开头命名的日志文件

注意：

- Windows 系统环境暂不支持软链采集。
- LogListener 2.3.9及以上版本才可以添加多个采集路径。
- 建议配置采集路径为 `log/*.log`，rename 日志轮转后的老文件命名为 `log/*.log.xxxx`。
- 默认情况下，一个日志文件只能被一个日志主题采集。如果一个文件需要对应多个采集配置，且文件所在系统环境为 Linux，请给源文件添加一个软链接，并将其加到另一组采集配置中。

配置采集路径黑名单

开启采集路径黑名单，可在采集时忽略指定的目录前缀或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



采集黑名单分为两类过滤类型，且可以同时使用：

- 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配*或？，支持**路径模糊匹配。
- 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配*或？，支持**路径模糊匹配。

注意：

- 需要 LogListener 2.3.9及以上版本。
- 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

配置采集策略

- 全量采集：LogListener 采集文件时，从文件的开头开始读。
- 增量采集：LogListener 采集文件时，只采集文件内新增的内容。

配置回溯采集



当采集策略选择为增量采集时，您可以进一步在此设置回溯采集的起始点，指定 LogListener 启动时是否从最新位置往前偏移指定字节数开始采集。

> **注意：**
> Windows 系统环境暂不支持。

编码模式

- UTF-8：若您的日志文件编码模式为 UTF-8，请选择该选项。
- GBK：若您的日志文件编码模式为 GBK，请选择该选项。

配置多行全文格式

1. 在“采集配置”页面，将“提取模式”设置为**多行全文**。如下图所示：



2. 根据如下规则，定义正则表达式。
   您可选择**自动生成**或者**手动输入**两种方式定义首行正则表达式，系统会根据样例内容进行正则表达式的验证。

自动生成

在文本框中，输入日志样例，单击**自动生成**，系统自动在置灰的文本框中生成行首正则表达式。如下图所示：



手动输入

在文本框中，输入日志样例，输入首行正则表达式，单击**验证**，系统将判断表达式是否通过。如下图所示：



配置自定义元数据

> **注意：**
> 仅LogListener 2.8.7及以上版本支持配置自定义元数据。

您可配置自定义元数据，可以用来区分日志。当前支持以下几种元数据配置。详情请参见 自定义元数据 。

- 机器组元数据：使用机器组元数据

- 采集路径：通过正则提取采集路径中的值作为元数据
- 自定义：自定义键值作为元数据

配置过滤器条件

过滤器的目的是根据业务需要添加日志采集过滤规则，以帮助您筛选出有价值的日志数据。
多行全文模式下，默认使用 `__CONTENT__` 作为全文的键（key）名，并支持以下过滤规则：

- 等于：仅采集指定字段值匹配指定字符的日志。支持完全匹配，或正则匹配。
- 不等于：仅采集指定字段值不匹配指定字符的日志。支持完全匹配，或正则匹配。

例如，多行全文日志样例格式为：

```
10.20.20.10 - - [Tue Jan 22 14:24:03 CST 2019 +0800] GET /online/sample HTTP/1.1 127.0.0.1 200 628 35
http://127.0.0.1/group/1
Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0 0.310 0.310
```

您希望采集 `10.20.20.10` 这台机器的所有日志，则 key 处填写 `__CONTENT__`，过滤规则选择**等于**，value 处填写 `10.20.20.10.*`。

> **注意：**
> - Windows 系统环境暂不支持。
> - 过滤规则 “不等于”、“字段存在”以及“字段不存在” 仅在 LogListener 2.9.3及以上版本支持。
> - 多条过滤条件之间关系是"与"逻辑，若同一 key 名配置多条过滤条件，规则会被覆盖。

配置上传解析失败日志

建议开启上传解析失败日志。开启后，LogListener 会上传各式解析失败的日志。若关闭上传解析失败日志，则会丢弃失败的日志。

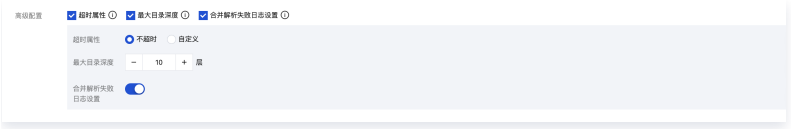


开启后需要配置解析失败的 Key 值（默认为 LogParseFailure），所有解析失败的日志，均以输入内容作为键名称（Key），原始日志内容作为值（Value）进行上传。

高级配置

> **注意：**
> Windows 系统环境暂不支持。

通过勾选，选择您需要定义的高级配置：



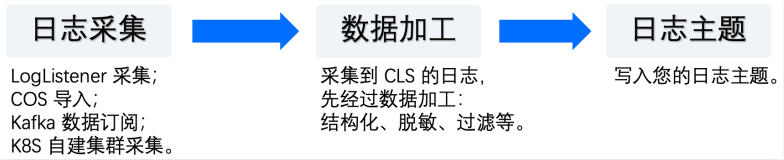
多行全文提取模式下，支持配置以下高级配置：

名称	描述	配置项
超时属性	该配置控制日志文件的超时时间。如果一个日志文件在指定时间内没有任何更新，则为超时。超时的日志文件 LogListener 将不再采集。当您的日志文件数量较大时，建议降低超时时间，避免 LogListener 性能浪费	• 不超时：日志文件永不超时 • 自定义：自定义日志文件的超时时间
最大目录深度	该配置控制日志采集的最大目录深度。LogListener 不会采集所在目录层级超过指定最大目录深度的日志文件。当您目标采集路径包含模糊匹配时，建议配置合适的最大目录深度，避免 LogListener 性能浪费。	大于0的整数。0代表不进行子目录的下钻
合并解析失败日志	> **注意：**	开启/关闭

	LogListener 2.8.8及以上版本才可以配置合并解析失败日志。 该配置支持在采集时将目标日志文件中连续解析失败的日志合并为一条日志上传。 若您的首行正则表达式无法覆盖所有的多行日志， 建议开启该配置，避免因首行匹配失败的单条多行日志被拆分至多条日志	

数据加工

如您需要对采集到的 CLS 日志进一步处理，如结构化、脱敏、过滤，然后再写入日志主题，您可点击**采集配置**页面下方的**数据加工**，添加数据加工，最后 配置索引 。



说明：

- 数据加工相关操作请参见 创建加工任务 文档中【前置数据加工】TAB 页面。
- 数据加工脚本的编写请参见 数据加工函数总览，或者 加工实战案例 。
- 数据加工会产生费用，详情请参见 计费概述 。

步骤4：索引配置

1. 单击**下一步**，进入**索引配置**页面。
2. 在“索引配置”页面，设置如下信息。配置详情请参见 索引配置 。



注意

检索必须开启索引配置，否则无法检索。

3. 单击**提交**，进入编辑索引配置确认页面。



如果您设置的索引配置仅需对新写入的日志生效，单击**确认**。如果您想对历史数据生效配置，单击**确认**后，详情请参见 重建索引，做进一步设置。

4. 操作成功，完成采集配置。

相关操作

检索日志，请参见 检索分析 文档。

使用单行-完全正则提取模式采集日志

最近更新时间：2025-07-03 17:35:32

操作场景

单行-完全正则模式适用于日志文本中每行内容为一条原始日志，且每条日志可按正则表达式提取为多个 key-value 键值的日志解析模式。若不需要提取 key-value，请参见 单行全文格式 进行配置。
配置单行-完全正则模式时，您需要先输入日志样例，再自定义正则表达式。配置完成后，系统将根据正则表达式中的捕获组提取对应的 key-value。
本文将为您详细介绍如何采集单行-完全正则模式日志。

前提条件

- 目标文件所在服务器已安装 LogListener。详情请参见：
  - LogListener 安装指南（Linux 版）
  - LogListener 安装指南（Windows 版）
- LogListener Linux 版本为2.2.2及以上或 LogListener Windows 版本为2.9.7及以上。

效果预览

假设您的一条日志原始数据为：

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703
"http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum" "Mozilla/5.0
(Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0"  0.354 0.354
```

配置的自定义正则表达式为：

```
(\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"([^"]+)"\s"
([^"]+)"\s+(\S+)\s(\S+).*
```

系统根据 `()` 捕获组提取对应的 key-value 后，您可以自定义每组的 key 名称如下所示：

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

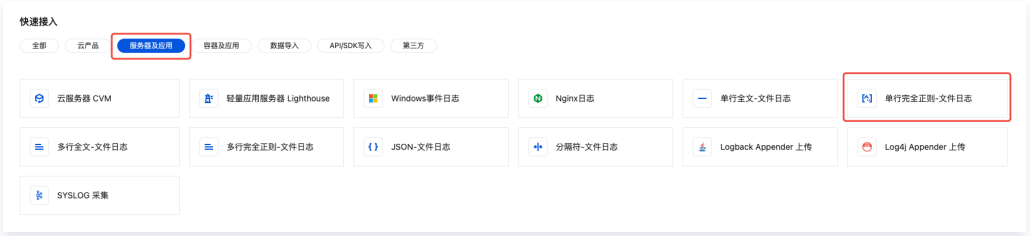
操作步骤

步骤1：创建/选择日志主题

创建新的日志主题

如果您想创建新的日志主题，可执行如下操作：
1. 登录 日志服务控制台 。

2. 在左侧导航栏中，单击**概览**，进入概览页面。
3. 在**快速接入**中，选择**服务器及应用**，找到并单击**单行完全正则-文件日志**，进入采集配置流程。



4. 在采集配置流程的**创建日志主题**页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步** 。



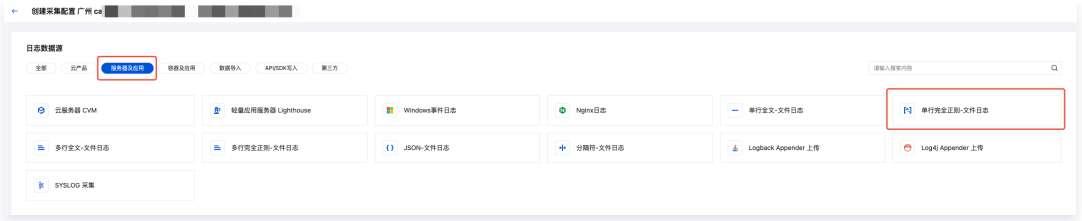
选择现有的日志主题

如果您想选择现有的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，选择需要投递的日志主题，单击日志主题名称，进入日志主题管理页面。
3. 选择**采集配置**页签，在 LogListener 采集配置栏下单击**新增**，进入日志数据源选择。



4. 在日志数据源选择页面中，选择**服务器及应用**，找到并单击**单行完全正则-文件日志**，进入采集配置流程。



步骤2：管理机器组（以从概览页创建/选择日志主题为例）

完成日志主题新建或选择之后，单击**下一步**进入采集配置流程 – 机器组管理页面。



若您需要采集的目标服务器未安装 LogListener， 可参见：

- LogListener 安装指南（Linux 版）
- LogListener 安装指南（Windows 版）

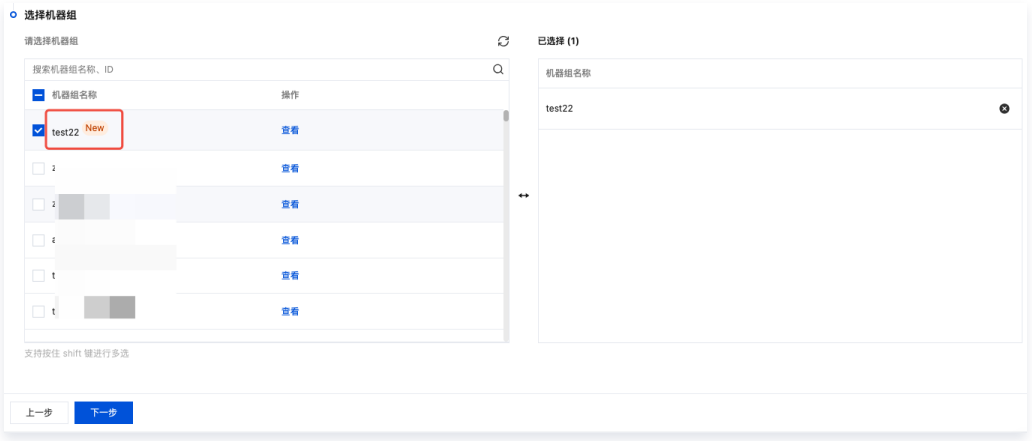
创建新的机器组

如果您想创建新的机器组，可执行如下操作：

1. 填写机器组名称，通过机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 机器组 。单击**下一步**。



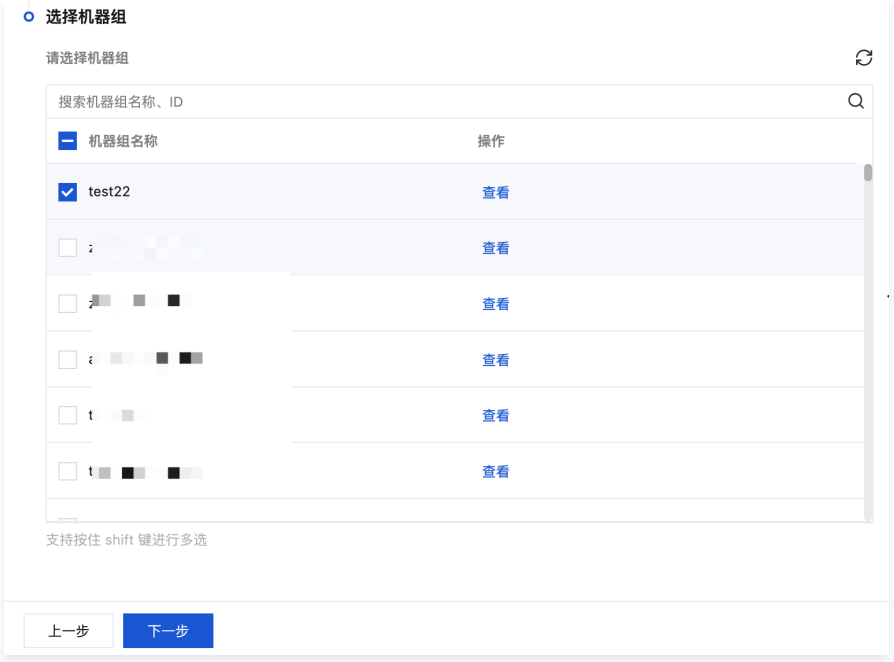
2. 完成创建后，在列表中勾选您创建的机器组。



选择已有的机器组

如果您想选择已有的机器组，可执行如下操作：

1. 单击**使用已有机器组**。
2. 在列表中勾选您的目标机器组，并单击**下一步**。



步骤3：采集配置

配置日志文件采集路径



在**采集配置**页面，填写采集规则名称，并根据日志采集路径格式填写**采集路径**。日志采集路径格式参考如下：

> **注意：**
> Linux日志路径必须以 `/` 开头。Windows文件路径必须以盘符开头，如 `C:\` 。

- Linux系统中的日志路径： `/[目录前缀表达式]/**/[文件名表达式]` ，例如： `/data/log/**/*.log` 。
- Windows系统中的日志路径 ： `[盘符]:\[目录前缀表达式]\**\[文件名表达式]` ，例如： `C:\Program Files\Tencent\...\*.log` 。

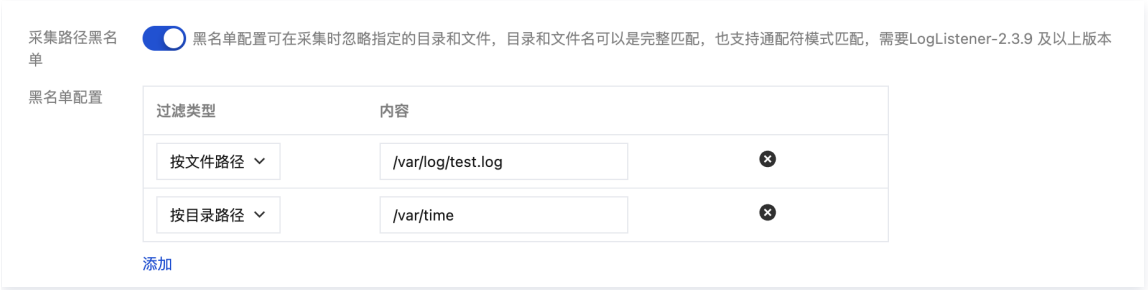
填写日志采集路径后，LogListener 会按照**[目录前缀表达式]**匹配所有符合规则的公共前缀路径，并监听这些目录（包含子层目录）下所有符合**[文件名表达式]**规则的日志文件。其参数详细说明如下：

序号	目录前缀表达式	文件名表达式	说明
1	/var/log/nginx	access.log	此例中，日志路径配置为 `/var/log/nginx/**/access.log` ，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `access.log` 命名的日志文件
2	/var/log/nginx	*.log	此例中，日志路径配置为 `/var/log/nginx/**/*.log` ，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `.log` 结尾的日志文件
3	/var/log/nginx	error*	此例中，日志路径配置为 `/var/log/nginx/**/error*` ，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `error` 开头命名的日志文件

> **注意：**
> - Windows 系统环境暂不支持软链采集。
> - LogListener 2.3.9及以上版本才可以添加多个采集路径。
> - 建议配置采集路径为 `log/*.log`，rename 日志轮转后的老文件命名为 `log/*.log.xxxx` 。
> - 默认情况下，一个日志文件只能被一个日志主题采集。如果一个文件需要对应多个采集配置，且文件所在系统环境为 Linux，请给源文件添加一个软链接，并将其加到另一组采集配置中。

配置采集路径黑名单

开启采集路径黑名单，可在采集时忽略指定的目录前缀或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



采集黑名单分为两类过滤类型，且可以同时使用：

- 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配*或？，支持**路径模糊匹配。
- 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配*或？，支持**路径模糊匹配。

> **注意：**
> - 需要 LogListener 2.3.9及以上版本。
> - 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

配置采集策略

- 全量采集：LogListener 采集文件时，从文件的开头开始读。
- 增量采集：LogListener 采集文件时，只采集文件内新增的内容。

配置回溯采集



当采集策略选择为增量采集时，您可以进一步在此设置回溯采集的起始点，指定 LogListener 启动时是否从最新位置往前偏移您指定字节数开始采集。

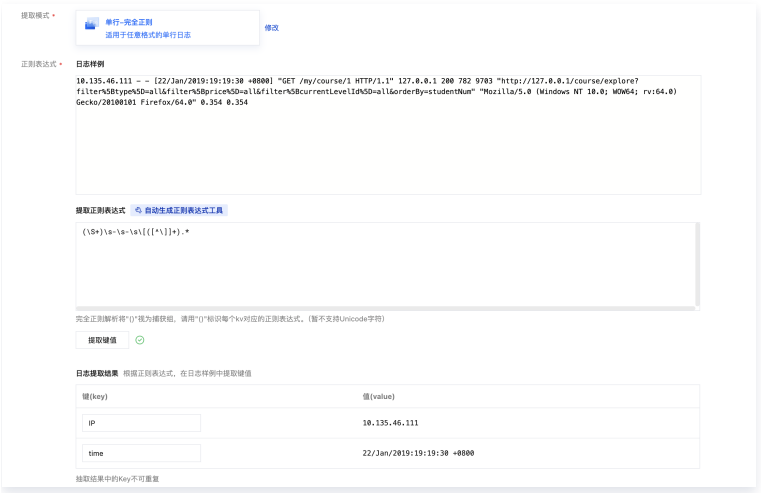
> **注意：**
> Windows 系统环境暂不支持。

编码模式

- UTF-8：若您的日志文件编码模式为 UTF-8，请选择该选项。
- GBK：若您的日志文件编码模式为 GBK，请选择该选项。

配置单行-完全正则模式

1. 在**采集配置**页面，将“提取模式”设置为**单行-完全正则**，并在“日志样例”文本框中，输入日志样例。如下图所示：

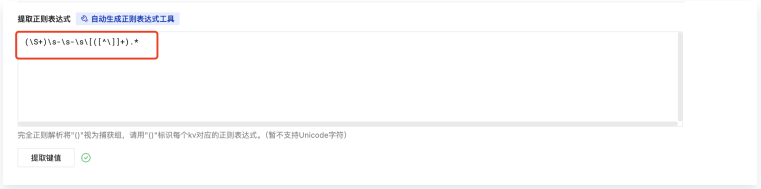


2. 根据如下规则，定义正则表达式：

   系统有**手动模式**和**自动模式**两种方式定义正则表达式。您可手动输入表达式提取 key-value 进行验证，也可单击**自动正则表达式工具**切换为自动模式。系统会根据您选择的模式以及定义好的正则表达式，提取 key-value 进行正则表达式的验证。

手动模式

1. 在“正则表达式”的文本框中，输入正则表达式。
2. 单击**验证**，系统将判断日志样例与正则表达式是否匹配。

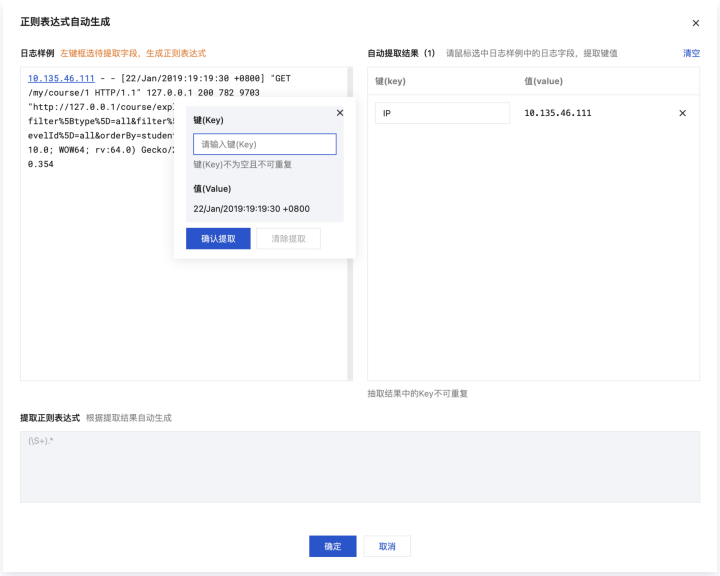


自动模式

1. 单击**自动正则表达式工具**。



2. 在弹出的“正则表达式自动生成”模态视图中，根据实际的检索分析需求，**鼠标左键**选中需要提取 key-value 的日志内容，并在弹出的文本框中，输入键(key)名，单击**确认提取**。如下图所示：



系统将自动对该部分内容提取一个正则表达式，**自动提取结果**会出现在右侧 key-value 表格中，根据**自动提取结果**自动生成的**提取正则表达式**会出现在下侧表格中。如下图所示：



3. 重复 **自动模式的步骤2**，直到提取完所有的 key-value 对。如下图所示：



4. 单击**确定**，系统将根据提取好的 key-value 对自动生成完整的正则表达式。

> **说明：**
> - 无论您使用手动模式还是自动模式，在完成定义并验证通过后，提取结果都将展示在“日志提取结果”中。若结果不符合预期，可返回 定义正则表达式 重新编辑，并再次验证。
> - 您只需定义每一组 key-value 对的 key 名称，即可将该名称用于日志检索分析。

更多日志验证

1. 当您有多条复杂日志样例数据时，可以将**匹配更多日志**设置为 ，即可验证您的正则表达式样例的通过率。

2. 输入待验证日志样例，以**换行符**分开多条数据，单击**验证**，系统将验证样通过率。出现 时，即为验证通过，如下图所示：



配置自定义元数据

您可配置自定义元数据，可以用来区分日志。当前支持以下几种元数据配置。详情请参见 自定义元数据 。
- 机器组元数据：使用机器组元数据
- 采集路径：通过正则提取采集路径中的值作为元数据
- 自定义：自定义键值作为元数据

> **注意：**
> 仅 LogListener 2.8.7及以上版本支持配置自定义元数据。

配置日志时间戳来源

您可选择以**日志采集时间**或者**指定日志字段**两种时间作为日志的时间戳：

日志采集时间

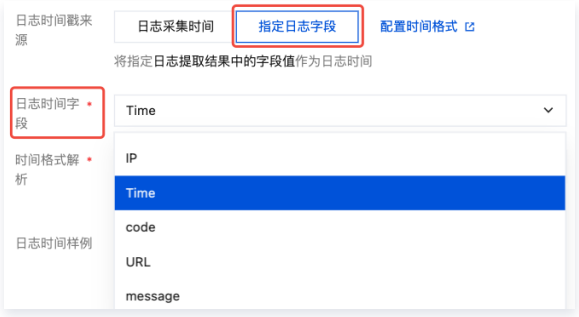
使用日志采集的时间作为日志时间。



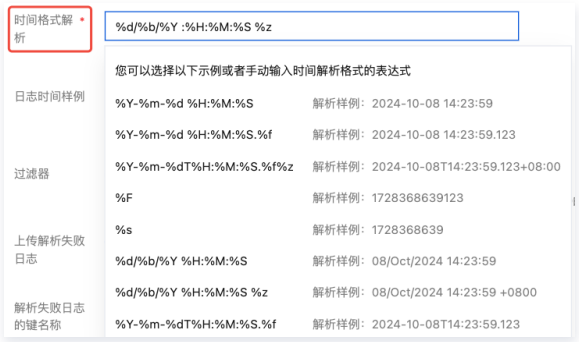
指定日志字段

使用日志中指定字段的值作为日志的时间。

1. 在**日志时间字段**中选择已经提取的 Value 值作为日志时间。



2. 在**时间格式解析**中手动输入或选择该值对应的解析表达式。例如：日志中代表时间的值为 07/Jul/2025:19:19:30 +0800，解析格式为：%d/%b/%Y :%H:%M:%S %z。更多请参见 配置时间格式 。



3. 单击**验证**。

> **注意：**
> 若时间格式填写错误，日志时间将以采集时间为准。

配置过滤器条件

过滤器的目的是根据业务需要添加日志采集过滤规则，以帮助您筛选出有价值的日志数据。
完全正则采集时，需要根据所自定义的键值对来配置过滤条件。支持以下过滤规则：

- 等于：仅采集指定字段值匹配指定字符的日志。支持完全匹配，或正则匹配。
- 不等于：仅采集指定字段值不匹配指定字符的日志。支持完全匹配，或正则匹配。
- 字段存在：仅采集指定字段存在的日志。
- 字段不存在：仅采集指定字段不存在的日志。

例如，样例日志使用完全正则模式解析后，您希望 status 字段为400或500的所有日志数据被采集，那么 key 处配置 status，过滤规则选择**等于**，value 处配置400|500。

> **注意：**
> - Windows 系统环境暂不支持。
> - 过滤规则 “不等于”、“字段存在”以及“字段不存在” 仅在 LogListener 2.9.3及以上版本支持。
> - 多条过滤条件之间关系是"与"逻辑，若同一 key 名配置多条过滤条件，规则会被覆盖。

配置上传解析失败日志

建议开启上传解析失败日志。开启后，LogListener 会上传各式解析失败的日志。若关闭上传解析失败日志，则会丢弃失败的日志。



开启后需要配置解析失败的 Key 值（默认为 LogParseFailure），所有解析失败的日志，均以输入内容作为键名称（Key），原始日志内容作为值（Value）进行上传。

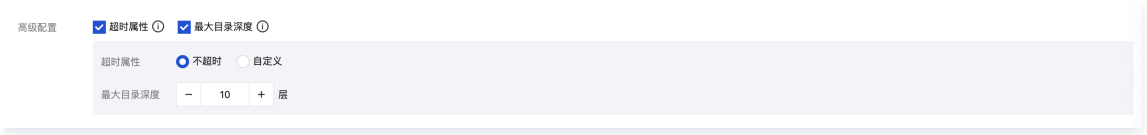
上传原始日志

开启后 LogListener 会将原始日志与解析后的日志一起上传。所有原始日志，均以您指定的键名称，原始日志内容将作为值（Value）进行上传。

高级配置

> **注意：**
> Windows 系统环境暂不支持。

通过勾选，选择您需要定义的高级配置。



单行-完全正则提取模式下，支持配置以下高级配置。

名称	描述	配置项
超时属性	该配置控制日志文件的超时时间。如果一个日志文件在指定时间内没有任何更新，则为超时。超时的日志文件 LogListener 将不再采集。当您的日志文件数量较大时，建议降低超时时间，避免 LogListener 性能浪费	• 不超时：日志文件永不超时 • 自定义：自定义日志文件的超时时间
最大目录深度	该配置控制日志采集的最大目录深度。LogListener 不会采集所在目录层级超过指定最大目录深度的日志文件。当您目标采集路径包含模糊匹配时，建议配置合适的最大目录深度，避免 LogListener 性能浪费。	大于0的整数。0代表不进行子目录的下钻

步骤4：索引配置

1. 单击**下一步**，进入“索引配置”页面。
2. 在“索引配置”页面，设置如下信息。配置详情请参见 索引配置 。



> **注意：**
> 检索必须开启索引配置，否则无法检索。

3. 单击**提交**，进入编辑索引配置确认页面。



如果您设置的索引配置仅需对新写入的日志生效，单击**确认**。如果您想对历史数据生效配置，单击**确认**后，详情请参见 重建索引，做进一步设置。

4. 操作成功，完成采集配置。

相关操作

检索日志，请参见 检索分析 文档。

使用多行-完全正则提取模式采集日志

最近更新时间：2025-07-03 17:35:32

操作场景

多行-完全正则模式适用于日志文本中一条完整的日志数据跨占多行（例如 Java 程序日志），且可按正则表达式提取为多个 key-value 键值的日志解析模式。
若不需要提取 key-value，请参见 多行全文格式 进行配置。
配置多行-完全正则模式时，您需要先输入日志样例，再自定义正则表达式。配置完成后，系统将根据正则表达式中的捕获组提取对应的 key-value。
本文将为您详细介绍如何采集多行-完全正则模式日志。

前提条件

- 目标文件所在服务器已安装 LogListener。详情请参见：
  - LogListener 安装指南（Linux 版）
  - LogListener 安装指南（Windows 版）
- LogListener Linux 版本为2.4.5以上或 LogListener Windows 版本为2.9.7及以上。

效果预览

假设您的一条日志原始数据为：

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

行首正则表达式为：

```
\[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*
```

配置的自定义正则表达式为：

```
\[(\d+-\d+-\w+:\d+:\d+,\d+)\]\s\[(\w+)\]\s(.*)
```

系统根据 `()` 捕获组提取对应的 key-value 后，您可以自定义每组的 key 名称如下所示：

```
time: 2018-10-01T10:30:01,000`
level: INFO`
msg: java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

操作步骤

步骤1：创建/选择日志主题

创建新的日志主题

如果您想创建新的日志主题，可执行如下操作：

1. 登录 日志服务控制台。
2. 在左侧导航栏中，单击**概览**，进入概览页面。
3. 在快速接入中，选择**服务器及应用**，找到并单击**多行完全正则-文件日志**，进入采集配置流程。



4. 在采集配置流程的**创建日志主题**页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步** 。



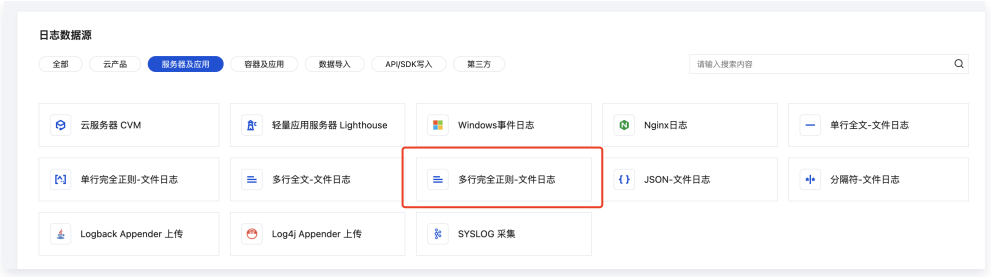
选择现有的日志主题

如果您想选择现有的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，选择需要投递的日志主题，单击日志主题名称，进入日志主题管理页面。
3. 选择**采集配置**页签，在 LogListener 采集配置栏下单击**新增**，进入日志数据源选择。



4. 在日志数据源选择页面中，选择**服务器及应用**，找到并单击**多行完全正则-文件日志**，进入采集配置流程。



步骤2：管理机器组（以从概览页创建/选择日志主题为例）

完成日志主题新建或选择之后，单击**下一步**进入采集配置流程 – 机器组管理页面。



若您需要采集的目标服务器未安装 LogListener， 可参见：

- LogListener 安装指南（Linux 版）
- LogListener 安装指南（Windows 版）

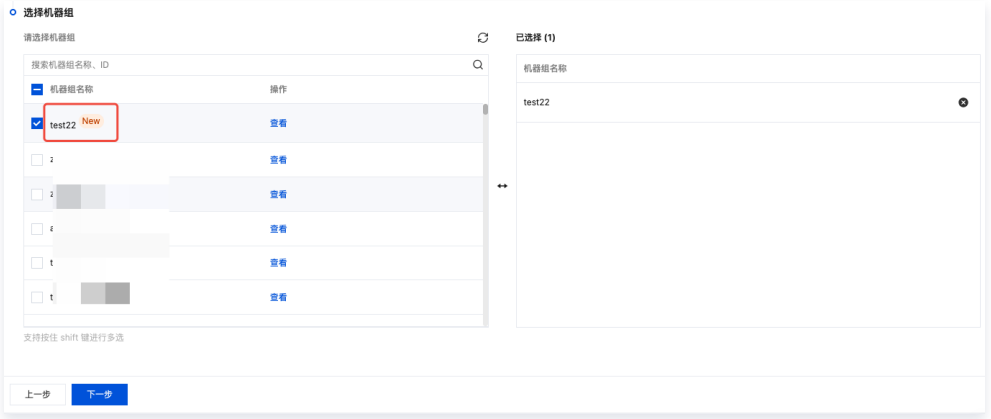
创建新的机器组

如果您想创建新的机器组，可执行如下操作：

1. 填写机器组名称，通过机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 机器组 。单击**下一步**。



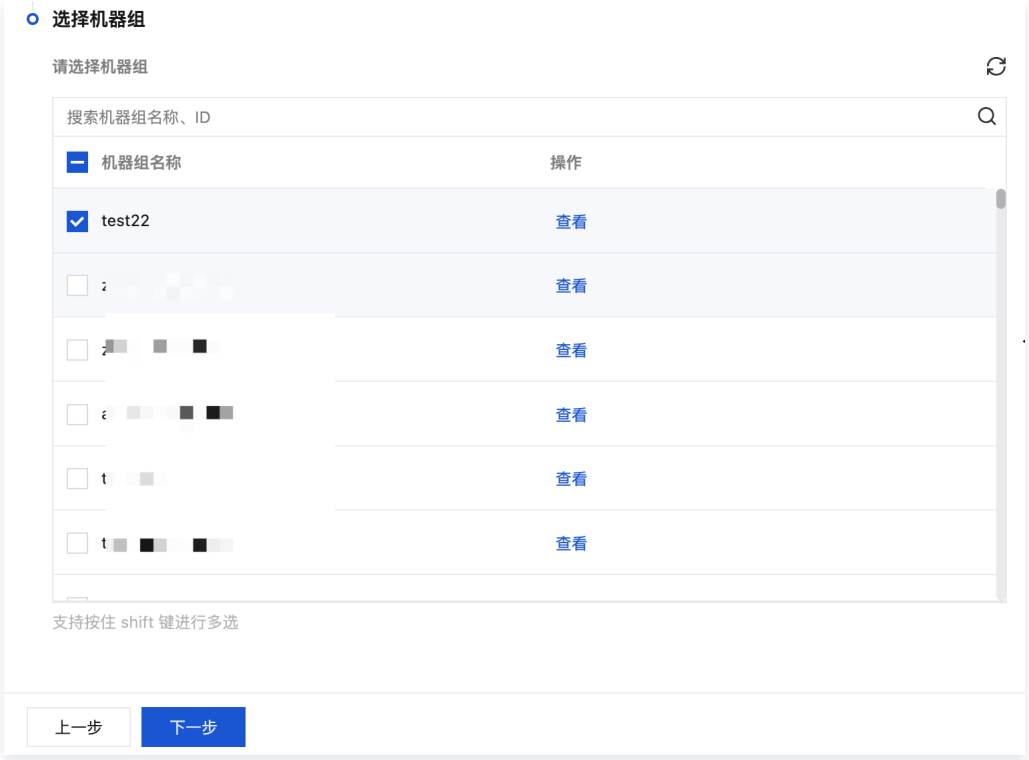
2. 完成创建后，在列表中勾选您创建的机器组。



选择已有的机器组

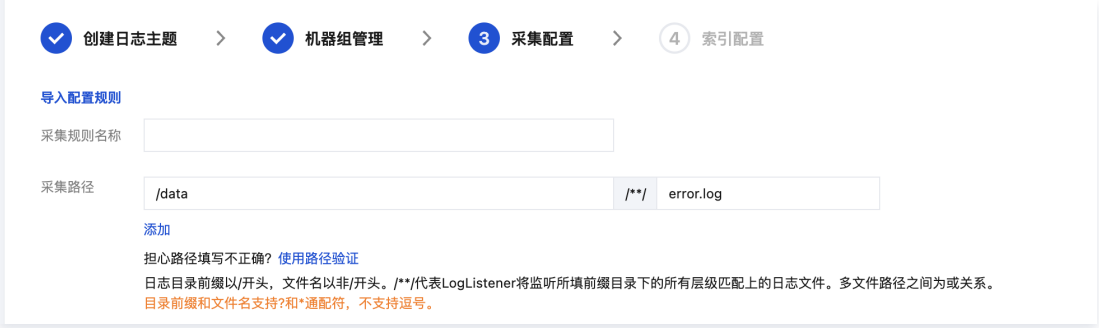
如果您想选择已有的机器组，可执行如下操作：

1. 单击**使用已有机器组**。
2. 在列表中勾选您的目标机器组，并单击**下一步**。



步骤3：采集配置

配置日志文件采集路径



在“采集配置”页面，填写采集规则名称，并根据日志采集路径格式填写“采集路径”。日志采集路径格式参考如下：

> **注意：**
> Linux日志路径必须以 `/` 开头。Windows文件路径必须以盘符开头，如 `C:\` 。

- Linux系统中的日志路径：`/[目录前缀表达式]/**/[文件名表达式]`，例如：`/data/log/**/*.log` 。
- Windows系统中的日志路径：`[盘符]:\[目录前缀表达式]\**\[文件名表达式]`，例如：`C:\Program Files\Tencent\...\*.log` 。

填写日志采集路径后，LogListener 会按照**[目录前缀表达式]**匹配所有符合规则的公共前缀路径，并监听这些目录（包含子层目录）下所有符合**[文件名表达式]**规则的日志文件。其参数详细说明如下：

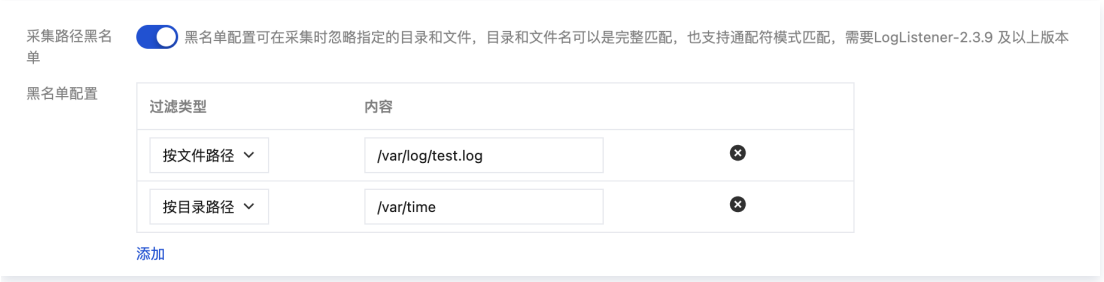
序号	目录前缀表达式	文件名表达式	说明
1	/var/log/nginx	access.log	此例中，日志路径配置为 `/var/log/nginx/**/access.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `access.log` 命名的日志文件
2	/var/log/nginx	*.log	此例中，日志路径配置为 `/var/log/nginx/**/*.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `.log` 结尾的日志文件

3	/var/log/nginx	error*	此例中，日志路径配置为 `/var/log/nginx/**/error*`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `error` 开头命名的日志文件

注意：
- Windows 系统环境暂不支持软链采集。
- LogListener 2.3.9及以上版本才可以添加多个采集路径。
- 建议配置采集路径为 `log/*.log`，rename 日志轮转后的老文件命名为 `log/*.log.xxxx`。
- 默认情况下，一个日志文件只能被一个日志主题采集。如果一个文件需要对应多个采集配置，且文件所在系统环境为 Linux，请给源文件添加一个软链接，并将其加到另一组采集配置中。

配置采集路径黑名单

开启采集路径黑名单，可在采集时忽略指定的目录前缀或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



采集黑名单分为两类过滤类型，且可以同时使用：
- 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配*或？，支持**路径模糊匹配。
- 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配*或？，支持**路径模糊匹配。

注意：
- 需要 LogListener 2.3.9及以上版本。
- 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

配置采集策略

- 全量采集：LogListener 采集文件时，从文件的开头开始读。
- 增量采集：LogListener 采集文件时，只采集文件内新增的内容。

配置回溯采集



当采集策略选择为增量采集时，您可以进一步在此设置回溯采集的起始点，指定 LogListener 启动时是否从最新位置往前偏移指定字节数开始采集。

注意：
Windows 系统环境暂不支持。

编码模式

- UTF-8：若您的日志文件编码模式为 UTF-8，请选择该选项。
- GBK：若您的日志文件编码模式为 GBK，请选择该选项。

配置多行-完全正则模式

1. 在采集配置页面，将“提取模式”设置为**多行-完全正则**，并在“日志样例”文本框中，输入日志样例。如下图所示：



2. 根据如下规则，定义首行正则表达式：
3. 您可选择是否开启**自动生成**方式定义行首正则表达式（不开启可以手动输入），确定跨行日志边界。待表达式验证成功后，系统会为您判断行首正则表达式匹配的日志条数。
   - 自动生成：开启**自动生成**，系统自动在置灰的文本框中生成行首正则表达式。如下图所示：



   - 手动输入：在文本框中，手动输入行首正则表达式，单击**验证**，系统将判断表达式是否通过。如下图所示：



4. 提取正则表达式。

   系统有**手动模式**和**自动模式**两种方式定义正则表达式。您可手动输入表达式提取 key-value 进行验证，也可单击**正则表达式自动生成**切换为自动模式。系统会根据您选择的模式以及定义好的正则表达式，提取 key-value 进行正则表达式的验证。

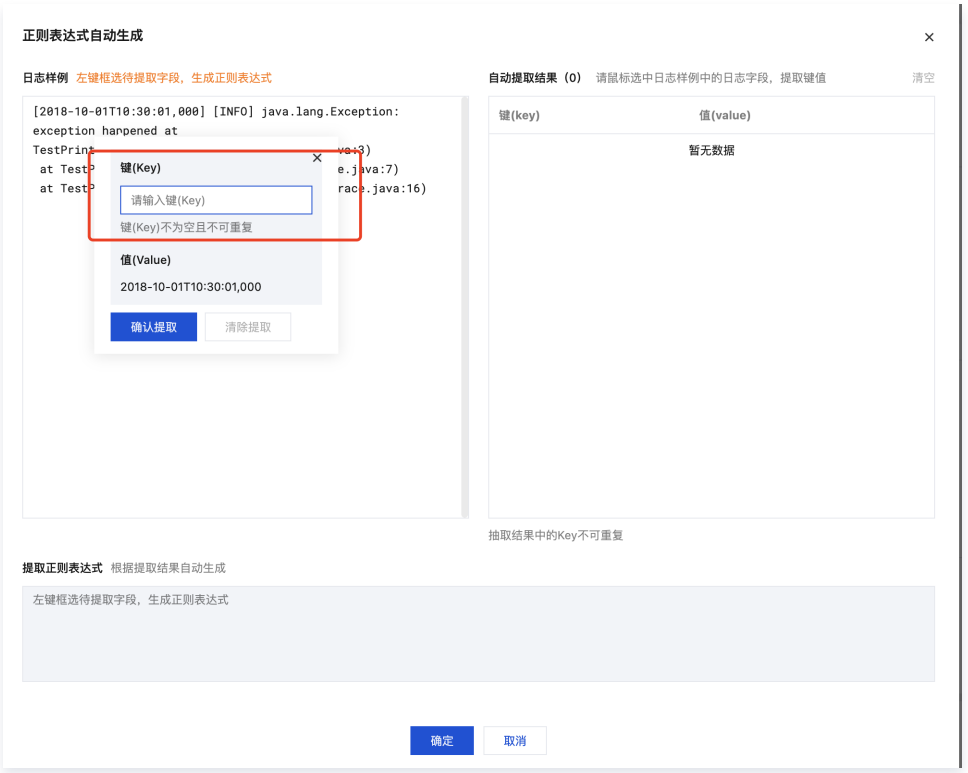
手动模式

1. 在“提取正则表达式”的文本框中，输入正则表达式。
2. 单击**验证**，系统将判断日志样例与正则表达式是否匹配。

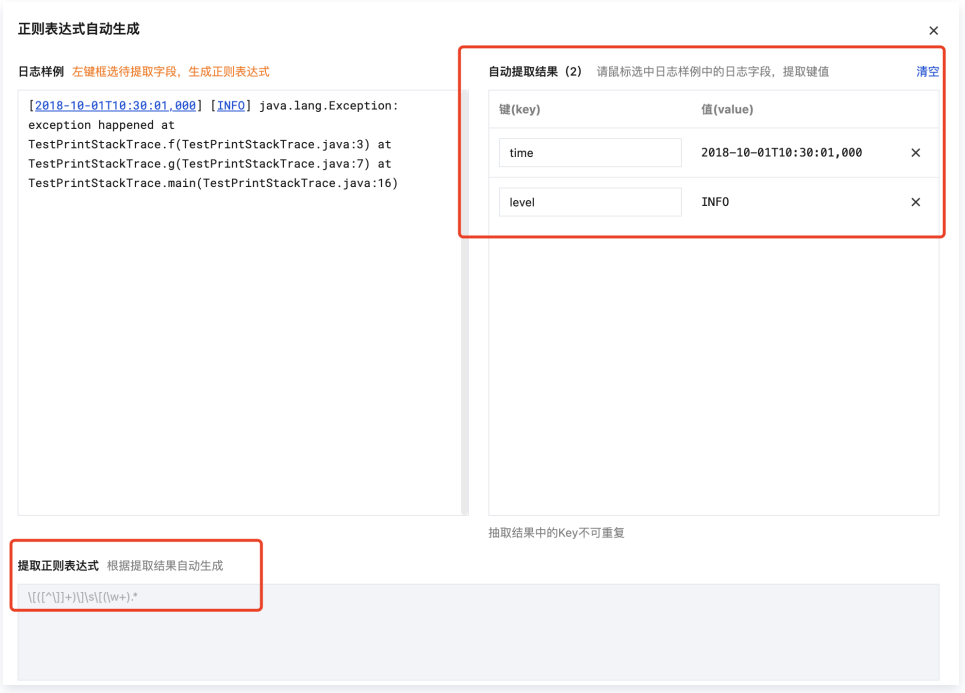


自动模式

1. 在弹出的“正则表达式自动生成”模态视图中，根据实际的检索分析需求，选中需要提取 key-value 的日志内容，并在弹出的文本框中，输入键(key)名，单击**确认提取**。如下图所示：



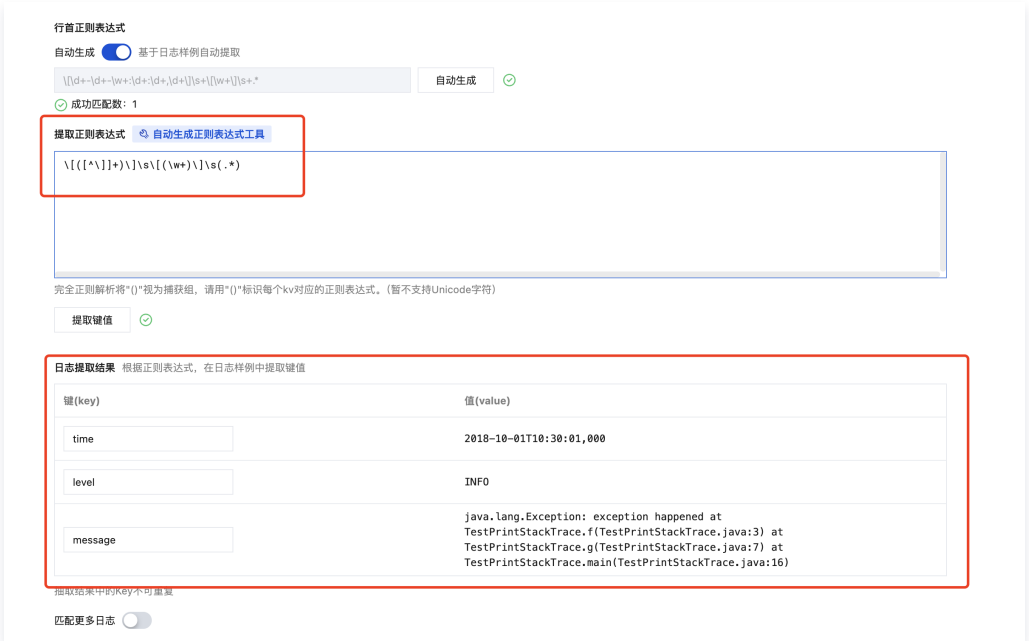
2. 系统将自动对该部分内容提取一个正则表达式，**自动提取结果**会出现在 key-value 表格中。如下图所示：



3. 重复 自动模式的步骤1，直到提取完所有的 key-value 对。如下图所示：



4. 单击**确定**，系统将根据提取好的 key-value 对自动生成完整的正则表达式。如下图所示：



> **说明：**
> 无论选择自动模式还是手动模式，正则提取模式均在完成定义并验证通过后，将提取结果展示在“抽取结果”中。您只需定义每一组 key-value 对的 key 名称，即可将该名称用于日志检索分析。

匹配更多日志

1. 当您的日志数据复杂时，可以将**匹配更多日志**设置为  ，即可验证您的正则表达式样例的通过率。

2. 输入多个待验证日志样例，单击**验证**，系统将验证样通过率。出现 时，即为验证通过，如下图所示：



配置自定义元数据

您可配置自定义元数据，可以用来区分日志。当前支持以下几种元数据配置。详情请参见 自定义元数据 。

- 机器组元数据：使用机器组元数据
- 采集路径：通过正则提取采集路径中的值作为元数据
- 自定义：自定义键值作为元数据

> **注意：**
> LogListener 2.8.7及以上版本才可以配置自定义元数据。

配置日志时间戳来源

您可选择以**日志采集时间**或者**指定日志字段**两种时间作为日志的时间戳：

日志采集时间

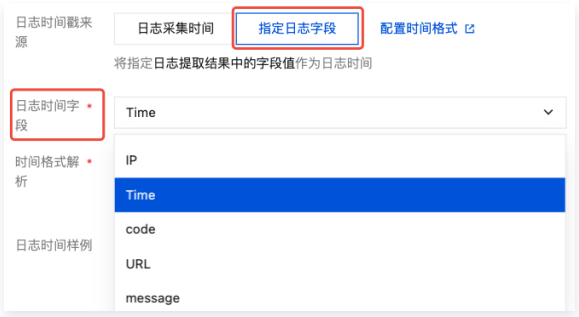
使用日志采集的时间作为日志时间。



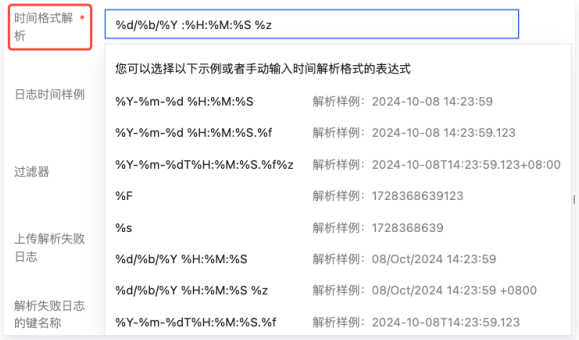
指定日志字段

使用日志中指定字段的值作为日志的时间。

- **在日志时间字段**中选择已经提取的 Value 值作为日志时间。



- **在时间格式解析**中手动输入或选择该值对应的解析表达式。例如：日志中代表时间的值为 07/Jul/2025:19:19:30 +0800，解析格式为：%d/%b/%Y:%H:%M:%S %z。更多请参见 配置时间格式 。



- 单击**验证**。

> **注意：**
> 若时间格式填写错误，日志时间将以采集时间为准。

配置过滤器条件

过滤器的目的是根据业务需要添加日志采集过滤规则，以帮助您筛选出有价值的日志数据。
完全正则采集时，需要根据所自定义的键值对来配置过滤条件。支持以下过滤规则：

- 等于：仅采集指定字段值匹配指定字符的日志。支持完全匹配，或正则匹配。
- 不等于：仅采集指定字段值不匹配指定字符的日志。支持完全匹配，或正则匹配。
- 字段存在：仅采集指定字段存在的日志。
- 字段不存在：仅采集指定字段不存在的日志。

例如，样例日志使用完全正则模式解析后，您希望 status 字段为400或500的所有日志数据被采集，那么 key 处配置 status，过滤规则选择**等于**，value 处配置 400|500。

> **注意**
> - Windows 系统环境暂不支持。
> - 过滤规则 “不等于”、“字段存在”以及“字段不存在” 仅在 LogListener 2.9.3及以上版本支持。
> - 多条过滤条件之间关系是"与"逻辑，若同一 key 名配置多条过滤条件，规则会被覆盖。

配置上传解析失败日志

建议开启上传解析失败日志。开启后，LogListener 会上传各式解析失败的日志。若关闭上传解析失败日志，则会丢弃失败的日志。



开启后需要配置解析失败的 Key 值（默认为 LogParseFailure），所有解析失败的日志，均以输入内容作为键名称（Key），原始日志内容作为值（Value）进行上传。

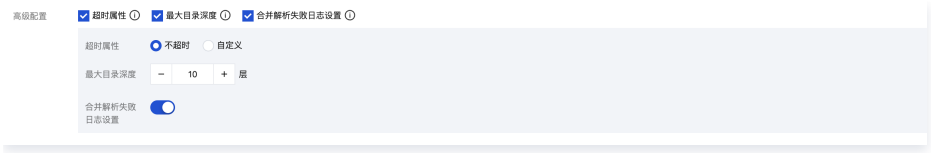
上传原始日志

开启后LogListener会将原始日志与解析后的日志一起上传。所有原始日志，均以您指定的键名称，原始日志内容将作为值（Value）进行上传。

高级配置

> **注意：**
> Windows 系统环境暂不支持。

通过勾选，选择您需要定义的高级配置。



多行-完全正则提取模式下，支持配置以下高级配置：

名称	描述	配置项
超时属性	该配置控制日志文件的超时时间。如果一个日志文件在指定时间内没有任何更新，则为超时。超时的日志文件 LogListener 将不再采集。当您的日志文件数量较大时，建议降低超时时间，避免 LogListener 性能浪费	• 不超时：日志文件永不超时 • 自定义：自定义日志文件的超时时间
最大目录深度	该配置控制日志采集的最大目录深度。LogListener 不会采集所在目录层级超过指定最大目录深度的日志文件。当您目标采集路径包含模糊匹配时，建议配置合适的最大目录深度，避免 LogListener 性能浪费。	大于0的整数。 0代表不进行子目录的下钻
合并解析失败日志	**⚠ 注意：** LogListener 2.8.8及以上版本才可以配置合并解析失败日志。 该配置支持在采集时将目标日志文件中连续解析失败的日志合并为一条日志上传。 若您的首行正则表达式无法覆盖所有的多行日志， 建议开启该配置，避免因首行匹配失败的单条多行日志被拆分至多条日志。	开启/关闭

步骤4：索引配置

1. **单击下一步**，进入“索引配置”页面。
2. 在“索引配置”页面，设置如下信息。配置详情请参见 索引配置 。



⚠ 注意：
检索必须开启索引配置，否则无法检索。

3. **单击提交**，进入编辑索引配置确认页面。



如果您设置的索引配置仅需对新写入的日志生效，单击**确认**。如果您想对历史数据生效配置，单击**确认**后，详情请参见 重建索引，做进一步设置。

4. 操作成功，完成采集配置。

相关操作

检索日志，请参见 检索分析 文档。

使用 JSON 提取模式采集日志

最近更新时间：2025-07-03 17:35:32

操作场景

JSON 提取模式适用于日志文本中每行内容为一条原始日志，且每条日志可按 JSON 解析规则提取为多个 key-value 键值的日志解析模式。若不需要提取 key-value，请参见 单行全文格式 进行配置。本文为您介绍如何使用 JSON 提取模式采集日志。

前提条件

- 目标文件所在服务器已安装 LogListener。详情请参见：
  - LogListener 安装指南（Linux 版）
  - LogListener 安装指南（Windows 版）
- LogListener Linux 版本为2.4.5以上或 LogListener Windows 版本为2.9.7及以上。

效果预览

假设您的一条 JSON 日志原始数据为：

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34
+0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost":"unix:/tmp/php-
cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","request":"POST /event/dispatch
HTTP/1.1","xff":"-","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0 (Windows NT 10.0; WOW64;
rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

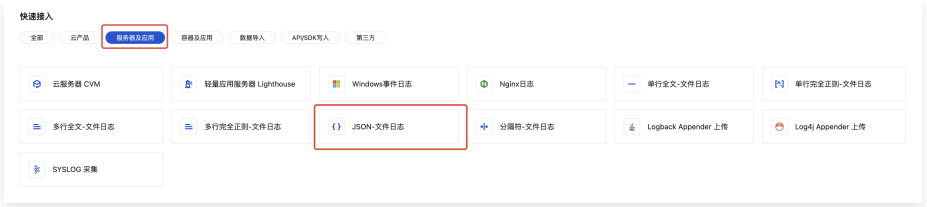
操作步骤

步骤1：创建/选择日志主题

创建新的日志主题

如果您想创建新的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**概览**，进入概览页面。
3. 在**快速接入**中，选择**服务器及应用**，找到并单击 **JSON-文件日志**，进入采集配置流程。



4. 在采集配置流程的**创建日志主题**页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步** 。



选择现有的日志主题

如果您想选择现有的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，选择需要投递的日志主题，单击指定日志主题名称，进入日志主题管理页面。
3. 选择**采集配置**页签，在 LogListener 采集配置栏下单击**新增**，进入日志数据源选择。



4. 在日志数据源选择页面中，选择**服务器及应用**，找到并单击 **JSON-文件日志**，进入采集配置流程。



步骤2：管理机器组（以从概览页创建/选择日志主题为例）

完成日志主题新建或选择之后，单击**下一步**进入采集配置流程 – 机器组管理页面。



若您需要采集的目标服务器未安装 LogListener， 可参见：

- LogListener 安装指南（Linux 版）
- LogListener 安装指南（Windows 版）

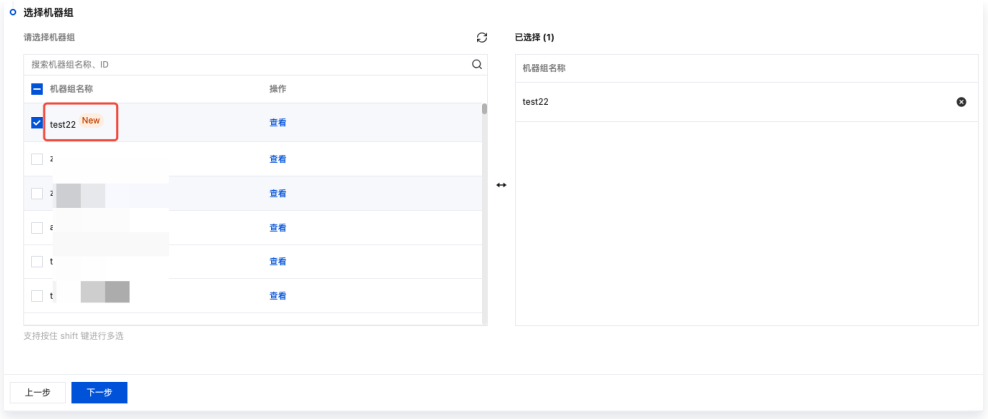
创建新的机器组

如果您想创建新的机器组，可执行如下操作：

1. 填写机器组名称，通过机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 机器组 。单击**下一步**。



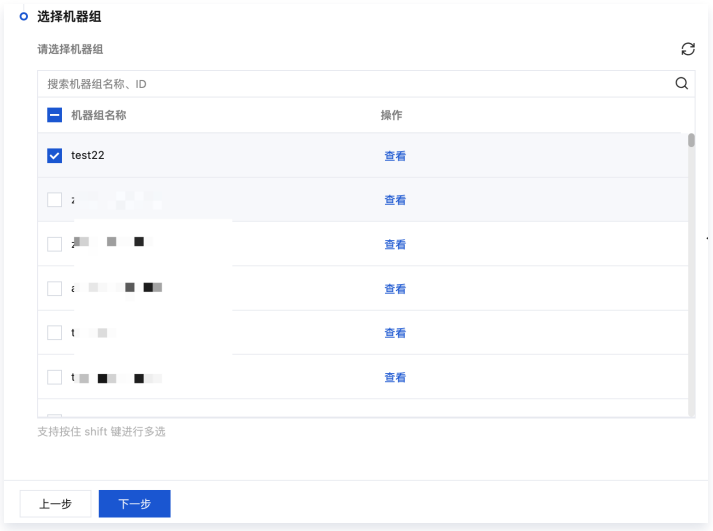
2. 完成创建后，在列表中勾选您创建的机器组。



选择已有的机器组

如果您想选择已有的机器组，可执行如下操作：

1. 单击**使用已有机器组**。
2. 在列表中勾选您的目标机器组，并单击**下一步**。



步骤3：采集配置

配置日志文件采集路径



在“采集配置”页面，填写采集规则名称，并根据日志采集路径格式填写“采集路径”。日志采集路径格式参考如下：

> **注意：**
> Linux日志路径必须以 `/` 开头。Windows文件路径必须以盘符开头，如 `C:\` 。

- Linux系统中的日志路径：`/[目录前缀表达式]/**/[文件名表达式]`，例如：`/data/log/**/*.log` 。
- Windows系统中的日志路径：`[盘符]:\[目录前缀表达式]\**\[文件名表达式]`，例如：`C:\Program Files\Tencent\...\*.log` 。

填写日志采集路径后，LogListener 会按照**[目录前缀表达式]**匹配所有符合规则的公共前缀路径，并监听这些目录（包含子层目录）下所有符合**[文件名表达式]**规则的日志文件。其参数详细说明如下：

序号	目录前缀表达式	文件名表达式	说明
1	/var/log/nginx	access.log	此例中，日志路径配置为 `/var/log/nginx/**/access.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `access.log` 命名的日志文件
2	/var/log/nginx	*.log	此例中，日志路径配置为 `/var/log/nginx/**/*.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `.log` 结尾的日志文件
3	/var/log/nginx	error*	此例中，日志路径配置为 `/var/log/nginx/**/error*`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `error` 开头命名的日志文件

注意：

- Windows 系统环境暂不支持软链采集。
- LogListener 2.3.9及以上版本才可以添加多个采集路径。
- 建议配置采集路径为 `log/*.log`，rename 日志轮转后的老文件命名为 `log/*.log.xxxx`。
- 默认情况下，一个日志文件只能被一个日志主题采集。如果一个文件需要对应多个采集配置，且文件所在系统环境为 Linux，请给源文件添加一个软链接，并将其加到另一组采集配置中。

配置采集路径黑名单

开启采集路径黑名单，可在采集时忽略指定的目录前缀或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



采集黑名单分为两类过滤类型，且可以同时使用：

- 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配*或？，支持**路径模糊匹配。
- 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配*或？，支持**路径模糊匹配。

注意：

- 需要 LogListener 2.3.9及以上版本。
- 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

配置采集策略

- 全量采集：LogListener 采集文件时，从文件的开头开始读。
- 增量采集：LogListener 采集文件时，只采集文件内新增的内容。

配置回溯采集



当采集策略选择为增量采集时，您可以进一步在此设置回溯采集的起始点，指定 LogListener 启动时是否从最新位置往前偏移您指定字节数开始采集。

注意：

Windows 系统环境暂不支持。

编码模式

- UTF-8：若您的日志文件编码模式为 UTF-8，请选择该选项。

- GBK：若您的日志文件编码模式为 GBK，请选择该选项。

配置 JSON 模式

> **注意：**
> LogListener 仅解析一层 JSON，更深层次的 JSON 解析请通过 索引配置 。

在**采集配置**页面，将“提取模式”设置为 **JSON**。如下图所示：



配置自定义元数据

您可配置自定义元数据，可以用来区分日志。当前支持以下几种元数据配置。详情请参见 自定义元数据 。

- 机器组元数据：使用机器组元数据。
- 采集路径：通过正则提取采集路径中的值作为元数据。
- 自定义：自定义键值作为元数据。

> **注意：**
> LogListener 2.8.7及以上版本才可以配置自定义元数据。

配置日志时间戳来源

日志采集时间

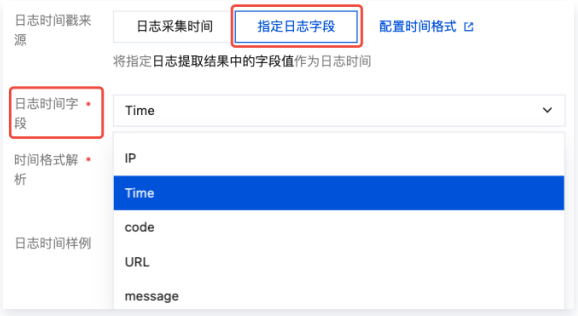
使用日志采集的时间作为日志时间。



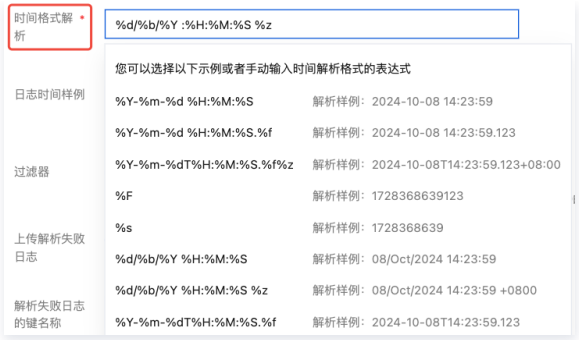
指定日志字段

使用日志中指定字段的值作为日志的时间。

1. 在**日志时间字段**中选择已经提取的 Value 值作为日志时间。



2. 在**时间格式解析**中手动输入或选择该值对应的解析表达式。例如：日志中代表时间的值为 07/Jul/2025:19:19:30 +0800，解析格式为：%d/%b/%Y:%H:%M:%S %z。更多请参见 配置时间格式 。



3. 单击**验证**。

> **注意：**
> 若时间格式填写错误，日志时间将以采集时间为准。

配置过滤器条件

过滤器的目的是根据业务需要添加日志采集过滤规则，以帮助您筛选出有价值的日志数据。
对于 JSON 格式日志，可以根据所解析成的键值对配置过滤条件。支持以下过滤规则：

- 等于：仅采集指定字段值匹配指定字符的日志。支持完全匹配，或正则匹配。
- 不等于：仅采集指定字段值不匹配指定字符的日志。支持完全匹配，或正则匹配。
- 字段存在：仅采集指定字段存在的日志。
- 字段不存在：仅采集指定字段不存在的日志。

例如，您希望原始 JSON 格式日志内容中 response_code 为400或500的所有日志数据被采集，那么 key 处配置 response_code，过滤规则选择等于，value 处配置400|500。

> **注意：**
> - Windows 系统环境暂不支持。
> - 过滤规则 “不等于”、“字段存在”以及“字段不存在” 仅在 LogListener 2.9.3及以上版本支持。
> - 多条过滤条件之间关系是"与"逻辑，若同一 key 名配置多条过滤条件，规则会被覆盖。

配置上传解析失败日志

建议开启上传解析失败日志。开启后，LogListener 会上传各式解析失败的日志。若关闭上传解析失败日志，则会丢弃失败的日志。



开启后需要配置解析失败的 Key 值（默认为 LogParseFailure），所有解析失败的日志，均以输入内容作为键名称（Key），原始日志内容作为值（Value）进行上传。

上传原始日志

开启后 LogListener 会将原始日志与解析后的日志一起上传。所有原始日志，均以您指定的键名称，原始日志内容将作为值（Value）进行上传。

高级配置

> **注意：**
> Windows 系统环境暂不支持。

通过勾选，选择您需要定义的高级配置。



JSON 提取模式下，支持使用以下高级配置：

名称	描述	配置项
超时属性	该配置控制日志文件的超时时间。如果一个日志文件在指定时间内没有任何更新，则为超时。超时的日志文件 LogListener 将不再采集。当您的日志文件数量较大时，建议缩短超时时间，避免 LogListener 性能浪费	• 不超时：日志文件永不超时 • 自定义：自定义日志文件的超时时间
最大目录深度	该配置控制日志采集的最大目录深度。LogListener 不会采集所在目录层级超过指定最大目录深度的日志文件。当您目标采集路径包含模糊匹配时，建议配置合适的最大目录深度，避免 LogListener 性能浪费。	大于0的整数。0代表不进行子目录的下钻

步骤4：索引配置

1. 单击**下一步**，进入“索引配置”页面。
2. 在“索引配置”页面，设置如下信息。配置详情请参见 索引配置 。



> **注意：**
> 检索必须开启索引配置，否则无法检索。

3. 单击**提交**，进入编辑索引配置确认页面。



如果您设置的索引配置仅需对新写入的日志生效，单击**确认**。如果您想对历史数据生效配置，单击**确认后**，详情请参见 重建索引 ，做进一步设置。

4. 操作成功，完成采集配置。

相关操作

检索日志，请参见 检索分析 文档。

使用分隔符提取模式采集日志

最近更新时间：2025-07-04 15:17:42

版权所有：腾讯云计算（北京）有限责任公司

操作场景

分隔符提取模式适用于日志文本中每行内容为一条原始日志，且每条日志可根据指定的分隔符提取为多个 key-value 键值的日志解析模式。若不需要提取 key-value，请参见 单行全文格式 进行配置。本文为您介绍如何使用分隔符提取模式采集日志。

前提条件

- 目标文件所在服务器已安装 LogListener。详情请参见：
  - LogListener 安装指南（Linux 版）
  - LogListener 安装指南（Windows 版）
- LogListener Linux 版本为2.4.5以上或 LogListener Windows 版本为2.9.7及以上。

效果预览

假设您的一条日志原始数据为：

```
10.20.20.10 - ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/1.1 ::: 127.0.0.1 :::
200 ::: 647 ::: 35 ::: http://127.0.0.1/
```

当日志解析的分隔符指定为 `:::`，该条日志会被分割成八个字段，并为这八个字段定义唯一的 key，如下所示：

```
IP: 10.20.20.10 -
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

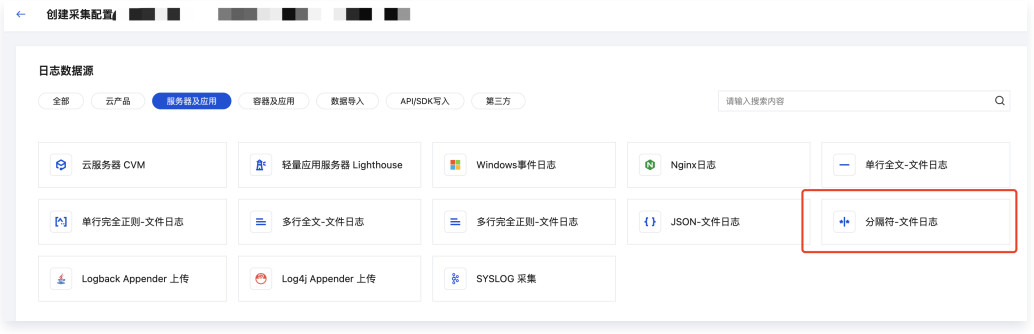
操作步骤

步骤1：创建/选择日志主题

创建新的日志主题

如果您想创建新的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**概览**，进入概览页面。
3. 在**快速接入**中，选择**服务器及应用**，找到并单击**分隔符-文件日志**，进入采集配置流程。



4. 在采集配置流程的**创建日志主题**页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步** 。



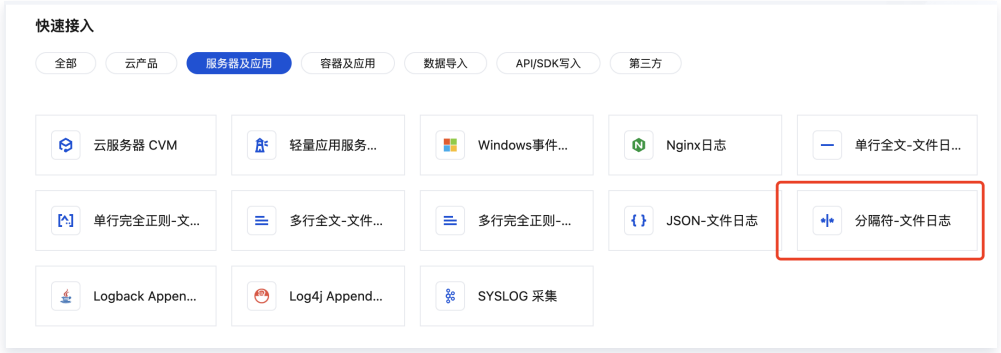
选择现有的日志主题

如果您想选择现有的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，选择需要投递的日志主题，单击指定日志主题名称，进入日志主题管理页面。
3. 选择**采集配置**页签，在 LogListener 采集配置栏下单击**新增**，进入日志数据源选择。



4. 在日志数据源选择页面中，选择**服务器及应用**，找到并单击**分隔符-文件日志**，进入采集配置流程。



步骤2：管理机器组（以从概览页创建/选择日志主题为例）

若您需要采集的目标服务器未安装 LogListener，可参见：

- LogListener 安装指南（Linux 版）
- LogListener 安装指南（Windows 版）

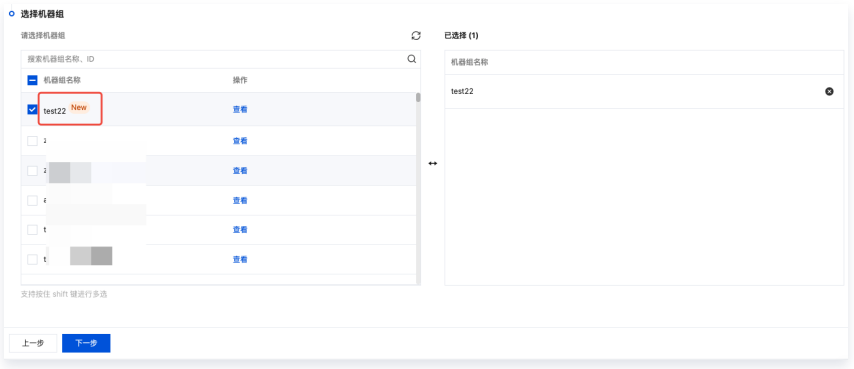
创建新的机器组

如果您想创建新的机器组，可执行如下操作：

1. 填写机器组名称，通过机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 机器组 。单击**下一步**。



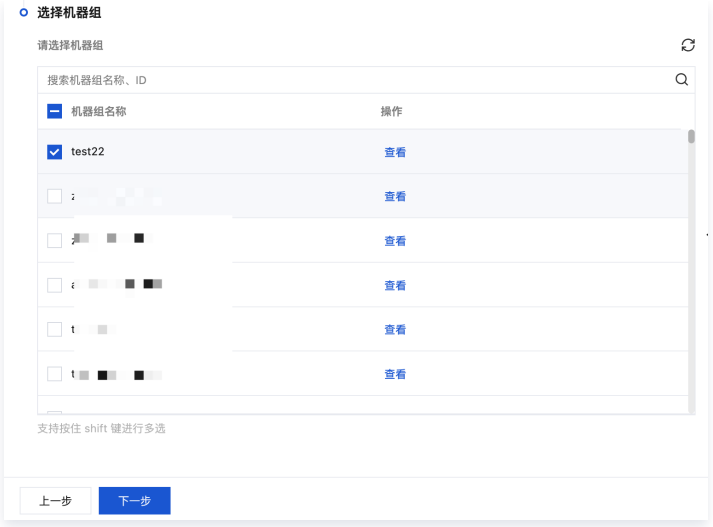
2. 完成创建后，在列表中勾选您创建的机器组。



选择已有的机器组

如果您想选择已有的机器组，可执行如下操作：

1. 单击**使用已有机器组**。
2. 在列表中勾选您的目标机器组，并单击**下一步**。



步骤3：采集配置

配置日志文件采集路径



在“采集配置”页面，填写采集规则名称，并根据日志采集路径格式填写“采集路径”。日志采集路径格式参考如下：

> **注意：**
> Linux日志路径必须以 `/` 开头。Windows 文件路径必须以盘符开头，如 `C:\` 。

- Linux 系统中的日志路径：`/[目录前缀表达式]/**/[文件名表达式]`，例如：`/data/log/**/*.log` 。
- Windows 系统中的日志路径：`[盘符]:\[目录前缀表达式]\**\[文件名表达式]`，例如：`C:\Program Files\Tencent\...\*.log` 。

填写日志采集路径后，LogListener 会按照**[目录前缀表达式]**匹配所有符合规则的公共前缀路径，并监听这些目录（包含子层目录）下所有符合**[文件名表达式]**规则的日志文件。其参数详细说明如下：

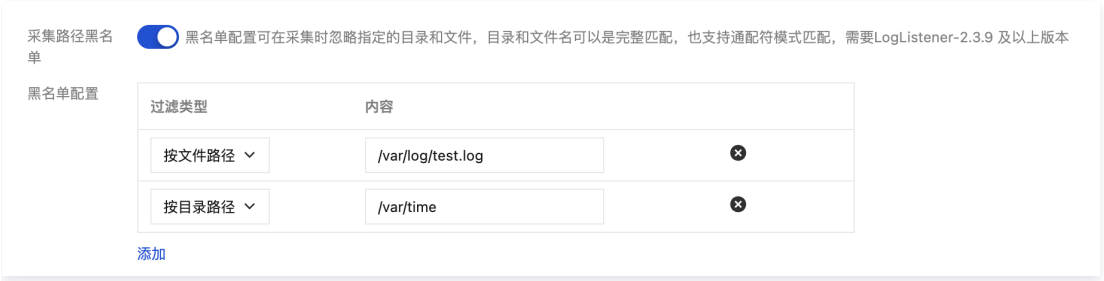
序号	目录前缀表达式	文件名表达式	说明
1	/var/log/nginx	access.log	此例中，日志路径配置为 `/var/log/nginx/**/access.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `access.log` 命名的日志文件
2	/var/log/nginx	*.log	此例中，日志路径配置为 `/var/log/nginx/**/*.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `.log` 结尾的日志文件
3	/var/log/nginx	error*	此例中，日志路径配置为 `/var/log/nginx/**/error*`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `error` 开头命名的日志文件

> **注意：**
> - Windows 系统环境暂不支持软链采集。
> - LogListener 2.3.9及以上版本才可以添加多个采集路径。
> - 建议配置采集路径为 `log/*.log`，rename 日志轮转后的老文件命名为 `log/*.log.xxxx` 。

- 默认情况下，一个日志文件只能被一个日志主题采集。如果一个文件需要对应多个采集配置，且文件所在系统环境为 Linux，请给源文件添加一个软链接，并将其加到另一组采集配置中。

配置采集路径黑名单

开启采集路径黑名单，可在采集时忽略指定的目录前缀或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



采集黑名单分为两类过滤类型，且可以同时使用：

- 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配*或？，支持**路径模糊匹配。
- 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配*或？，支持**路径模糊匹配。

注意：

- 需要 LogListener-2.3.9及以上版本。
- 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

配置采集策略

- 全量采集：LogListener 采集文件时，从文件的开头开始读。
- 增量采集：LogListener 采集文件时，只采集文件内新增的内容。

配置回溯采集



当采集策略选择为增量采集时，您可以进一步在此设置回溯采集的起始点，指定 LogListener 启动时是否从最新位置往前偏移您指定字节数开始采集。

注意：

Windows 系统环境暂不支持。

编码模式

- UTF-8：若您的日志文件编码模式为 UTF-8，请选择该选项。
- GBK：若您的日志文件编码模式为 GBK，请选择该选项。

配置分隔符模式

1. 将“提取模式”设置为**分隔符**。
2. 选择**分隔符**，并在“日志样例”文本框中，输入日志样例，单击**提取**。
   系统根据确定的分隔符将日志样例进行切分，并展示在抽取结果栏中，您需要为每个字段定义唯一的 key。目前，日志采集支持多种分隔符，常见的分隔符有：空格、制表符、逗号、分号、竖线，若您的日志数据所采用的分隔符是其他符号，例如 `:::` ，也可以通过自定义分词符进行解析。



配置自定义元数据

> **注意：**
> LogListener 2.8.7及以上版本才可以配置自定义元数据。

您可配置自定义元数据，可以用来区分日志。当前支持以下几种元数据配置。详情请参见 自定义元数据 。

- 机器组元数据：使用机器组元数据。
- 采集路径：通过正则提取采集路径中的值作为元数据。
- 自定义：自定义键值作为元数据。

配置日志时间戳来源

日志采集时间

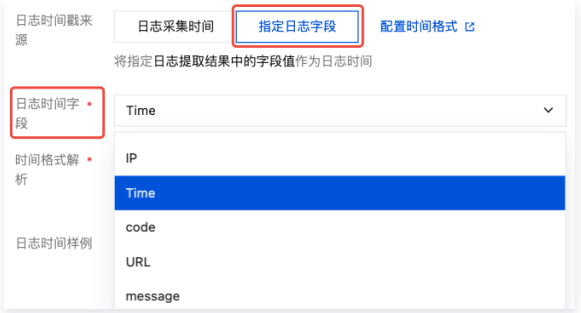
使用日志采集的时间作为日志时间。



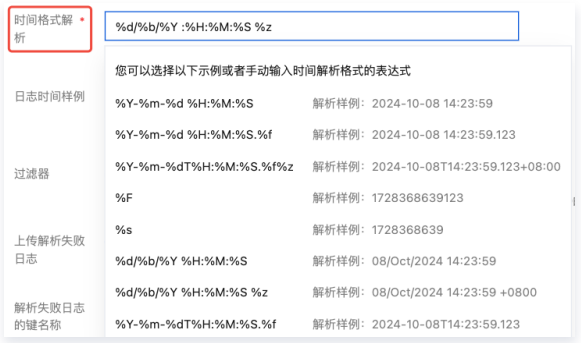
指定日志字段

使用日志中指定字段的值作为日志的时间。

1. 在**日志时间字段**中选择已经提取的 Value 值作为日志时间。



2. 在**时间格式解析**中手动输入或选择该值对应的解析表达式。例如：日志中代表时间的值为 07/Jul/2025:19:19:30 +0800，解析格式为：%d/%b/%Y :%H:%M:%S %z。更多请参见 配置时间格式 。



3. **单击验证。**

> **注意：**
> 若时间格式填写错误，日志时间将以采集时间为准。

配置过滤器条件

过滤器的目的是根据业务需要添加日志采集过滤规则，以帮助您筛选出有价值的日志数据。
分隔符格式日志需要根据自定义的键值对来配置过滤条件。支持以下过滤规则：

- 等于：仅采集指定字段值匹配指定字符的日志。支持完全匹配，或正则匹配。
- 不等于：仅采集指定字段值不匹配指定字符的日志。支持完全匹配，或正则匹配。
- 字段存在：仅采集指定字段存在的日志。
- 字段不存在：仅采集指定字段不存在的日志。

例如，样例日志使用分隔符模式解析后，您希望 status 字段为400或500的所有日志数据被采集，那么 key 处配置 status，过滤规则选择**等于**，value 处配置400|500。

> **注意：**
> - Windows 系统环境暂不支持。
> - 过滤规则 “不等于” 、“字段存在”以及“字段不存在”仅在 LogListener 2.9.3及以上版本支持。
> - 多条过滤条件之间关系是“与”逻辑，若同一 key 名配置多条过滤条件，规则会被覆盖。

配置上传解析失败日志

建议开启上传解析失败日志。开启后，LogListener 会上传各式解析失败的日志。若关闭上传解析失败日志，则会丢弃失败的日志。



开启后需要配置解析失败的 Key 值（默认为 LogParseFailure），所有解析失败的日志，均以输入内容作为键名称（Key），原始日志内容作为值

（Value）进行上传。

高级配置

> **注意：**
> Windows 系统环境暂不支持。

通过勾选，选择您需要定义的高级配置。



分隔符提取模式下，支持使用以下高级配置：

名称	描述	配置项
超时属性	该配置控制日志文件的超时时间。如果一个日志文件在指定时间内没有任何更新，则为超时。超时的日志文件 LogListener 将不再采集。当您的日志文件数量较大时，建议降低超时时间，避免 LogListener 性能浪费	• 不超时：日志文件永不超时 • 自定义：自定义日志文件的超时时间
最大目录深度	该配置控制日志采集的最大目录深度。LogListener 不会采集所在目录层级超过指定最大目录深度的日志文件。当您目标采集路径包含模糊匹配时，建议配置合适的最大目录深度，避免 LogListener 性能浪费。	大于0的整数。0代表不进行子目录的下钻

步骤4：索引配置

1. **单击下一步，进入索引配置页面。**
2. **在“索引配置”页面，设置如下信息。配置详情请参见** 索引配置 。



> **注意：**
> 检索必须开启索引配置，否则无法检索。

3. 单击**提交**，进入编辑索引配置确认页面。



如果您设置的索引配置仅需对新写入的日志生效，单击**确认**。如果您想对历史数据生效配置，单击**确认**后，详情请参见 重建索引，做进一步设置。

4. 操作成功，完成采集配置。

相关操作

检索日志，请参见 检索分析 文档。

使用组合解析提取模式采集日志

最近更新时间：2025-07-03 17:35:32

> **注意：**
> Windows 系统环境暂不支持。

操作场景

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。本文为您介绍如何使用组合解析提取模式采集日志。

前提条件

- 目标文件所在服务器已安装 LogListener，详情请参见 LogListener 安装指南（Linux 版）。
- LogListener Linux 版本为2.6.4以上。

效果预览

假设您的一条日志的原始数据为：

```
1571394459,http://127.0.0.1/my/course/4|10.135.46.111|200,status:DEAD
```

自定义插件内容如下：

```
{
 "processors": [
   {
     "type": "processor_split_delimiter",
     "detail": {
       "Delimiter": ",",
       "ExtractKeys": [ "time", "msg1","msg2"]
     },
     "processors": [
       {
         "type": "processor_timeformat",
         "detail": {
           "KeepSource": true,
           "TimeFormat": "%s",
           "SourceKey": "time"
          }
       },
       {
         "type": "processor_split_delimiter",
         "detail": {
           "KeepSource": false,
           "Delimiter": "|",
           "SourceKey": "msg1",
           "ExtractKeys": [ "submsg1","submsg2","submsg3"]
          },
          "processors": []
       },
       {
         "type": "processor_split_key_value",
         "detail": {
           "KeepSource": false,
           "Delimiter": ":",
           "SourceKey": "msg2"
```

```
          }
        }
      ]
    }
  ]
}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
time: 1571394459
submsg1: http://127.0.0.1/my/course/4
submsg2: 10.135.46.111
submsg3: 200
status: DEAD
```

配置说明

自定义插件种类

插件功能	插件名称	功能
提取字段	processor_log_string	使用 processor_log_string 插件对字段进行多字符解析（换行符），一般用于单行日志的高级功能
提取字段	processor_multiline	使用 processor_multiline 插件（正则模式）对字段进行首行正则解析，一般用于多行日志的高级功能
提取字段	processor_multiline_fullregex	使用 processor_multiline_fullregex 插件（正则模式）对字段进行首行正则解析，一般用于多行日志的高级功能；并对多行日志进行正则提取
提取字段	processor_fullregex	使用 processor_fullregex 插件（正则模式）提取字段（单行日志）
提取字段	processor_json	使用 processor_json 插件对字段值进行 JSON 展开
提取字段	processor_split_delimiter	使用 processor_split_delimiter 插件（单字符/多字符分隔符模式）提取字段
提取字段	processor_split_key_value	使用 processor_split_key_value 插件（键值对模式）提取字段
处理字段	processor_drop	使用 processor_drop 插件丢弃字段
处理字段	processor_timeformat	使用 processor_timeformat 插件，解析原始日志中的时间字段，用于转换时间格式，并可将解析结果设置为日志时间

自定义插件详细参数

插件名称	是否支持子项解析	插件参数	是否必须	功能
processor_multiline	否	BeginRegex	是	定义多行日志的行首匹配正则
processor_multiline_fullregex	是	BeginRegex	是	定义多行日志的行首匹配正则
		ExtractRegex	是	定义提取到多行日志后的提取正则
		ExtractKeys	是	定义提取键值
processor_fullregex	是	ExtractRegex	是	定义提取正则
		ExtractKeys	是	定义提取键值
processor_json	是	SourceKey	否	当前 processor 处理的上一级 processor 中的 key name

		KeepSource	否	最终键值名称中，是否保留 SourceKey
processor_split_delimiter	是	SourceKey	否	当前 processor 处理的上一级 processor 中的 key name
		KeepSource	否	最终键值名称中，是否保留 SourceKey
		Delimiter	是	指定分隔符（单/多字符）
		ExtractKeys	是	定义分隔符分割之后的提取键值
processor_split_key_value	否	SourceKey	否	当前 processor 处理的上一级 processor 中的 key name
		KeepSource	否	最终键值名称中，是否保留 SourceKey
		Delimiter	是	定义字符串中 Key 与 Value 之间的分隔符
processor_drop	否	SourceKey	是	当前 processor 处理的上一级 processor 中的 key name
processor_timeformat	否	SourceKey	是	当前 processor 处理的上一级 processor 中的 key name
		TimeFormat	是	定义对 SourceKey 的值（日志中的时间数据字符串）的时间解析格式

操作步骤

步骤1：创建日志主题

创建新的日志主题

如果您想创建新的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**概览**，进入概览页面。
3. 在快速接入中，找到全部，单击**云服务器 CVM**，进入采集配置流程。



4. 在采集配置流程的**创建日志主题**页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步** 。



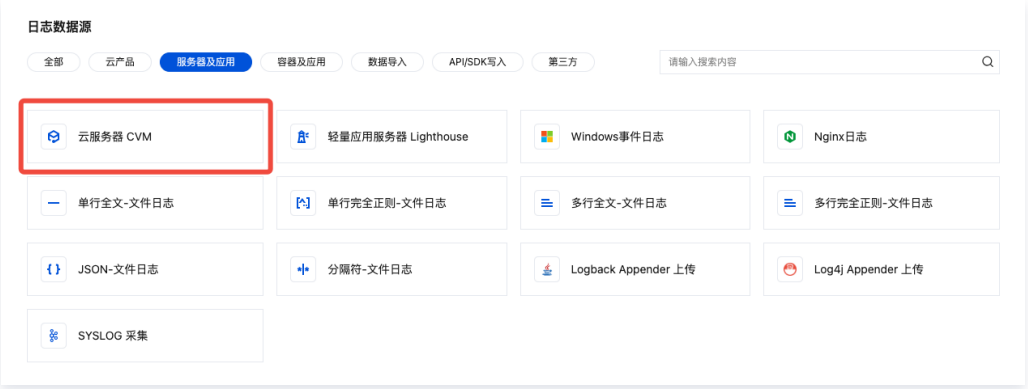
选择现有的日志主题

如果您想选择现有的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，选择需要投递的日志主题，单击指定日志主题名称，进入日志主题管理页面。
3. 选择**采集配置**页签，在 LogListener 采集配置栏下单击**新增**，进入日志数据源选择。



4. 在日志数据源选择页面中，选择**服务器及应用**，找到并单击**云服务器 CVM**，进入采集配置流程。



步骤2：机器组管理（以从概览页创建/选择日志主题为例）

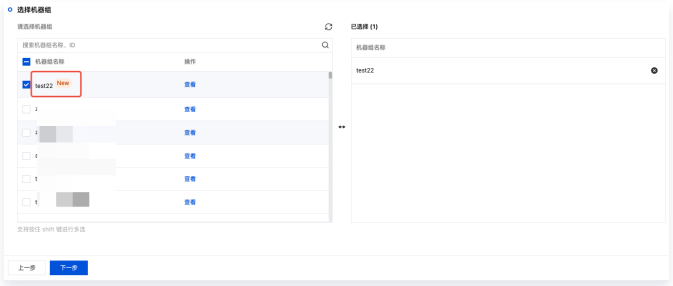
创建新的机器组

如果您想创建新的机器组，可执行如下操作：

1. 填写机器组名称，通过机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 机器组 。单击**下一步**。



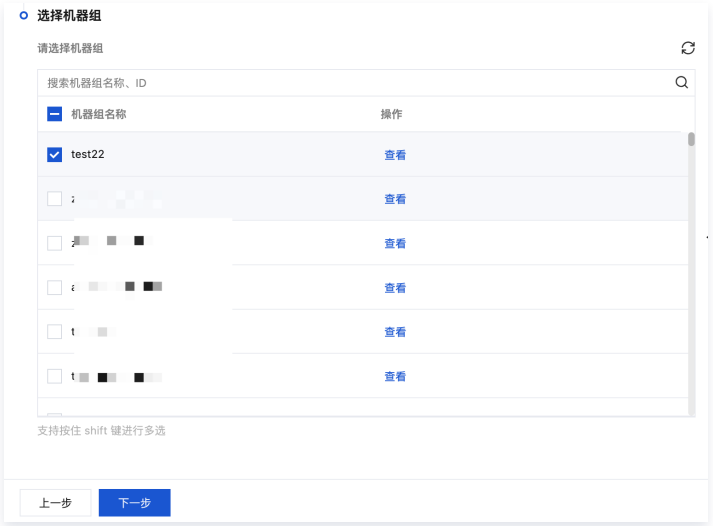
2. 完成创建后，在列表中勾选您创建的机器组。



选择已有的机器组

如果您想选择已有的机器组，可执行如下操作：

1. 单击**使用已有机器组**。
2. 在列表中勾选您的目标机器组，并单击**下一步**。



步骤3：采集配置

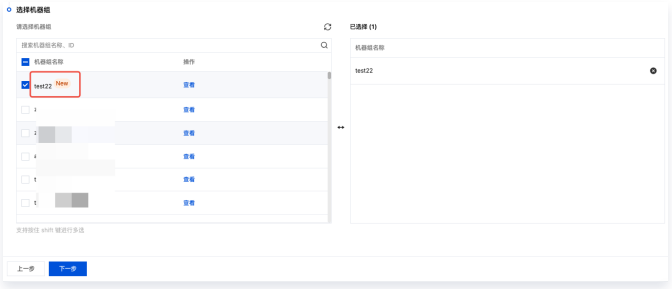
创建新的机器组

如果您想创建新的机器组，可执行如下操作：

1. 填写机器组名称，通过机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 机器组 。单击**下一步**。



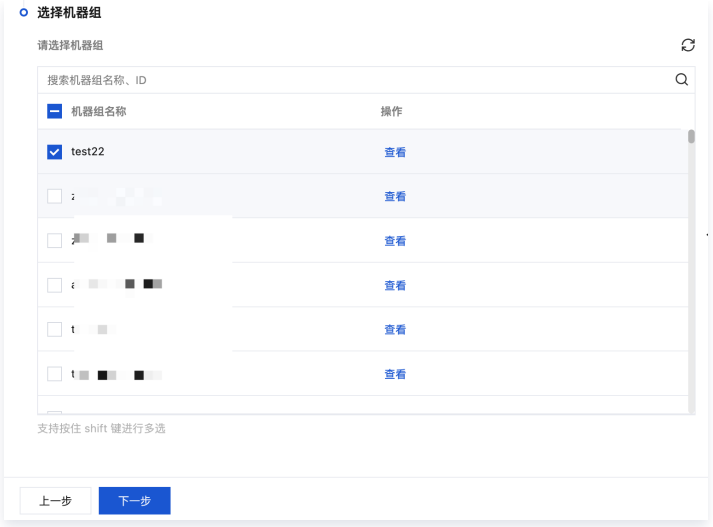
2. 完成创建后，在列表中勾选您创建的机器组。



选择已有的机器组

如果您想选择已有的机器组，可执行如下操作：

1. 单击**使用已有机器组**。
2. 在列表中勾选您的目标机器组，并单击**下一步**。



配置日志文件采集路径

在“采集配置”页面，根据日志采集路径格式，填写“采集路径”。如下图所示：
日志采集路径格式：`[目录前缀表达式]/**/[文件名表达式]`。



填写日志采集路径后，LogListener 会按照**[目录前缀表达式]**匹配所有符合规则的公共前缀路径，并监听这些目录（包含子层目录）下所有符合**[文件名表达式]**规则的日志文件。其参数详细说明如下：

字段	说明
目录前缀	日志文件前缀目录结构，仅支持通配符 * 和 ？，* 表示匹配多个任意字符?， 表示匹配单个任意字符，不支持填写逗号。
/**/	表示当前目录以及所有子目录。
文件名	日志文件名，仅支持通配符 * 和 ？，* 表示匹配多个任意字符？， 表示匹配单个任意字符，不支持填写逗号。

常用的配置模式如下：

- [公共目录前缀]/**/[公共文件名前缀]*
- [公共目录前缀]/**/*[公共文件名后缀]
- [公共目录前缀]/**/[公共文件名前缀]*[公共文件名后缀]
- [公共目录前缀]/**/*[公共字符串]*

填写示例如下：

序号	目录前缀表达式	文件名表达式	说明
1	/var/log/nginx	access.log	此例中，日志路径配置为 `/var/log/nginx/**/access.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `access.log` 命名的日志文件
2	/var/log/nginx	*.log	此例中，日志路径配置为 `/var/log/nginx/**/*.log`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `.log` 结尾的日志文件

版权所有：腾讯云计算（北京）有限责任公司

3	/var/log/nginx	error*	此例中，日志路径配置为 `/var/log/nginx/**/error*`，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `error` 开头命名的日志文件

注意：
- LogListener 2.3.9及以上版本才可以添加多个采集路径。
- 建议配置采集路径为 `log/*.log`，rename 日志轮转后的老文件命名为 `log/*.log.xxxx`。
- 默认情况下，一个日志文件只能被一个日志主题采集。如果一个文件需要对应多个采集配置，请给源文件添加一个软链接，并将其加到另一组采集配置中。

配置采集路径黑名单

开启采集路径黑名单，可在采集时忽略指定的目录前缀或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



采集黑名单分为两类过滤类型，且可以同时使用：
- 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配*或？，支持**路径模糊匹配。
- 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配*或？，支持**路径模糊匹配。

注意：
- 需要 LogListener 2.3.9及以上版本。
- 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

配置采集策略

- 全量采集：LogListener 采集文件时，从文件的开头开始读。
- 增量采集：LogListener 采集文件时，只采集文件内新增的内容。

配置回溯采集



当采集策略选择为增量采集时，您可以进一步在此设置回溯采集的起始点，指定 LogListener 启动时是否从最新位置往前偏移您指定字节数开始采集。

注意：
Windows 系统环境暂不支持。

编码模式

- UTF-8：若您的日志文件编码模式为 UTF-8，请选择该选项。
- GBK：若您的日志文件编码模式为 GBK，请选择该选项。

配置组合解析模式

将**提取模式**设置为**组合解析**。如下图所示：



配置自定义元数据

> **注意：**
> LogListener 2.8.7及以上版本才可以配置自定义元数据。

您可配置自定义元数据可以用来区分日志。当前支持以下几种元数据配置。 详情请参见 自定义元数据 。

- 机器组元数据：使用机器组元数据。
- 采集路径：通过正则提取采集路径中的值作为元数据。
- 自定义：自定义键值作为元数据。

配置上传解析失败日志

建议开启上传解析失败日志。开启后，LogListener 会上传各式解析失败的日志。若关闭上传解析失败日志，则会丢弃失败的日志。



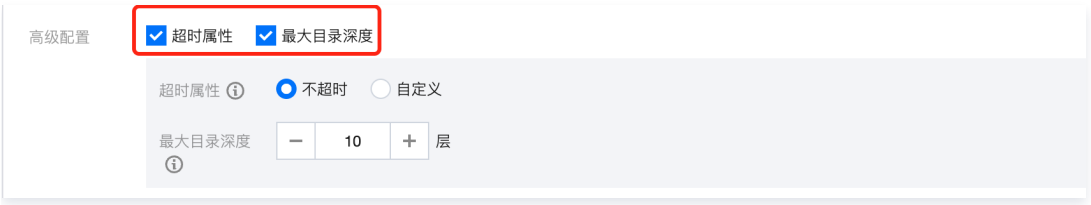
开启后需要配置解析失败的 Key 值（默认为 LogParseFailure），所有解析失败的日志，均以输入内容作为键名称（Key），原始日志内容作为值（Value）进行上传。

上传原始日志

开启后 LogListener 会将原始日志与解析后的日志一起上传。所有原始日志，均以您指定的键名称，原始日志内容将作为值（Value）进行上传。

高级配置

通过勾选，选择您需要定义的高级配置。



组合解析提取模式下，支持配置以下高级配置。

名称	描述	配置项
超时属性	该配置控制日志文件的超时时间。如果一个日志文件在指定时间内没有任何更新，则为超时。超时的日志文件 LogListener 将不再采集。当您的日志文件数量较大时，建议缩短超时时间，避免 LogListener 性能浪费	• 不超时：日志文件永不超时 • 自定义：自定义日志文件的超时时间
最大目录深度	该配置控制日志采集的最大目录深度。LogListener 不会采集所在目录层级超过指定最大目录深度的日志文件。当您目标采集路径包含模糊匹配时，建议配置合适的最大目录深度，避免 LogListener 性能浪费。	大于0的整数。 0代表不进行子目录的下钻

步骤4：索引配置

1. 单击**下一步**，进入“索引配置”页面。
2. 在“索引配置”页面，设置如下信息。配置详情请参见 索引配置 。



> **注意：**
> 检索必须开启索引配置，否则无法检索。

3. 单击**提交**，进入编辑索引配置确认页面。



如果您设置的索引配置仅需对新写入的日志生效，单击**确认**。如果您想对历史数据生效配置，单击**确认**后，详情请参见 重建索引，做进一步设置。

4. 操作成功，完成采集配置。

使用限制

- 使用组合解析模式解析数据时，LogListener 会需要消耗更多的资源，不建议您使用过于复杂的插件组合来处理数据。
- 使用组合解析模式后，文本模式使用采集功能、过滤器功能将失效，但其中部分功能可通过相关自定义插件实现。
- 使用组合解析模式后，上传解析失败日志功能默认开启，解析失败的日志均以输入名称为键（Key），原始日志内容作为值（Value）进行上传。

相关操作

检索日志

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**检索分析**，进入检索分析页面。
3. 根据实际需求，选择地域、日志集与日志主题，即可开始按照设定的查询条件检索日志。

配置时间格式

最近更新时间：2025-02-08 15:34:12

日志服务要求每条日志必须具有时间属性，以便系统按时间维度对数据进行管理。当使用 Loglistener 采集日志时，时间属性有两种配置方式：

- 默认方式：使用 Loglistener 采集时间作为时间属性。
- 自定义方式：将日志内容中的某个时间字段作为时间属性，需要用户配置时间解析格式。

> **说明：**
> Loglistener 采集时间精度为毫秒，所以时间解析格式支持配置到毫秒，按格式配置后不到毫秒级的会自动使用0填充。

解析格式说明

参数格式	说明	示例
%a	星期英文单词名称的简写	Fri
%A	星期英文单词名称的全称	Friday
%b	月份英文单词名称的简写	Jan
%B	月份英文单词名称的全称	January
%d	一个月中的第几天（01 – 31）	31
%h	月份英文单词名称的简写，与 `%b` 相同	Jan
%H	小时，24小时制（00 – 23）	22
%I	小时，12小时制（01 – 12）	11
%m	月份（01 – 12），01表示一月份	08
%M	分钟（00 – 59），01表示第一分钟	59
%n	换行符	换行符
%p	上午（AM）或下午（PM）	AM/PM
%r	一种特定的12小时制时间组合格式，等价于 `%I:%M:%S %p`	11:59:59 AM
%R	一种特定的24小时时间组合格式，等价于 `%H:%M`	23:59
%S	秒数（00 – 59）	59
%f	毫秒时间	0.123
%o	微秒时间	0.123123
%q	纳秒时间	0.123123123
%t	tab 制表符	tab 制表符
%y	不包含世纪的年份数字（00 – 99）	19
%Y	包含世纪的年份数字，2018表示2018年	2019
%C	世纪数字（年份除100，范围00 – 99）	20
%e	一个月中的第几天（01 – 31）	31
%j	一年中的第几天（001 – 366）	365
%u	星期数字的表示方式（1 – 7），1表示星期一，7表示星期天	1

%U	一年中的第几周（00 – 53），星期天是一周的开始，即从第一个星期日开始，作为第一周的第一天	23
%w	星期数字的表示方式（0 – 6），0代表星期天，6表示星期六	5
%W	一年中的第几周（00 – 53），星期一是一周的开始，即从第一个星期一开始，作为第一周的第一天	23
%s	秒级（10位）Unix 时间戳	1571394459
%F	毫秒级（13位）Unix 时间戳（需要 loglistener 2.6.4及以上版本）	1571394459123
%Q	微秒级（16位）Unix 时间戳（需要 loglistener 2.8.4及以上版本）	1571394459123456
%z	支持时间字段的时区解析，包括 ISO 8601时间格式和 GMT 时间格式	UTC/+0800/MST

配置示例

时间表示示例	时间提取格式
2018-07-16 13:12:57.123	%Y-%m-%d %H:%M:%S.%f
[2018-07-16 13:12:57.012]	[%Y-%m-%d %H:%M:%S.%f]
06/Aug/2019 12:12:19	%d/%b/%Y %H:%M:%S
Monday, 02-Oct-19 16:07:05	%A, %d-%b-%y %H:%M:%S
1571394459	%s
1571394459123	%F
06/Aug/2019 12:12:19 +0800	%d/%b/%Y %H:%M:%S %z
Monday, 02-Oct-19 16:07:05 MST	%A, %d-%b-%y %H:%M:%S %z

版权所有：腾讯云计算（北京）有限责任公司

自定义元数据

最近更新时间：2025-07-01 15:02:12

概述

自定义元数据可以用来区分日志， 您可以在日志服务控制台配置采集规则时为日志定义元数据。CLS 当前支持以下几种自定义元数据：

- 机器组元数据
- 采集路径
- 自定义

前提条件

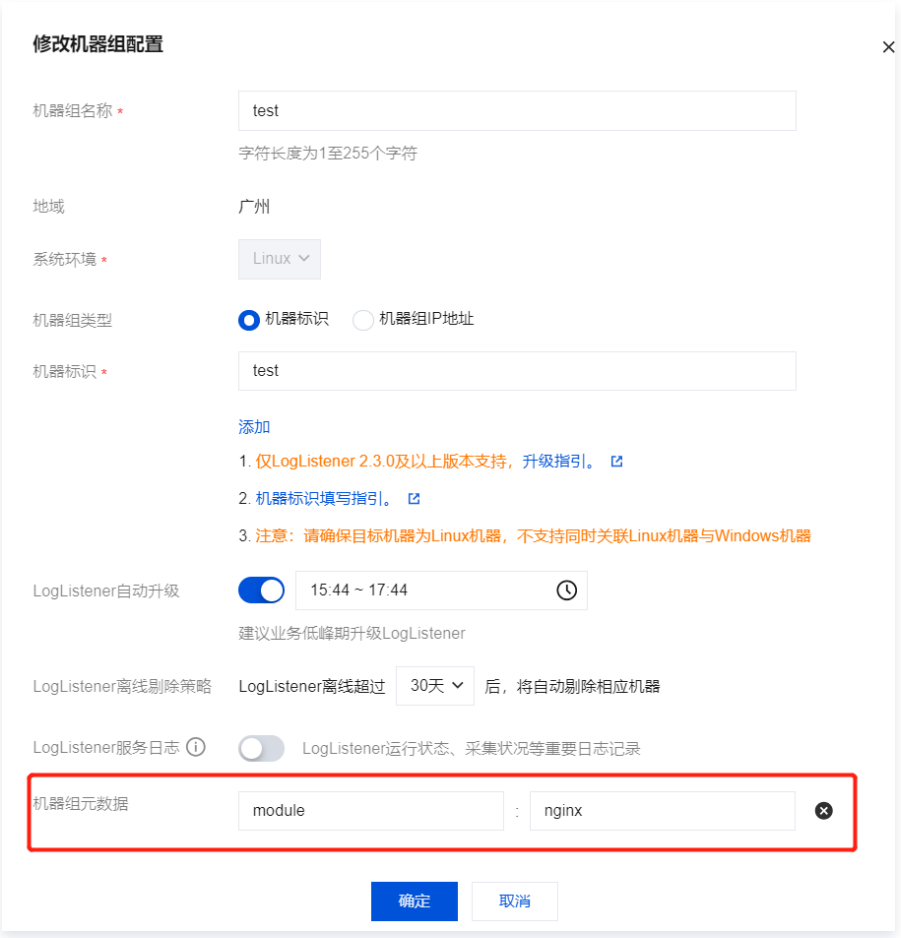
Loglistener 2.8.7及以上版本。

机器组元数据

当您需要采集多个机器组的日志，且目标采集路径均一致，这个时候您可以通过单一采集配置关联所有机器组，并通过机器组元数据来明确区分采集上来的日志是由哪个机器组产生的。

具体操作

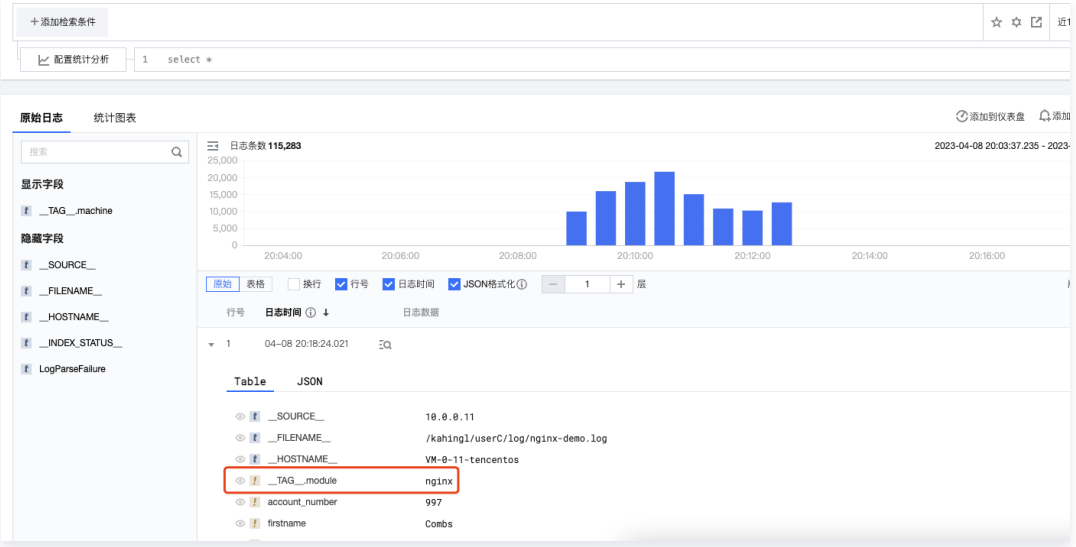
1. 在 创建/修改机器组 时，为所选机器组配置符合该机器组特征的机器组元数据。



2. 创建采集配置，并关联带有机器组元数据的目标机器组。
3. 在**采集规则**配置步骤中，开启**自定义元数据**，并将**元数据类型**设置为**机器组元数据**。



4. 完成配置后，LogListener 上报日志时会将日志所在机器组的机器组的元数据属性以 `__TAG__.{机器组元数据键}:{机器组元数据值}` 的形式上传至日志服务。



采集路径

当您的目标采集路径为模糊匹配时，您可提取文件路径中的指定字段作为自定义元数据， 并通过该自定义元数据明确区分日志来自于哪个文件路径。

具体操作

1. 在**采集规则**配置中， 开启**自定义元数据**， 将**元数据类型**设置为**采集路径**。
2. 填写文件路径正则表达式。该正则表达式需**完全匹配**文件路径，同时请用"()"标识路径中目标 KEY 对应的正则表达式。



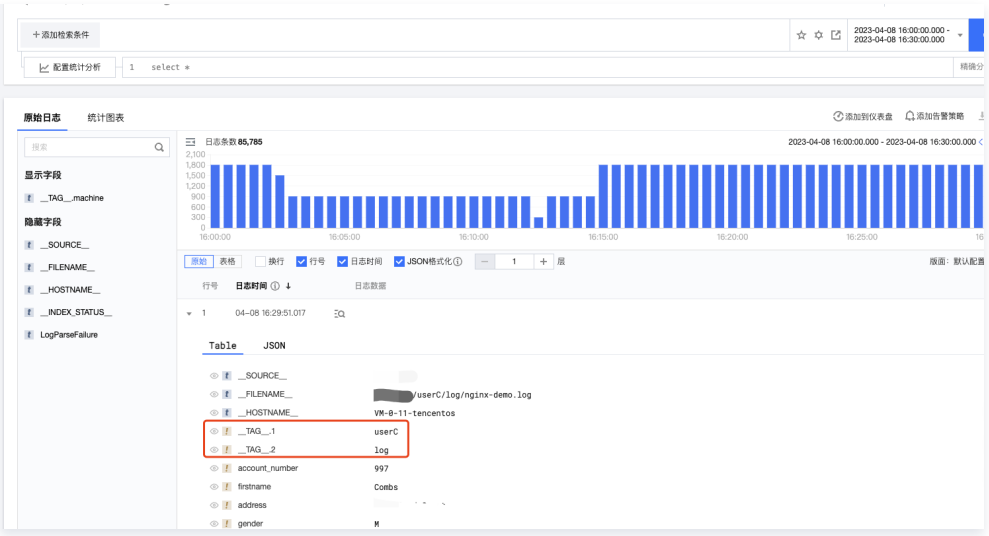
3. LogListener 采集日志时将视"()"为捕获组。LogListener 支持以下两种捕获组：
   - 命名捕获组

     命名捕获组通过 `?<>` 标识括号中正则提取的字段的名称，并以 `__TAG__.{字段名称}:{提取字段}` 的形式与日志一起上报。如：`(?<name>.*?)` 则代表通过 `.*?` 提取出来的字段将被命名为 name。最多支持5个命名捕获组。



   - 非命名捕获组

     非命名捕获组不会为提取字段进行特别命名，而是默认使用捕获组的顺位序号作为字段名称，并将以 `__TAG__.{i}:{提取字段}` 的形式与日志一起上报，其中 i 为捕获组的序号。最多支持5个非命名捕获组。



样例

假设文件目录结构如下：

```
/logs
  | - /appA/userA
    | - access.log
  | - /appB/userB
    | - access.log
  | - /appC/userC
    | - access.log
```

如果配置目标采集路径时采用模糊匹配的方式将目录前缀设为/logs，文件名设为 access.log，三个 access.log 将一起被采集至同一日志主题。这导致在检索分析时无法明确区分日志是来自于哪一个日志文件。然而我们可以通过以下正则表达式，提取文件路径中的值， 并作为元数据与日志一起上传。

- **正则表达式：**

```
/logs/(.*?)/.*
```

- **通过以上正则提取，将上报以下元数据：**

```
# /logs/appA/userA/access.log 中的日志将新增键值
__TAG__.1: appA

# /logs/appB/userB/access.log 中的日志将新增键值
__TAG__.1: appB

# /logs/appC/userC/access.log 中的日志将新增键值
__TAG__.1: appC
```

进一步， 您也可以使用以下正则表达式，通过多个命名捕获组从文件路径中提取多个值作为元数据上报， 并提取值命名。

- **正则表达式：**

```
/logs/(?<APP>.*?)/(?<USER>.*?)/access.log
```

- **通过以上正则提取，将上报以下元数据：**

```
# /logs/appA/userA/access.log 中的日志将新增键值
__TAG__.APP: appA
__TAG__.USER: userA
```

```
# /logs/appB/userB/access.log 中的日志将新增键值
__TAG__.APP: appB
__TAG__.USER: userB

# /logs/appC/userC/access.log 中的日志将新增键值
__TAG__.APP: appC
__TAG__.USER: userC
```

自定义

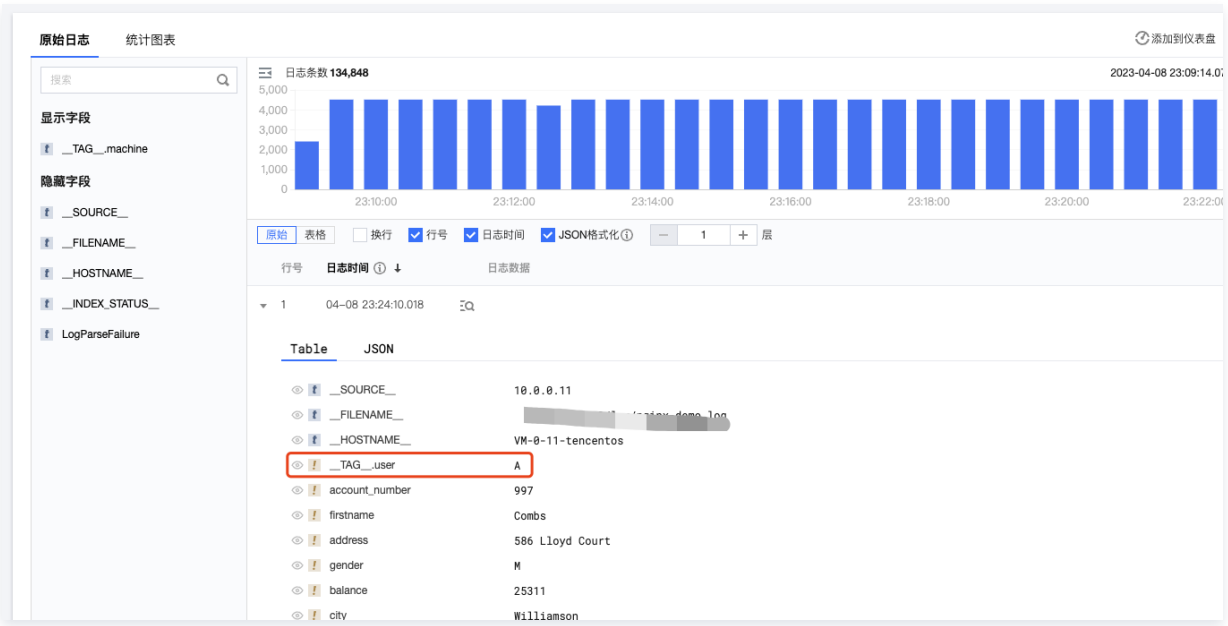
您也可以在配置采集规则时， 配置自定义的元数据。LogListener 在上报每一条日志时都会带上所指定的自定义元数据。

具体操作

1. **在采集规则配置中，开启自定义元数据， 将元数据类型设置为自定义。**
2. **填写自定义元数据的键与值。**



3. LogListener 在上报每一条日志时都会带上所指定的自定义元数据。



采集自建 K8s 集群日志
使用控制台配置自建 K8s 集群日志采集

最近更新时间：2025-06-27 15:03:21

本文将介绍如何在控制台配置自建 K8s 环境的日志采集规则并投递到 腾讯云日志服务 CLS 。

使用场景

自建 K8s 日志采集功能是为用户提供的非腾讯云 Kubernetes 集群内的日志采集工具，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 。日志采集功能适用于需要对 Kubernetes 集群内服务日志进行存储和分析的用户。
日志采集功能需要为每个集群手动开启并配置采集规则。日志采集功能开启后，日志采集 Agent 会在集群内以 DaemonSet 的形式运行，并根据用户通过日志采集规则配置的采集源、CLS 日志主题和日志解析方式，从采集源进行日志采集，将日志内容发送到日志消费端。您可根据以下操作开启日志采集功能。

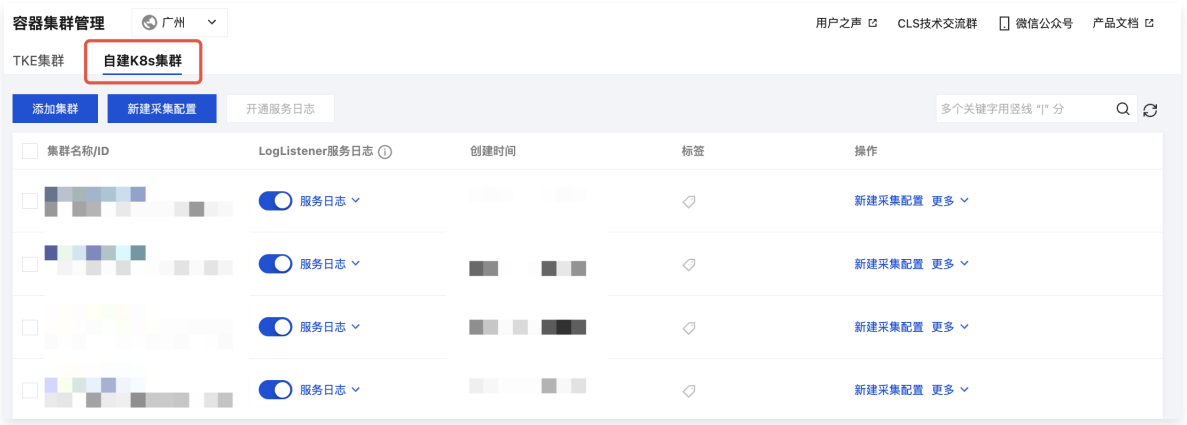
前提条件

- 在自建 K8s 集群上安装部署好采集器 LogListener，安装 LogListener 请参见 自建 K8s 集群安装 LogListener 。
- 配置好相应的日志上报权限，权限配置请参见 使用自建 K8s 上传数据 。

操作步骤

步骤1：选择集群

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**容器集群管理**，进入容器集群管理页面。
3. 在页面右上角选中**自建 K8s 集群**。



4. 若您已完成 自建 K8s 集群安装 LogListener，CLS 将会自动在您安装时指定的 CLS 地域下，创建一个集群。
5. 单击右侧**新建采集配置**，进入集群日志采集配置流程。



步骤2：配置日志主题

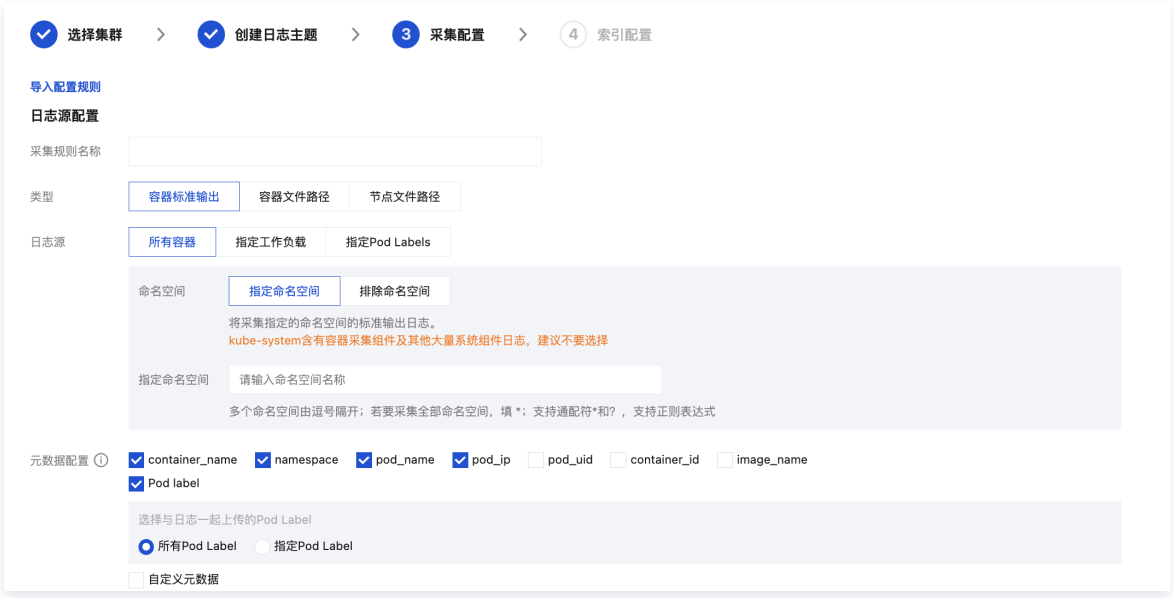
进入集群日志采集配置流程，在**创建日志主题**步骤中，您可选择已有日志主题或创建用于存储日志的日志主题。日志主题的相关信息请参考 日志主题 。



步骤3：配置采集规则

完成日志主题的配置后，单击下一步进入采集配置即可配置采集规则，配置信息如下：

- **日志源配置：**



  - **采集规则名称：您可以自定义日志采集规则名称。**
  - **采集类型：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

支持所有容器、指定工作负载、指定 Pod Labels 三种方式指定容器标准输出的采集日志源。

- **所有容器**：代表采集指定命名空间下的所有容器中的标准输出日志，如下图所示：



- **指定工作负载**：代表采集指定命名空间下，指定工作负载中，指定容器的标准输出日志，如下图所示：



- **指定 Pod Labels**：代表采集指定命名空间下，所有带有指定 Pod Label 的容器中的标准输出日志，如下图所示：



容器文件路径

> **注意：**
> “容器文件路径” **不能为软链接**，否则会导致软链接的实际路径在采集器的容器内不存在，采集日志失败。

支持指定工作负载、指定 Pod Labels 两种方式指定容器文件路径的采集日志源。

- **指定工作负载**：代表采集指定命名空间下、指定工作负载中、指定容器中的容器文件路径，如下图所示：



- **指定 Pod Labels：代表采集指定命名空间下，所有带有指定 Pod Label 的容器中的容器文件路径，如下图所示：**

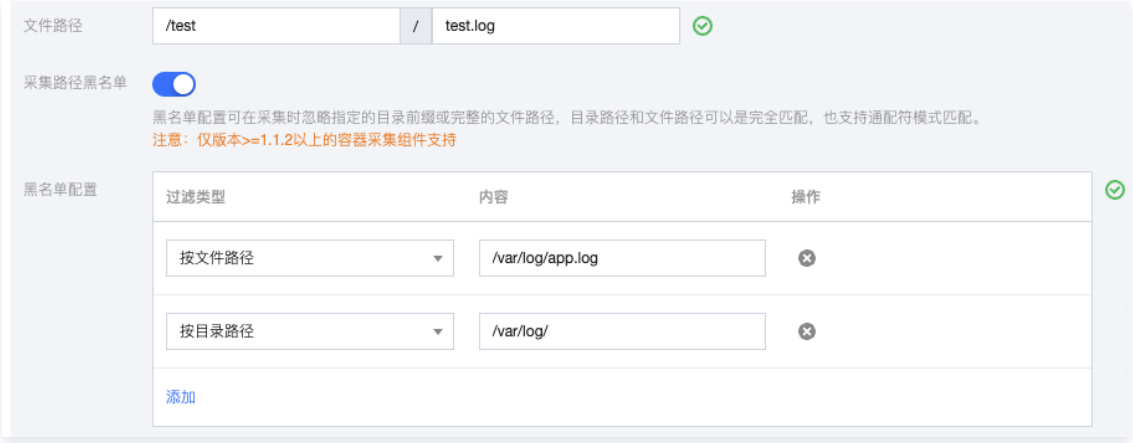


  - **容器文件路径由日志目录与文件名组成，日志目录前缀以 / 开头，文件名以非 / 开头，前缀和文件名均支持使用通配符 ？和 *，不支持逗号。/**/代表日志采集组件将监听所填前缀目录下所有层级中匹配的日志文件。多文件路径之间为或关系。例如：当容器文件路径为 `/opt/logs/*.log`，可以指定目录前缀为 `/opt/logs`，文件名为 `*.log` 的日志。**

⚠ 注意：

- **仅版本1.1.12及以上的容器采集组件支持多采集路径。**
- **仅升级至1.1.12及以上版本后创建的采集配置支持定义多采集路径。**
- **版本1.1.12以下创建的采集配置在升级至1.1.12后不支持配置多采集路径，需重新创建采集配置。**

  - **采集路径黑名单。 开启后可在采集时忽略指定的目录路径或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。**



- 采集黑名单分为两类过滤类型，且可以同时使用：
  - 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配符*或？，支持**路径模糊匹配。
  - 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配符*或？，支持**路径模糊匹配。

⚠ 注意：

- 需要容器日志采集组件1.1.2及以上版本。
- 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

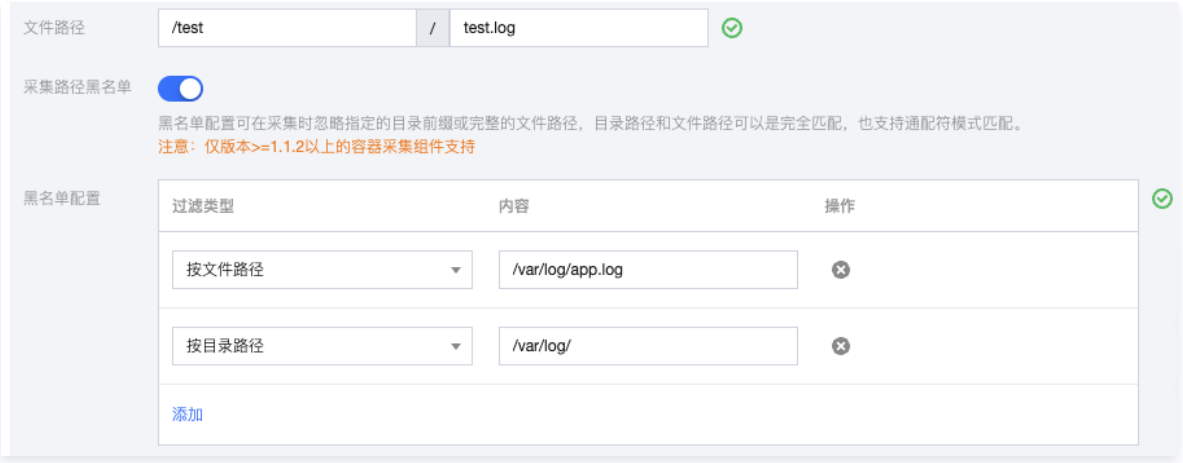
节点文件路径

节点文件路径由日志目录与文件名组成，日志目录前缀以 / 开头，文件名以非 / 开头，前缀和文件名均支持使用通配符 ? 和 *，不支持逗号。/**/代表日志采集组件将监听所填前缀目录下所有层级中匹配的日志文件。多文件路径之间为或关系。例如：当节点文件路径为 `/opt/logs/*.log`，可以指定目录前缀为 `/opt/logs`，文件名为 `*.log` 的日志。

⚠ 注意：

- 仅版本1.1.12及以上的采集组件支持多采集路径。
- 仅升级至1.1.12及以上版本后创建的采集配置支持定义多采集路径。
- 版本1.1.12以下创建的采集配置在升级至1.1.12后不支持配置多采集路径，需重新创建采集配置。

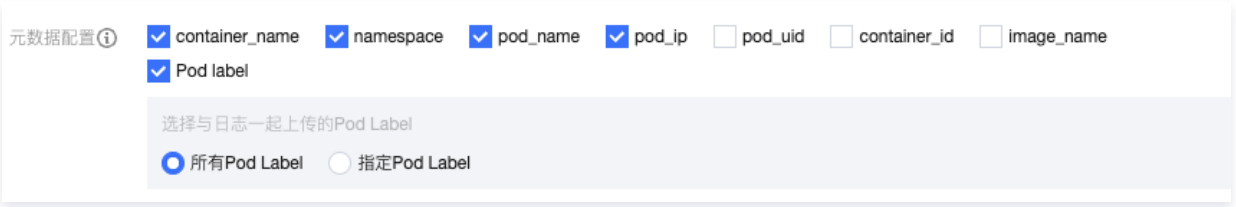
- 采集路径黑名单。 开启后可在采集时忽略指定的目录路径或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



- 采集黑名单分为两类过滤类型，且可以同时使用：
  - 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配符*或？，支持**路径模糊匹配。
  - 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配符*或？，支持**路径模糊匹配。

> **注意：**
> - 需要容器日志采集组件1.1.2及以上版本。
> - 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

- 元数据配置：



除了原始的日志内容， 日志服务还会带上容器或 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。您可以自行选择是否需要上报这些元数据，按需勾选上传。
容器或 kubernetes 相关的元数据请参见下方表格：

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。
pod_lable_{label name}	日志所属 pod 的 label（例如一个 pod 带有两个 label：app=nginx，env=prod，则在上传的日志会附带两个 metedata：pod_label_app:nginx，pod_label_env:prod）。

> **说明：**
> 如果想采集部分 podlabel，需要手动输入想要的 label key (可以输入多个，每输入一个以回车结束)，命中的话会采集。

- 解析规则配置：



- 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：增量采集指只采集文件内新增的内容
- 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：

单行全文格式

单行全文日志是指一行日志内容为一条完整的日志。日志服务在采集的时候，将使用换行符 \n 来作为一条日志的结束符。为了统一结构化管理，每条日志都会存在一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条日志原始数据为：

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

采集到日志服务的数据为：

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

多行全文格式

多行全文日志是指一条完整的日志数据可能跨占多行（例如 Java stacktrace）。在这种情况下，以换行符 \n 为日志的结束标识符就显得有些不合理，为了能让日志系统明确区分开每条日志，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.
多行全文也会设置一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条多行日志原始数据为：

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
    at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
    at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

首行正则表达式为如下：

```
\d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2},\d{3}\s.+
```

采集到日志服务的数据为：

```
__CONTENT__:2019-12-15 17:13:06,043 [main] ERROR
com.test.logging.FooFactory:\njava.lang.NullPointerException\n    at
com.test.logging.FooFactory.createFoo(FooFactory.java:15)\n    at
com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

单行-完全正则格式

单行完全正则格式通常用来处理结构化的日志，指将一条完整日志按正则方式提取多个 key-value 的日志解析模式。
假设一条日志原始数据为：

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703
"http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum"
"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0"  0.354 0.354
```

配置的正则表达式为如下：

```
(\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"
([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
```

采集到日志服务的数据为：

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

多行-完全正则格式

假设您的一条日志原始数据为：

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

版权所有：腾讯云计算（北京）有限责任公司

行首正则表达式为：

```
\[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*
```

配置的自定义正则表达式为：

```
\[(\d+-\d+-\w+:\d+:\d+,\d+)\]\s\[(\w+)\]\s(.*)
```

系统根据 `()` 捕获组提取对应的 key-value 后，您可以自定义每组的 key 名称如下所示：

```
time: 2018-10-01T10:30:01,000`
level: INFO`
msg: java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

JSON格式

假设您的一条 JSON 日志原始数据为：

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34
+0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost":"unix:/tmp/php-
cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","request":"POST
/event/dispatch HTTP/1.1","xff":"-","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0
(Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

分隔符

假设您的一条日志原始数据为：

```
10.20.20.10 - ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/1.1 ::: 127.0.0.1
::: 200 ::: 647 ::: 35 ::: http://127.0.0.1/
```

当日志解析的分隔符指定为 `:::`，该条日志会被分割成八个字段，并为这八个字段定义唯一的 key，如下所示：

```
IP: 10.20.20.10 -
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

组合解析

假设您的一条日志的原始数据为：

```
1571394459, http://127.0.0.1/my/course/4|10.135.46.111|200, status:DEAD,
```

自定义插件内容如下：

```
{
 "processors": [
   {
     "type": "processor_split_delimiter",
     "detail": {
       "Delimiter": ",",
       "ExtractKeys": [ "time", "msg1","msg2"]
     },
     "processors": [
       {
         "type": "processor_timeformat",
         "detail": {
           "KeepSource": true,
           "TimeFormat": "%s",
           "SourceKey": "time"
          }
       },
       {
         "type": "processor_split_delimiter",
         "detail": {
           "KeepSource": false,
           "Delimiter": "|",
           "SourceKey": "msg1",
           "ExtractKeys": [ "submsg1","submsg2","submsg3"]
          },
          "processors": []
       },
       {
         "type": "processor_split_key_value",
         "detail": {
           "KeepSource": false,
           "Delimiter": ": ",
           "SourceKey": "msg2"
         }
       }
     ]
   }
 ]
}
```

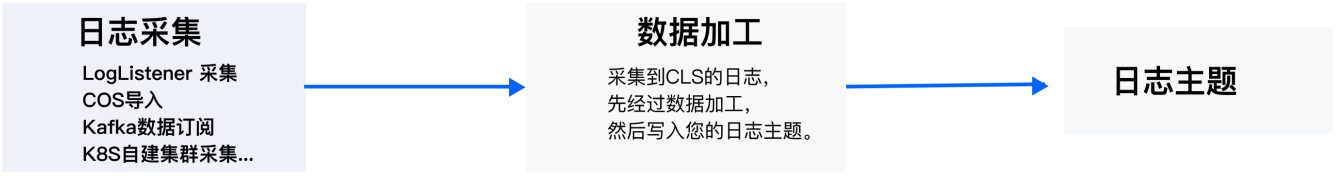
经过日志服务结构化处理后，该条日志将变为如下：

```
time: 1571394459
submsg1: http://127.0.0.1/my/course/4
submsg2: 10.135.46.111
submsg3: 200
status: DEAD
```

- 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。

数据加工

如您需要对采集到的 CLS 日志进一步处理，如结构化、脱敏、过滤，然后再写入日志主题，您可点击页面下方的**数据加工**，添加数据加工，最后 配置索引 。



说明：

- 数据加工相关操作请参考 创建加工任务 文档中【前置数据加工】TAB 页面；
- 数据加工脚本的编写请参考 数据加工函数总览，或者 加工实战案例；
- 数据加工会产生费用，详情请参见 计费概述 。

步骤4：配置索引

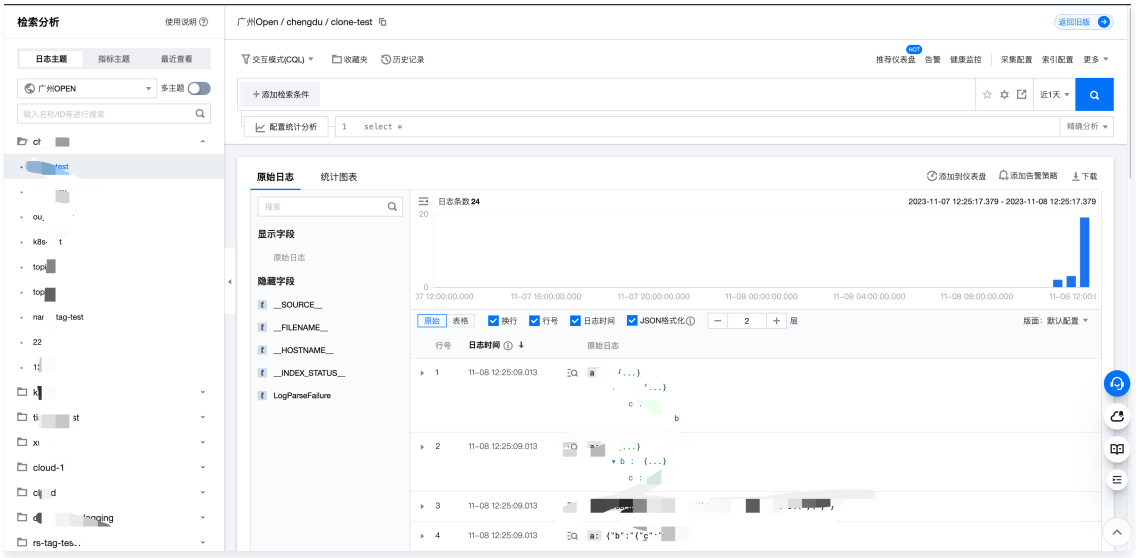
完成采集规则配置后， 单击**下一步进入索引配置**，在配置页面中建议直接使用推荐配置，完成索引的自动配置。



如需手动配置索引，可关闭使用推荐配置，参照 全文索引 及 键值索引 说明完成详细配置。

步骤5：检索日志

至此， 即完成了自建 K8s 集群业务日志采集的所有部署。您可以前往 CLS 控制台 > 检索分析 查看采集上来的日志。



使用 CRD 配置自建 K8S 集群日志采集

最近更新时间：2025-04-03 16:44:23

操作场景

日志服务（Cloud Log Service，CLS）支持采集自建 K8S 集群上的日志，在进行日志采集前，需要在 K8S 自建集群上通过 CRD 定义日志采集配置（LogConfig），并部署安装 Log-Provisioner、Log-Agent，以及 LogListener。针对使用腾讯云容器服务（Tencent Kubernetes Engine，TKE）的用户，可参见 采集 TKE K8S 集群日志 文档，通过控制台快速接入并使用日志服务。

前提条件

1. 在自建 K8S 集群上安装部署好采集器 LogListener，安装 LogListener 请参见 自建 K8S 集群安装 LogListener。
2. 配置好相应的日志上报权限，权限配置请参见 使用自建 K8S 上传数据。
3. 已获取日志主题所在地域（region）。详细 CLS 地域列表请参见 可用地域。

K8S 日志采集原理

K8S 集群上部署日志采集主要涉及 Log-Provisioner、Log-Agent、LogListener 三个组件和一个 LogConfig 采集配置。

- LogConfig：日志采集配置，定义了日志在哪里被采集，采集后如何解析，以及解析后投递至哪个 CLS 日志主题中。
- Log-Agent：监听 LogConfig 和节点上容器的变化，动态计算容器中的日志文件在节点宿主机上的实际位置。
- Log-Provisioner：将 LogConfig 中定义日志采集配置信息同步至 CLS。
- LogListener：采集节点宿主机上的相应日志文件内容，解析并上传至 CLS。

操作流程

1. 自建 K8S 集群安装 LogListener
2. 定义 LogConfig 对象
3. 创建 LogConfig 对象
4. 查看 LogConfig

操作步骤

步骤一： 自建 K8S 集群安装 LogListener

首先，您需要在自建 Kubernetes 集群上安装 LogListener 组件，从而将日志收集到 CLS。详情请参见 自建 K8S 集群安装 LogListener。

步骤二：定义 LogConfig 对象

通过 CRD 定义 LogConfig 对象中的日志采集配置。以 Master 节点路径/usr/local/为例，使用 wget 命令下载 LogConfig.yaml CRD 声明文件。

```
wget https://mirrors.tencent.com/install/cls/k8s/LogConfig.yaml
```

LogConfig.yaml 声明文件主要分为如下两部分：

- clsDetail：日志采集配置。
- inputDetail：日志源配置。

```
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:                               ## 日志源配置
    ...
```

clsDetail（CLS 采集配置）字段说明

```
  clsDetail:
    ### 指定日志集和日志主题
    logsetName: test                          ## CLS 日志集的 name，非必填。若无该 logsetName 的日志集，会自动创建；若有，会在该日志集下创建日志主题
    topicName: test                           ## CLS 日志主题的 name，非必填。若在 logsetName 下没有该name的日志主题，会自动创建。
                                              ## logsetName 和 topicName 指定之后，不指定 logsetId 和 topicId，会尝试自动创建日志主题；否则不会去创建 topic。
    logsetId: xxxxxx-xx-xx-xx-xxxxxxxx        ## CLS 日志集的 ID，非必填，
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx         ## CLS 日志主题的 ID，非必填。如果不填 topicId，则必须填 logsetName 和 topicName，并且会尝试去自动创建 topic，并将创建后的 topic 的 topicId 填在该字段

    ### 自动创建日志主题时，定义日志主题配置。 定义后不可修改，修改不生效
    period: 30                                ## 非必填，生命周期，单位天，可取值范围1~3600，低频存储取值范围7~3600天；取值为3640时代表永久保存
    storageType: hot                          ## 非必填，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。
    hotPeriod: 7                              ## 非必填，沉降周期，单位天。可取值范围7~3600。仅在 storageType:hot 时生效，hotPeriod 需要大于 period，沉降的周期为 Period-hotPeriod
    partitionCount: 1                         ## 非必填，日志主题分区个数。默认创建1个，最大支持创建10个分区。
    autoSplit: "true"                         ## 非必填，是否开启自动分裂，默认值为 true。值必须以双引号""包裹
    maxSplitPartitions: 50                    ## 非必填，开启自动分裂后的最大分裂数量。默认值为50
    tags:                                     ## 非必填，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

    ### 定义采集规则
    logType: json_log                         ## 日志解析格式，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 代表单行完全正则格式，multiline_fullregex_log 代表多行完全正则格式，user_define_log 代表组合解析。默认为 minimalist_log
    excludePaths:                             ## 非必填，采集黑名单路径列表。仅在 inputDetail 为 containerFile 或 hostFile 时生效
    - type: File                              ## 类型，选填 File 或 Path
      value: /xx/xx/xx/xx.log                 ## type 对应的值
    userDefineRule: xxxxxx                    ## 非必填，用户自定义采集规则，Json 格式序列化的字符串

    ### 提取、过滤规则
    extractRule:                              ## 非必填，提取、过滤规则。如果设置了 ExtractRule，则必须设置 LogType，详情参考 extractRule 对象说明
      timeKey: xxx                            ## 时间字段的 key 名字，time_key 和 time_format 必须成对出现
      timeFormat: xxx                         ## 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: xxx                          ## 分隔符类型日志的分隔符，只有 log_type 为 delimiter_log 时有效
      logRegex: xxx                           ## 整条日志匹配规则，只有 log_type 为 fullregex_log 时有效
      beginningRegex: xxx                     ## 行首正则匹配规则，只有 log_type 为 multiline_log 或 fullregex_log 时有效
      keys:                                   ## 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
      - a
      - b
      filterKeys:                             ## 与 filterRegex 要对应，filterKeys 是数组，表示需要过滤的日志的 key
      - a
      - b
      filterRegex:                            ## 与 filterKeys 要对应，filterRegex 是正则字符串数组，key 对应的过滤规则 regex。
      - aRegx
      - bRegx
```

```
      advancedFilters:                          ## 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
      - key: level                              ## 需要过滤的 Key
        rule: 0                                 ## 过滤规则，支持的数值枚举为：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。
      unMatchUpload: "false"                    ## 解析失败日志是否上传，true 表示上传，false 表示不上传。默认不上传
      unMatchedKey: "parseFailed"               ## 解析失败的日志上传 CLS 的 key 名称
      backtracking: "-1"                        ## “-1”或者“0”，-1表示全量采集，0表示增量采集，其他大于0的数字表示增
量采集模式下的回溯数据量。默认全量。
      isGBK: "true"                             ## "true"或者"false"，默认"false"非 jbk
      jsonStandard: "true"                      ## "true"或者"false"，默认"true"标准 json

    ###  高级采集配置
    advancedConfig:                             ## 非必填
      ClsAgentMaxDepth: 1                       ## 采集器采集最大目录深度，大于等于0的整数，0表示当前目录。
      ClsAgentFileTimeout: 60                   ## 文件超时属性，即超过该指定时间的日志文件没有被修改，则采集器不在监
控。 该参数需要大于等于0的整数，单位秒。0为不超时，全部采集。
      ClsAgentParseFailMerge: true              ## 合并解析失败日志，取值范围：true 或 false

    ### 创建 topic 时可自定义全文索引
    ### 仅在自动创建日志主题时支持定义， 不支持修改已存在日志主题的索引
    fullTextIndex:                              ## 非必填，全文索引设置
      status: "on"                              ## 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 on 打开，
不设置其他参数，则 caseSensitive 和 tokenizer 都会是默认值；如果设置 off 关闭，则不打开全文索引
      tokenizer: "@&()='\",;:<>[]{}/ \n\t\r" ## 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
      caseSensitive: false                      ## 是否大小写敏感
      containZH: false                          ## 是否包含中文

    ### 创建 topic 时可自定义TAG和键值索引设置
    indexStatus: "on"                           ## 非必填，TAG 索引和键值索引设置开关，如果不设置，则默认创建
pod_name/namespace/container_name三个TAG索引;off则表示不设置键值索引
    autoIndex: "off"                            ## 非必填， off 则表示不设置自动配置索引，on 标识开启自动配置索引
    indexs:                                     ## 非必填。
    - indexName: xxx                            ## 需要配置键值索引或者 TAG 索引的字段；Pod 相关的
namespace/pod_name/pod_ip/pod_uid/container_id/container_name/image_name/cluster_id 八个元数据字段和
pod_label_前缀字段会设置为 TAG 索引，其他字段会设置为键值索引。
      indexType: "text"                         ## 非必填。字段类型，目前支持的类型有：long、text、double。非必填
      tokenizer: "@&()='\",;:<>[]{}/ \n\t\r"    ## 非必填。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: true                             ## 非必填。字段是否开启分析功能。true/false 字段是否开启分析功能，如
果是日志主题是低频日志主题，sqlFlag 选项不能被打开。
      containZH: false                          ## 非必填。是否包含中文。
```

日志采集规则配置示例

单行全文格式

单行全文日志是指一行日志内容为一条完整的日志。日志服务在采集的时候，将使用换行符 \n 来作为一条日志日志的结束符。为了统一结构化管理，每条日志都会存在一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条日志原始数据为：

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

LogConfig 配置参考示例如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test                                    ## 采集配置名称
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # 单行日志
    logType: minimalist_log
```

采集到日志服务的数据为：

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

多行全文格式

多行全文日志是指一条完整的日志数据可能跨占多行（例如 Java stacktrace）。在这种情况下，以换行符 \n 为日志的结束标识符就显得有些不合理，为了能让日志系统明确区分开每条日志，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。
多行全文也会设置一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条多行日志原始数据为：

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
    at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
    at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test                                    ## 采集配置名称
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # 多行日志
    logType: multiline_log
    extractRule:
      # 只有以日期时间开头的行才被认为是新一条日志的开头，否则就添加换行符\n并追加到当前日志的尾部
      beginningRegex: \d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2},\d{3}\s.+
```

采集到日志服务的数据为：

```
__CONTENT__:2019-12-15 17:13:06,043 [main] ERROR
com.test.logging.FooFactory:\njava.lang.NullPointerException\n    at
com.test.logging.FooFactory.createFoo(FooFactory.java:15)\n    at
com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

单行-完全正则格式

单行完全正则格式通常用来处理结构化的日志，指将一条完整日志按正则方式提取多个 key-value 的日志解析模式。
假设一条日志原始数据为：

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703
"http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum"
"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0"  0.354 0.354
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test                                    ## 采集配置名称
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # 完全正则格式
    logType: fullregex_log
    extractRule:
      # 正则表达式，会根据()捕获组提取对应的value
      logRegex: (\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"
(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
      beginningRegex: (\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"
(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
      # 提取的key列表，与提取的value的一一对应
      keys:
['remote_addr','time_local','request_method','request_url','http_protocol','http_host','status','reque
st_length','body_bytes_sent','http_referer','http_user_agent','request_time','upstream_response_time']
```

采集到日志服务的数据为：

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

多行-完全正则格式

多行-完全正则模式适用于日志文本中一条完整的日志数据跨占多行（例如 Java 程序日志），可按正则表达式提取为多个 key-value 键值的日志解析模式。若不需要提取 key-value，请参阅多行全文格式进行配置。
假设一条日志原始数据为：

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
```

```
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test                                    ## 采集配置名称
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
        #多行-完全正则格式
        logType: multiline_fullregex_log
        extractRule:
          #行首完全正则表达式，只有以日期时间开头的行才被认为是新一条日志的开头，否则就添加换行符\n并追加到当前日志的尾
部
          beginningRegex: \[\d+-\d+-\w+:\d+:\d+,\d+\]\s\[\w+\]\s.*
          #正则表达式，会根据()捕获组提取对应的value
          logRegex: \[(\d+-\d+-\w+:\d+:\d+,\d+)\]\s\[(\w+)\]\s(.*)
          # 提取的 key 列表，与提取的 value 的一一对应
          keys: ['time','level','msg']
```

根据提取的 key，采集到日志服务的数据为：

```
time:  2018-10-01T10:30:01,000`
level:  INFO`
msg: java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

JSON 格式

JSON 格式日志会自动提取首层的 key 作为对应字段名，首层的 value 作为对应的字段值，以该方式将整条日志进行结构化处理，每条完整的日志以换行符\n为结束标识符。
假设一条 JSON 日志原始数据为：

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34
+0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost":"unix:/tmp/php-
cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","request":"POST
/event/dispatch HTTP/1.1","xff":"-","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0
(Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test                                    ## 采集配置名称
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # JSON格式日志
    logType: json_log
```

采集到日志服务的数据为：

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

分隔符格式

分隔符日志是指一条日志数据可以根据指定的分隔符将整条日志进行结构化处理，每条完整的日志以换行符 \n 为结束标识符。日志服务在进行分隔符格式日志处理时，您需要为每个分开的字段定义唯一的 key。
假设您的一条日志原始数据为：

```
10.20.20.10 ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/1.1 ::: 127.0.0.1 :::
200 ::: 647 ::: 35 ::: http://127.0.0.1/
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test                                    ## 采集配置名称
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # 分隔符日志
    logType: delimiter_log
    extractRule:
      # 分隔符
      delimiter: ':::'
      # 提取的key列表，与被分割的字段一一对应
      keys: ['IP','time','request','host','status','length','bytes','referer']
```

采集到日志服务的数据为：

```
IP: 10.20.20.10
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

组合解析

假设您的一条日志的原始数据为：

```
1571394459, http://127.0.0.1/my/course/4|10.135.46.111|200, status:DEAD,
```

自定义插件内容如下：

```
{
 "processors": [
   {
     "type": "processor_split_delimiter",
     "detail": {
       "Delimiter": ",",
       "ExtractKeys": [ "time", "msg1","msg2"]
     },
     "processors": [
       {
         "type": "processor_timeformat",
         "detail": {
           "KeepSource": true,
           "TimeFormat": "%s",
           "SourceKey": "time"
          }
       },
       {
         "type": "processor_split_delimiter",
         "detail": {
           "KeepSource": false,
           "Delimiter": "|",
           "SourceKey": "msg1",
           "ExtractKeys": [ "submsg1","submsg2","submsg3"]
          },
          "processors": []
       },
       {
         "type": "processor_split_key_value",
         "detail": {
           "KeepSource": false,
           "Delimiter": ": ",
           "SourceKey": "msg2"
         }
       }
     ]
   }
 ]
}
```

LogConfig 配置的参考如下：

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test                                    ## 采集配置名称
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # 分隔符日志
```

```
    logType: user_define_log
    userDefineRule: "{ \"processors\": [ { \"type\": \"processor_split_delimiter\", \"detail\": {
\"Delimiter\": \",\", \"ExtractKeys\": [ \"time\", \"msg1\",\"msg2\"] }, \"processors\": [ { \"type\":
\"processor_timeformat\", \"detail\": { \"KeepSource\": true, \"TimeFormat\": \"%s\", \"SourceKey\":
\"time\" } }, { \"type\": \"processor_split_delimiter\", \"detail\": { \"KeepSource\": false,
\"Delimiter\": \"|\", \"SourceKey\": \"msg1\", \"ExtractKeys\": [ \"submsg1\",\"submsg2\",\"submsg3\"]
}, \"processors\": [] }, { \"type\": \"processor_split_key_value\", \"detail\": { \"KeepSource\":
false, \"Delimiter\": \": \", \"SourceKey\": \"msg2\" } } ] } ] }"
```

经过日志服务结构化处理后，该条日志将变为如下：

```
time: 1571394459
submsg1: http://127.0.0.1/my/course/4
submsg2: 10.135.46.111
submsg3: 200
status: DEAD
```

inputDetail（日志源）字段说明

```
  inputDetail:
    type: container_stdout                          ## 指定采集日志的类型，包括 container_stdout（容器标准输出）、
container_file（容器文件）、host_file（主机文件）
    containerStdout:                                ## 容器标准输出配置，仅在 type:container_stdout 时生效
      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
      containerOperator: in                         ## container 选择方式，包含填in，排除填notin
      container: xxx                                ## 指定采集或不采集日志的容器名，支持填写多个，逗号隔开。若
containerOperator 为 in，则可填*，代表采集所有容器
      includeLabels:                                ## 采集包含指定 label 的 Pod，与 workload 不能同时指定
        key: value1                                 ## 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
      excludeLabels:                                ## 采集不包含包含指定 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
      metadataLabels:                               ## 指定具体哪些 pod label 被当做元数据采集，如果不指定，则采集所
有 pod label 为元数据
      - label1
      metadataContainer:                            ## 指定具体哪些容器环境相关元数据被采集，如果不指定，则采集所有容器
环境相关元数据（namespace,pod_name,pod_ip,pod_uid,container_id,container_name,image_name）
      - namespace
      customLabels:                                 ## 用户自定义 metadata
        label: l1
      workloads:                                    ## 采集指定命名空间 -> 指定工作负载类型中 -> 指定工作负载 -> 指定
容器中的日志
      - container: xxx                              ## 要采集的容器名，如果不指定，代表 workload Pod 中的所有容器
        containerOperator: in                       ## container 选择方式，包含填 in，排除填 notin
        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:                               ## 容器内文件配置，仅在 type:container_file 时生效
      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
      containerOperator: in                      ## container 选择方式，包含填 in，排除填 notin
      container: xxx                             ## 指定采集或不采集日志的容器名，支持填写多个，逗号隔开。 若
containerOperator 为 in，则可填*，代表采集所有容器。若制定了 workload 类型和名称，此处可定义指定工作负载中容器的名称
      logPath: /var/logs                         ## 日志文件夹，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
      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

      ## 指定容器内多个日志路径和文件，该参数和 logPath、filePattern 不兼容，只能指定 filePaths 或者(logPath、
filePattern)
      ## 如果从(logPath 和 filePattern) -> filePaths 会触发日志重采；相反从 filePaths->logPath 和 filePattern)
也一样
      filePaths:
      - file: /var/logs                          ## 日志文件夹，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
        path: 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
      - file: /var/logs1
        path: app_*.log

      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
      excludeLabels:                             ## 采集不包含包含指定 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
      metadataLabels:                            ## 指定具体哪些 pod label 被当做元数据采集，如果不指定，则采集所
有 pod label 为元数据
      - namespace
      metadataContainer:                         ## 指定具体哪些容器环境相关元数据被采集，如果不指定，则采集所有容器
环境相关元数据（namespace,pod_name,pod_ip,pod_uid,container_id,container_name,image_name）
      customLabels:                              ## 用户自定义 metadata
        key: value
      workload:                                  ## 采集指定命名空间 -> 指定工作负载类型中 -> 指定工作负载中的所有
容器。若要指定工作负载中的容器，需在外层 container 中指定
        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

    hostFile:                                    ## 节点文件路径，仅在 type:host_file 时生效
      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，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

      ## 指定容器内多个日志路径和文件，该参数和 logPath、filePattern 不兼容，只能指定 filePaths 或者(logPath、
filePattern)
      ## 如果从(logPath 和 filePattern) -> filePaths 会触发日志重采；相反从 filePaths->logPath 和 filePattern)
也一样
```

```
      filePaths:
      - file: /var/logs                          ## 日志文件夹，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
        path: 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
      - file: /var/logs1
        path: app_*.log

      customLabels:                              ## 用户自定义 metadata
        label1: v1
```

日志源配置示例

容器标准输出

示例1：采集 default 命名空间中的所有容器的标准输出。

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test                                     ## 采集配置名称
spec:
  inputDetail:
    type: container_stdout
    containerStdout:
      namespace: default
      allContainers: true
 ...
```

示例2：采集 production 命名空间中属于 ingress-gateway deployment 的 pod 中的所有容器的标准输出。

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

示例3：采集 production 命名空间中 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:
    type: container_stdout
    containerStdout:
      namespace: production
      allContainers: false
      includeLabels:
        k8s-app: nginx
  ...
```

示例4：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:
    type: container_stdout
    containerStdout:
      allContainers: false
      containerOperator: in
      container: ingress
      workloads:
      - namespace: production
        name: ingress-gateway
        kind: deployment
  ...
```

容器文件

示例1：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                                    ## 采集配置名称
spec:
  inputDetail:
    type: container_file
    containerFile:
      namespace: production
      workload:
        name: ingress-gateway
        kind: deployment
      container: nginx
      filePaths:
      - file: access.log
        path: /data/nginx/log
  ...
```

示例2：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:
    type: container_file
    containerFile:
      namespace: production
      includeLabels:
        k8s-app: ingress-gateway
      container: nginx
      filePaths:
      - file: access.log
        path: /data/nginx/log
```

版权所有：腾讯云计算（北京）有限责任公司

```
  ...
```

主机文件

示例：采集主机 /data/ 路径下所有 .log 文件

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test                                    ## 采集配置名称
spec:
  inputDetail:
    type: host_file
    hostFile:
      logPath: /data
      filePattern: *.log
  ...
```

步骤三：创建 LogConfig 对象

步骤2 定义了 LogConfig.yaml 声明文件，我们可以基于 LogConfig.yaml 声明文件，使用 kubectl 命令创建 LogConfig 对象。

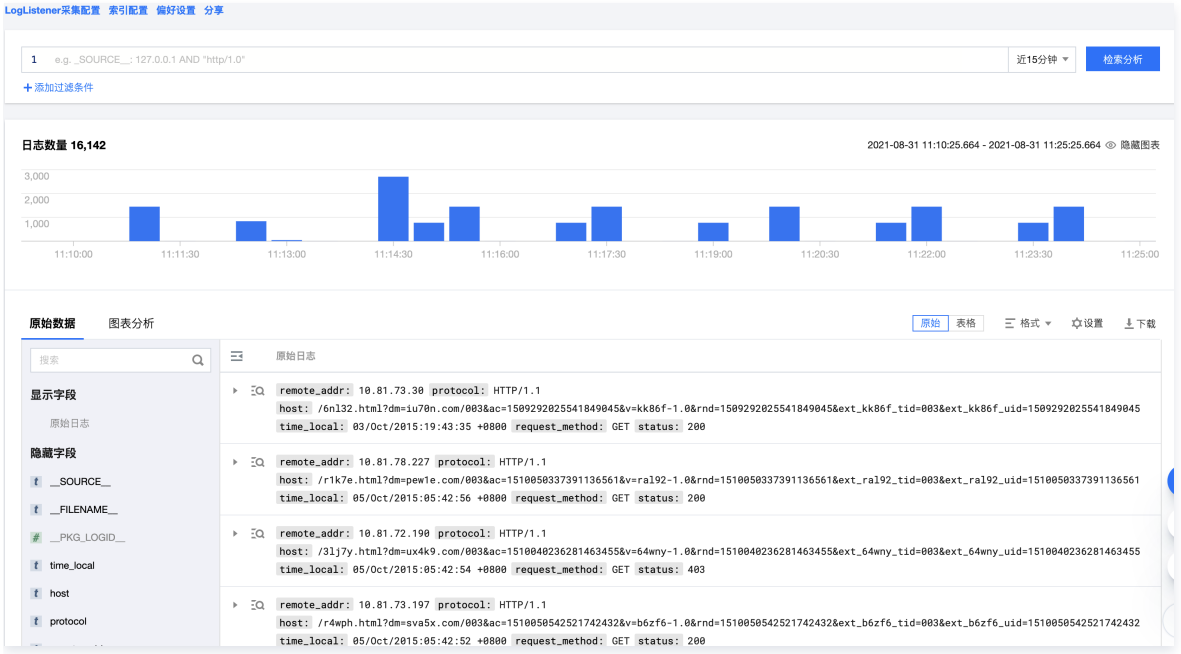
```
kubectl create -f /usr/local/LogConfig.yaml
```

步骤四：查看 LogConfig

```
kubectl get logconfig
```

后续操作

至此， 即完成了集群日志采集的所有部署。您可以前往 CLS 控制台 > 检索分析 查看采集上来的日志。



采集 TKE K8s 集群日志

最近更新时间：2025-06-11 09:55:12

本文将介绍如何在控制台配置腾讯云容器服务（Tencent Kubernetes Engine，TKE）K8s 集群的日志采集规则并投递到 腾讯云日志服务 CLS。若需使用 CRD 配置 TKE 的 K8s 集群投递日志，可参考 TKE 场景使用 CRD 进行日志采集配置指导。

使用场景

TKE K8s 业务日志采集功能是为用户提供的腾讯云 TKE Kubernetes 集群内的业务日志采集工具，可以将集群内的业务日志、审计日志和事件日志发送至 腾讯云日志服务 CLS。
日志采集功能需要为集群安装日志采集组件并配置采集规则。日志采集组件安装后，日志采集 Agent 会在集群内以 DaemonSet 的形式运行，并根据用户通过日志采集规则配置的采集源、CLS 日志主题和日志解析方式，从采集源进行日志采集，将日志内容发送到日志消费端。您可根据以下操作安装并配置日志采集。

前提条件

已在 TKE控制台 创建集群。CLS 支持采集 TKE 的标准集群、弹性集群、边缘集群。

集群业务日志采集

步骤1：选择集群

1. 登录 日志服务控制台。
2. 在左侧导航栏中，单击**容器集群管理**，进入容器集群管理页面。
3. 在页面右上角选中 **TKE 集群**。

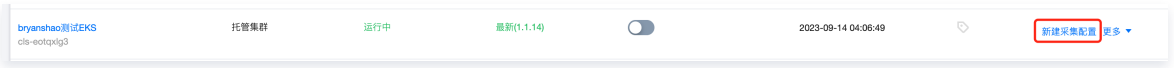


4. 选择 TKE 集群所在地域， 并找到目标采集集群。
5. 若采集组件状态为**未安装**，单击**安装**，安装日志采集组件。



> **注意：**
> 安装日志采集组件将在集群 kube-system 命名空间下，以 DaemonSet 的方式部署一个 tke-log-agent 的 pod 和一个 cls-provisioner 的 pod。请为每个节点至少预留0.1核16Mib以上的可用资源。

6. 若采集组件状态为**最新**，单击右侧**新建采集配置**，进入集群日志采集配置流程。



步骤2：配置日志主题

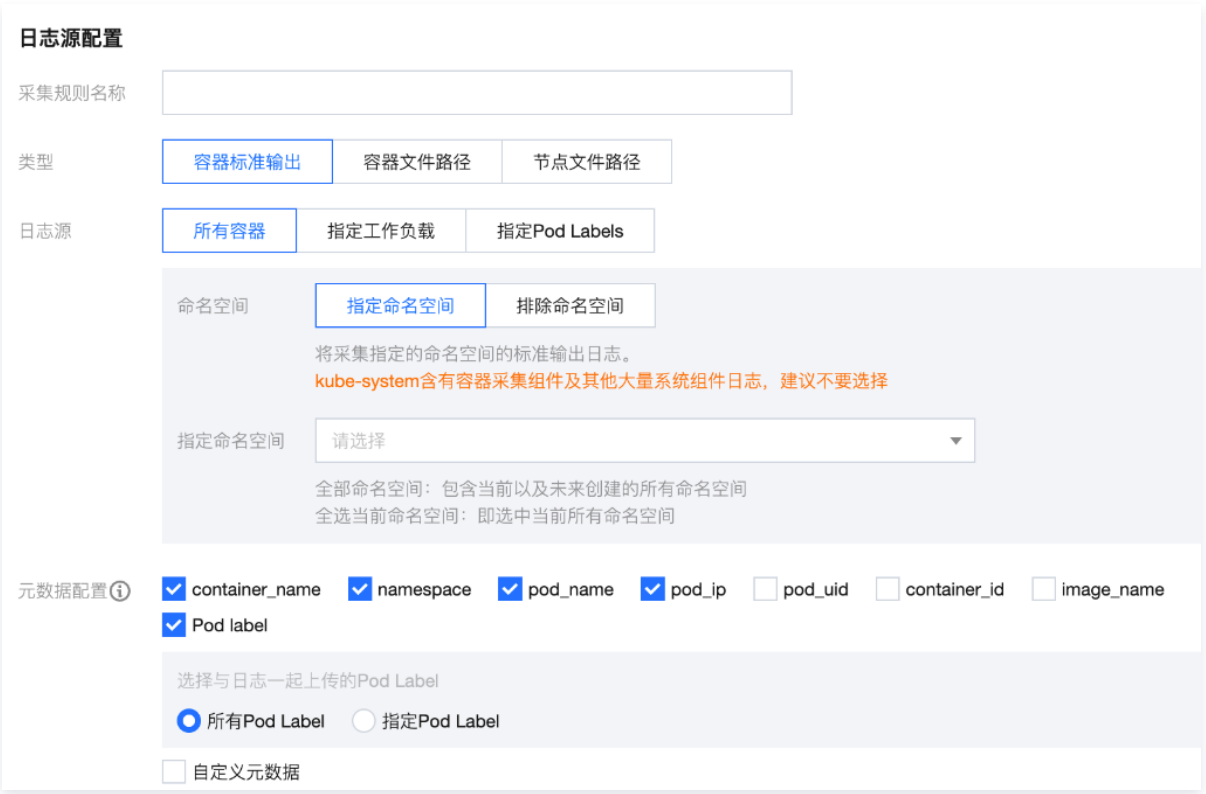
进入集群日志采集配置流程，在**创建日志主题**步骤中，您可选择已有或创建用于存储日志的日志主题。关于日志主题请参考 日志主题 。



步骤3：配置采集规则

完成日志主题的选择后，单击**下一步进入采集配置**即可配置采集规则，配置信息如下：

- 日志源配置



- **采集规则名称：您可以自定义日志采集规则名称。**
- **采集类型：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。**

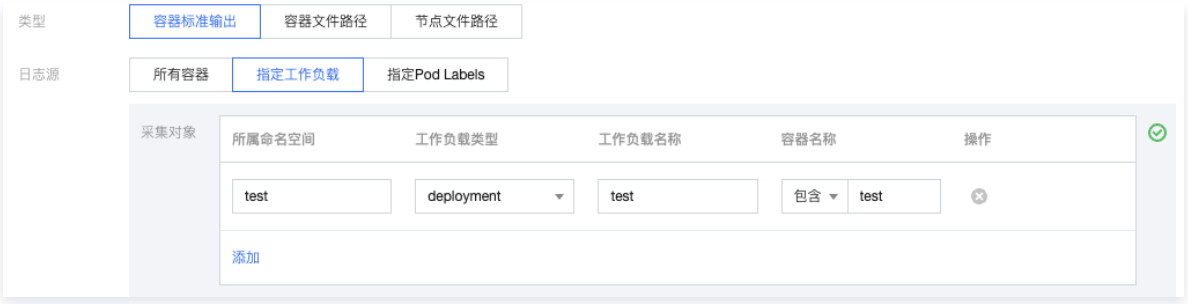
容器标准输出

支持所有容器、指定工作负载、指定 Pod Lables 三种方式指定容器标准输出的采集日志源。

- **所有容器：代表采集指定命名空间下的所有容器中的标准输出日志，如下图所示：**



- **指定工作负载：代表采集指定命名空间下，指定工作负载中，指定容器的标准输出日志如下图所示：**



- **指定 Pod Lables**：代表采集指定命名空间下，所有带有指定Pod Label的容器中的标准输出日志，如下图所示：



容器文件路径

> **注意：**
> “容器文件路径” **不能为软链接**，否则会导致软链接的实际路径在采集器的容器内不存在，采集日志失败。

支持指定工作负载、指定 Pod Lables 两种方式指定容器文件路径的采集日志源。

- **指定工作负载**：代表采集指定命名空间下、指定工作负载中、指定容器中的容器文件路径，如下图所示：



- **指定 Pod Lables**：代表采集指定命名空间下，所有带有指定 Pod Label 的容器中的容器文件路径，如下图所示：

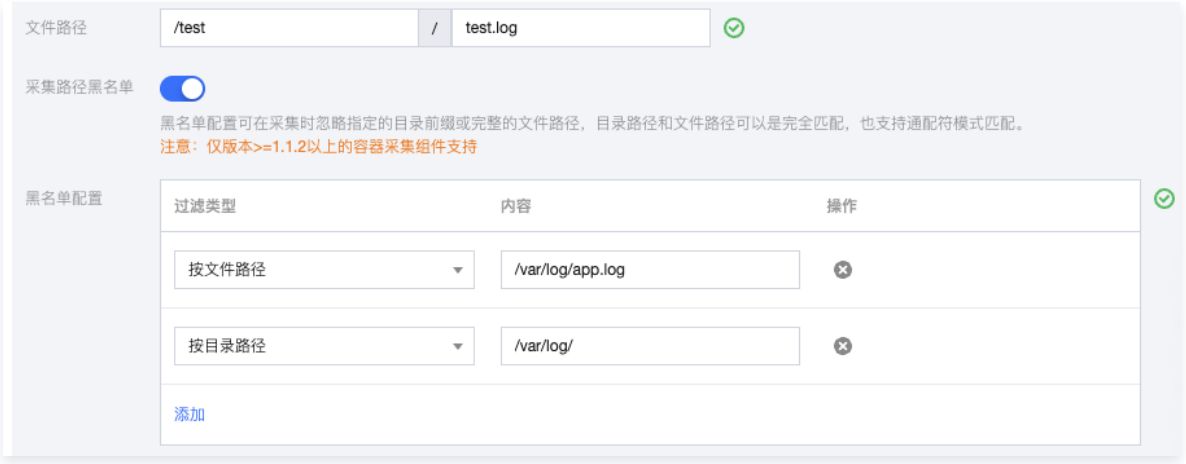


- 容器文件路径由日志目录与文件名组成，日志目录前缀以 / 开头，文件名以非 / 开头，前缀和文件名均支持使用通配符 ？和 *，不支持逗号。/**/代表日志采集组件将监听所填前缀目录下的所有层级匹配上的日志文件。多文件路径之间为或关系。例如：当容器文件路径为 `/opt/logs/*.log`，可以指定目录前缀为 `/opt/logs`，文件名为 `*.log` 的日志。

注意：

- 仅版本1.1.12及以上的容器采集组件支持多采集路径。
- 仅升级至1.1.12及以上版本后创建的采集配置支持定义多采集路径。
- 版本1.1.12以下创建的采集配置在升级至1.1.12后不支持配置多采集路径，需重新创建采集配置。

- 采集路径黑名单。 开启后可在采集时忽略指定的目录路径或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



- 采集黑名单分为两类过滤类型，且可以同时使用：
  - 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配符*或？，支持**路径模糊匹配。
  - 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配符*或？，支持**路径模糊匹配。

注意：

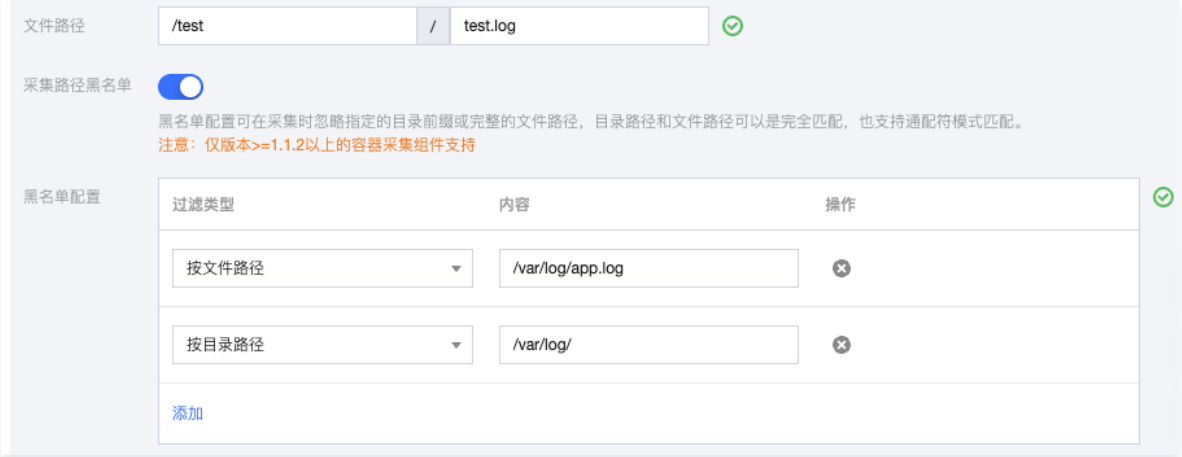
- 需要容器日志采集组件1.1.2及以上版本。
- 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

节点文件路径

节点文件路径由日志目录与文件名组成，日志目录前缀以 / 开头，文件名以非 / 开头，前缀和文件名均支持使用通配符？和 *，不支持逗号。/**/代表日志采集组件将监听所填前缀目录下的所有层级匹配上的日志文件。多文件路径之间为或关系。例如：当节点文件路径为 `/opt/logs/*.log`，可以指定目录前缀为 `/opt/logs`，文件名为 `*.log` 的日志。

> **注意：**
> - 仅版本1.1.12及以上的采集组件支持多采集路径。
> - 仅升级至1.1.12及以上版本后创建的采集配置支持定义多采集路径。
> - 版本1.1.12以下创建的采集配置在升级至1.1.12后不支持配置多采集路径，需重新创建采集配置。

采集路径黑名单。 开启后可在采集时忽略指定的目录路径或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。

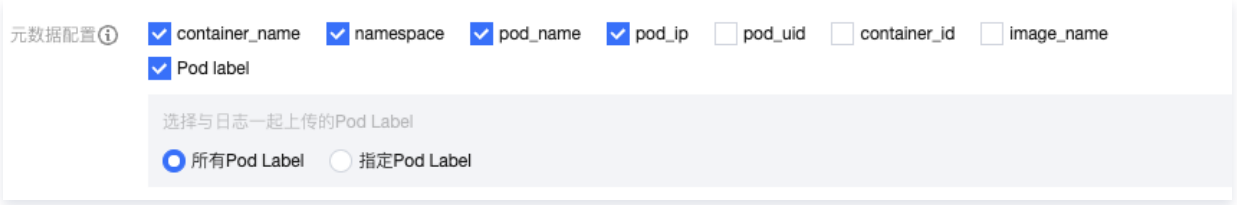


采集黑名单分为两类过滤类型，且可以同时使用：

- 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配符*或？，支持**路径模糊匹配。
- 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配符*或？，支持**路径模糊匹配。

> **注意：**
> - 需要容器日志采集组件1.1.2及以上版本。
> - 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

- 元数据配置



除了原始的日志内容， 日志服务还会带上容器或 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。您可以自行选择是否需要上报这些元数据，按需勾选上传。
容器或 kubernetes 相关的元数据请参见下方表格：

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。
pod_lable_{label name}	日志所属 pod 的 label（例如一个 pod 带有两个 label：app=nginx，env=prod，则在上传的日志会附带两个 metedata：pod_label_app:nginx，pod_label_env:prod）。

说明：
如果想采集部分 podlabel，需要手动输入想要的 label key (可以输入多个，每输入一个以回车结束)，命中的话会采集。

- **解析规则配置**



  - **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：增量采集指只采集文件内新增的内容**
  - **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：**

单行全文格式

单行全文日志是指一行日志内容为一条完整的日志。日志服务在采集的时候，将使用换行符 \n 来作为一条日志的结束符。为了统一结构化管理，每条日志都会存在一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条日志原始数据为：

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

采集到日志服务的数据为：

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

多行全文格式

多行全文日志是指一条完整的日志数据可能跨占多行（例如 Java stacktrace）。在这种情况下，以换行符 \n 为日志的结束标识符就显得有些不合理，为了能让日志系统明确区分开每条日志，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.
多行全文也会设置一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条多行日志原始数据为：

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
    at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
    at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

首行正则表达式为如下：

```
\d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2},\d{3}\s.+
```

采集到日志服务的数据为：

```
__CONTENT__:2019-12-15 17:13:06,043 [main] ERROR
com.test.logging.FooFactory:\njava.lang.NullPointerException\n    at
com.test.logging.FooFactory.createFoo(FooFactory.java:15)\n    at
com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

单行-完全正则格式

单行完全正则格式通常用来处理结构化的日志，指将一条完整日志按正则方式提取多个 key-value 的日志解析模式。
假设一条日志原始数据为：

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703
"http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum"
"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0"  0.354 0.354
```

配置的正则表达式为如下：

```
(\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"
([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
```

采集到日志服务的数据为：

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
```

版权所有：腾讯云计算（北京）有限责任公司

```
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

多行-完全正则格式

假设您的一条日志原始数据为：

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

行首正则表达式为：

```
\[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*
```

配置的自定义正则表达式为：

```
\[(\d+-\d+-\w+:\d+:\d+,\d+)\]\s\[(\w+)\]\s(.*)
```

系统根据 `()` 捕获组提取对应的 key-value 后，您可以自定义每组的 key 名称如下所示：

```
time:  2018-10-01T10:30:01,000`
level:  INFO`
msg: java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

JSON 格式

假设您的一条 JSON 日志原始数据为：

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34
+0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost":"unix:/tmp/php-
cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","request":"POST
/event/dispatch HTTP/1.1","xff":"-","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0
(Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
```

```
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

分隔符

假设您的一条日志原始数据为：

```
10.20.20.10 - ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/1.1 ::: 127.0.0.1
::: 200 ::: 647 ::: 35 ::: http://127.0.0.1/
```

当日志解析的分隔符指定为 `:::` ，该条日志会被分割成八个字段，并为这八个字段定义唯一的 key，如下所示：

```
IP: 10.20.20.10 -
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

组合解析

假设您的一条日志的原始数据为：

```
1571394459, http://127.0.0.1/my/course/4|10.135.46.111|200, status:DEAD,
```

自定义插件内容如下：

```
{
 "processors": [
   {
     "type": "processor_split_delimiter",
     "detail": {
       "Delimiter": ",",
       "ExtractKeys": [ "time", "msg1","msg2"]
     },
     "processors": [
       {
         "type": "processor_timeformat",
         "detail": {
           "KeepSource": true,
           "TimeFormat": "%s",
           "SourceKey": "time"
          }
       },
       {
         "type": "processor_split_delimiter",
         "detail": {
           "KeepSource": false,
           "Delimiter": "|",
           "SourceKey": "msg1",
```

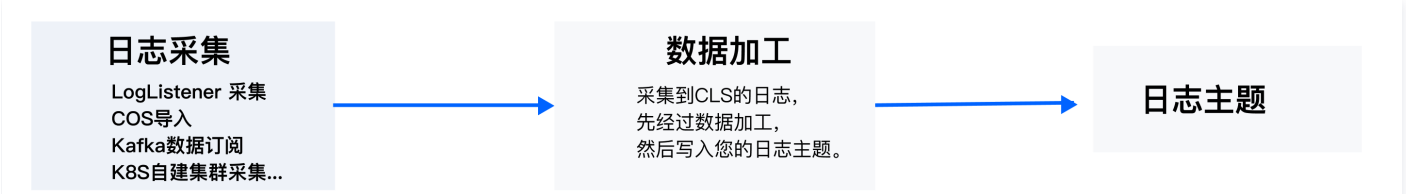
```
            "ExtractKeys": [ "submsg1","submsg2","submsg3"]
          },
          "processors": []
        },
        {
          "type": "processor_split_key_value",
          "detail": {
            "KeepSource": false,
            "Delimiter": ": ",
            "SourceKey": "msg2"
          }
        }
      ]
    }
  ]
}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
time: 1571394459
submsg1: http://127.0.0.1/my/course/4
submsg2: 10.135.46.111
submsg3: 200
status: DEAD
```

- 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。

如您需要对采集到的 CLS 日志进一步处理，如结构化、脱敏、过滤，然后再写入日志主题，您可单击页面下方的**数据加工**，添加数据加工，最后 配置索引 。



说明：

- 数据加工相关操作请参考 创建加工任务 文档中【前置数据加工】TAB 页面。
- 数据加工脚本的编写请参考 数据加工函数总览，或者 加工实战案例 。
- 数据加工会产生费用，详情请参见 计费概述 。

步骤4：配置索引

完成采集规则配置后，单击**下一步**，进入**索引配置**，有关索引请参考 索引 。

并在索引配置中配置如下信息：

- 索引状态：确认是否开启索引， 以使用日志检索等分析功能。
- 全文索引：确认是否需要设置大小写敏感。全文分词符：默认为"@&()='",;:<>[]{}/ \n\t\r"，确认是否需要修改。
- 是否包含中文：确认是否开启。

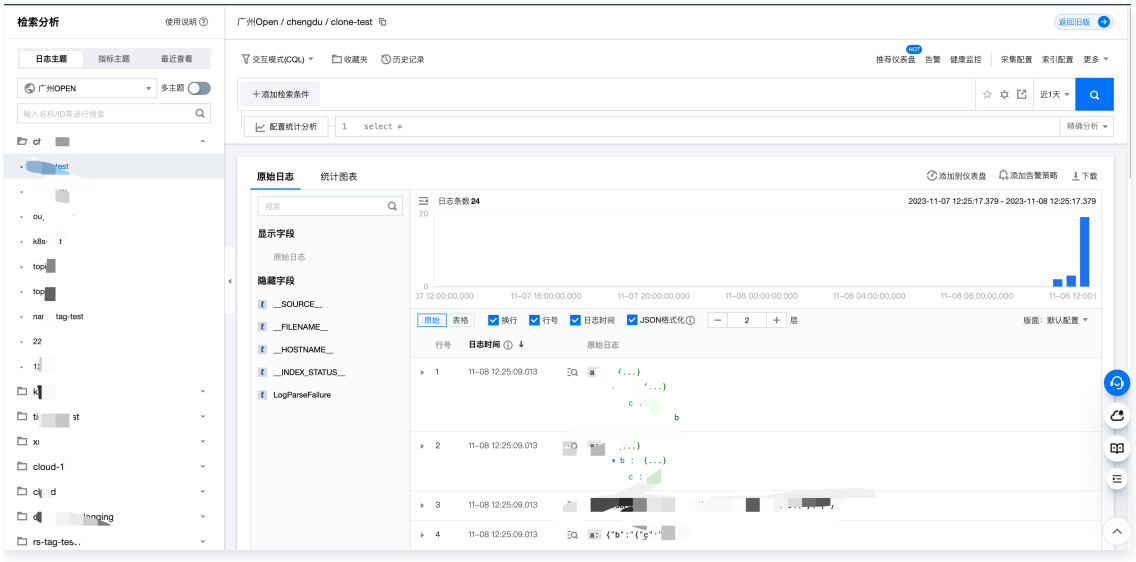
- 键值索引：您可根据 key 名按需进行字段类型、分词符以及是否开启统计分析的配置。若您需要开启键值索引，可打开开关。



> **注意：**
> - 检索必须开启索引配置，否则无法检索。
> - 若需要基于日志字段检索日志，需配置键值索引。
> - 若需要基于日志字段进行统计分析，需配置键值索引，并开启统计。
> - 索引规则编辑后仅对新写入的日志生效，已有数据不会更新。

步骤5：检索日志

至此，即完成了 TKE K8s 集群业务日志采集的所有部署。您可以前往 CLS 控制台 > 检索分析 查看采集上来的日志。



集群审计/事件日志采集

> **说明：**
> - 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。
> - 集群事件日志记录了包括了 集群的运行和各类资源的调度情况。

步骤1：选择集群

1. 登录 日志服务控制台 。

2. 在左侧导航栏中，单击**概览容器集群管理**，进入概览容器集群管理页面。
3. 在页面右上角选中 **TKE 集群**。



4. 选择 TKE 集群所在地域，并找到目标采集集群。
5. 若采集组件状态为**未安装**，单击**安装**，安装日志采集组件。



> **注意：**
> 安装日志采集组件将在集群 kube-system 命名空间下，以 DaemonSet 的方式部署一个 tke-log-agent 的 pod 和一个 cls-provisioner 的 pod。请为每个节点至少预留0.1核16Mib以上的可用资源。

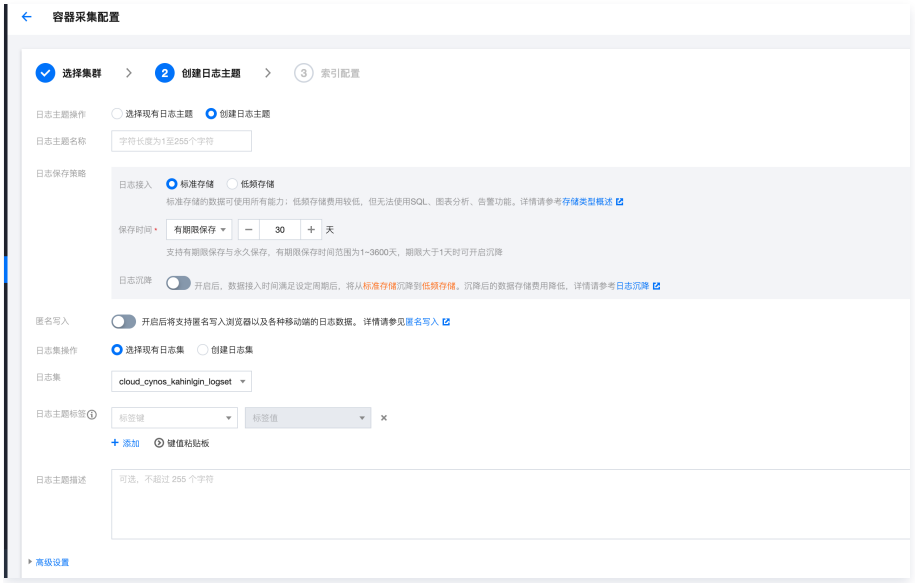
6. 若采集组件状态为**最新**，单击集群名称，进入集群详情页，并在集群详情页中找到**集群审计日志**或**集群事件日志**。



7. 单击**开启**集群审计或事件日志，并进入**集群审计**或**事件日志配置流程**。

步骤2：选择日志主题

进入审计或事件日志配置流程，在**创建日志主题**步骤中，您可选择已有或创建用于存储日志的日志主题。日志主题的相关信息请参考 日志主题 。



步骤3：索引配置

完成日志主题配置后， 单击**下一步进入索引配置**，索引的相关信息请参考 索引 。
索引配置中的配置信息如下：

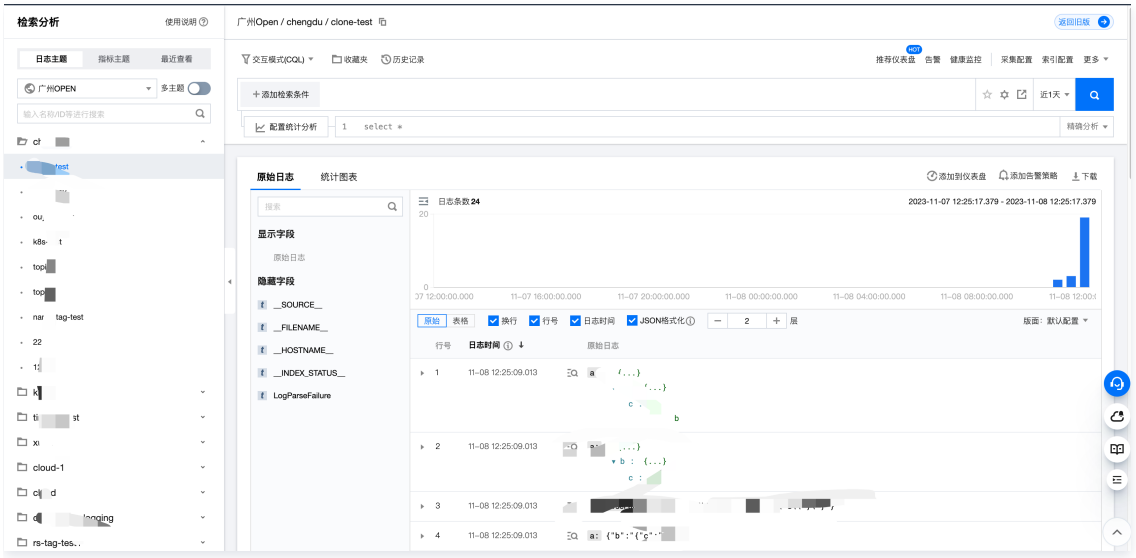
- 索引状态：确认是否开启索引， 以使用日志检索等分析功能。
- 全文索引：确认是否需要设置大小写敏感。全文分词符：默认为"@&()="',;:<>[]{}/ \n\t\r"，确认是否需要修改。
- 是否包含中文：确认是否开启。
- 键值索引：您可根据 key 名按需进行字段类型、分词符以及是否开启统计分析的配置。若您需要开启键值索引，可打开开关。



> **注意：**
>
> - 检索必须开启索引配置，否则无法检索。
> - 若需要基于日志字段检索日志，需配置键值索引。
> - 若需要基于日志字段进行统计分析，需配置键值索引，并开启统计。
> - 索引规则编辑后仅对新写入的日志生效，已有数据不会更新。

步骤4：检索日志

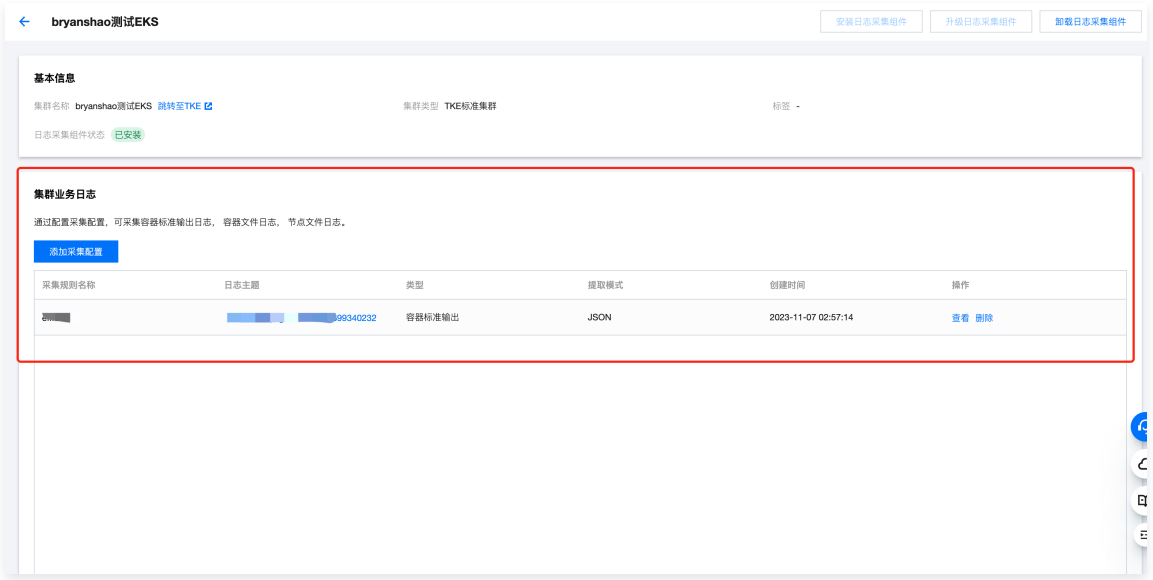
至此，即完成了 TKE K8s 集群审计或事件业务日志采集的所有部署。您可以前往 CLS 控制台 > 检索分析 查看采集上来的日志。



其他操作

管理业务日志采集配置

1. 在**容器集群管理**页中，找到目标 TKE 集群，并单击集群名称，进入集群详情页。
2. 在集群详情页中，您可以在**集群业务日志**中查看并管理您的集群业务日志采集配置。



升级日志采集组件

在**容器集群管理**页中，找到目标 TKE 集群，若采集组件状态显示**可升级**，可单击**升级**，将日志采集组件升级至最新版本。



卸载日志采集组件

在**容器集群管理**页中，找到目标 TKE 集群，单击操作中的**更多**，并在下拉框中单击**卸载日志采集组件**。



版权所有：腾讯云计算（北京）有限责任公司

采集 Syslog

最近更新时间：2025-07-03 17:35:32

使用场景

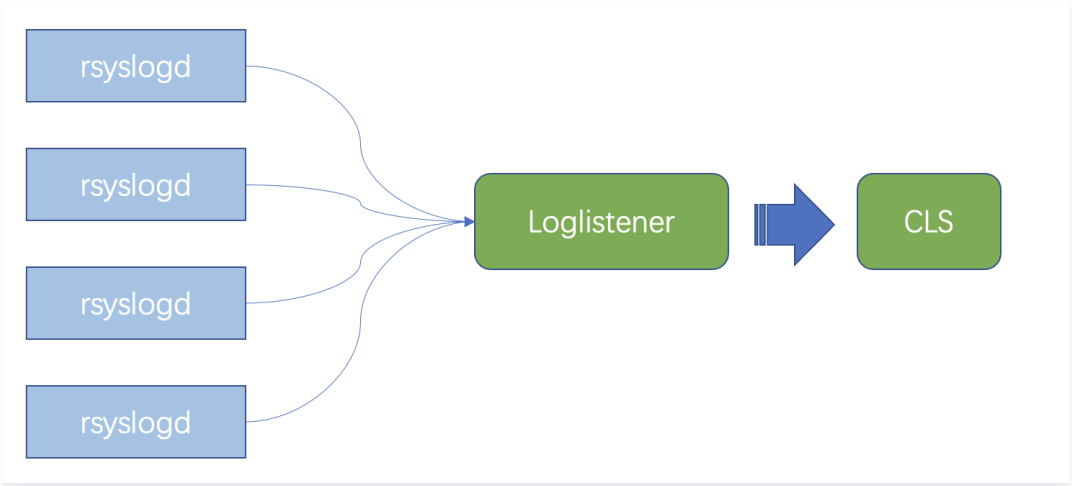
Syslog 常被称为系统日志或系统记录，是一种用来在因特网协定中传递记录档讯息的标准。网络中的路由器、交换机、防火墙、Unix/Linux 服务器等众多设备都支持它。Syslog 监控和管理对于每个组织来说都很重要，可以减少系统停机时间、提高网络性能并加强企业的安全策略。

前提条件

- 已部署好 rsyslog。
- 开通腾讯云日志服务。
- 在 rsyslog 目标 IP 的机器上已安装好 Loglistener 3.0.1.0 及以上版本。Loglistener 3.0.1.0及以上版本为白名单供应，需 提交申请 使用。

采集原理

采集原理如下图所示：



操作步骤

配置 rsyslog 转发

- 在 syslog 所在的服务器上修改 rsyslog 的配置文件/etc/rsyslog.conf，在配置文件的最后添加一行转发规则。添加转发规则后，rsyslog 会将 syslog 转发至指定 IP 地址和端口上。
- 如果通过当前服务器采集本机 syslog，配置转发地址为127.0.0.1，端口为任意非知名的空闲端口。
- 如果通过其他服务器采集本机 syslog，配置转发地址为其他服务器的公网 IP，端口为任意非知名的空闲端口。

例如以下配置表示将所有的日志都通过 TCP 转发至127.0.0.1:1000，配置文件详细说明请参见 RSyslog Documentation 。

```
*.* @@127.0.0.1:1000
```

执行以下命令重启 rsyslog，使日志转发规则生效。

```
sudo service rsyslog restart
```

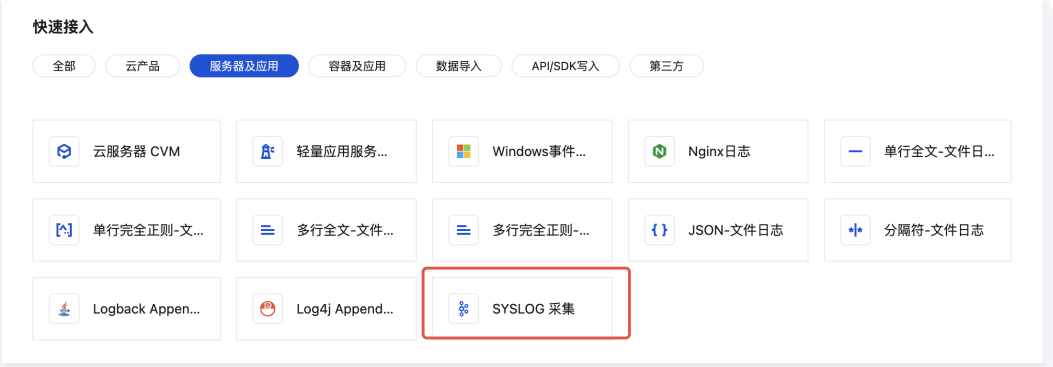
进入 CLS 控制台配置 syslog 采集规则

步骤1：选择日志主题

创建新的日志主题

1. 登录 日志服务控制台 。

2. 在左侧导航栏中，单击**概览**，进入概览页面。
3. 在**快速接入** > **服务器及应用**中，找到并单击 **SYSLOG 采集**。



4. 在创建日志主题页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步** 。

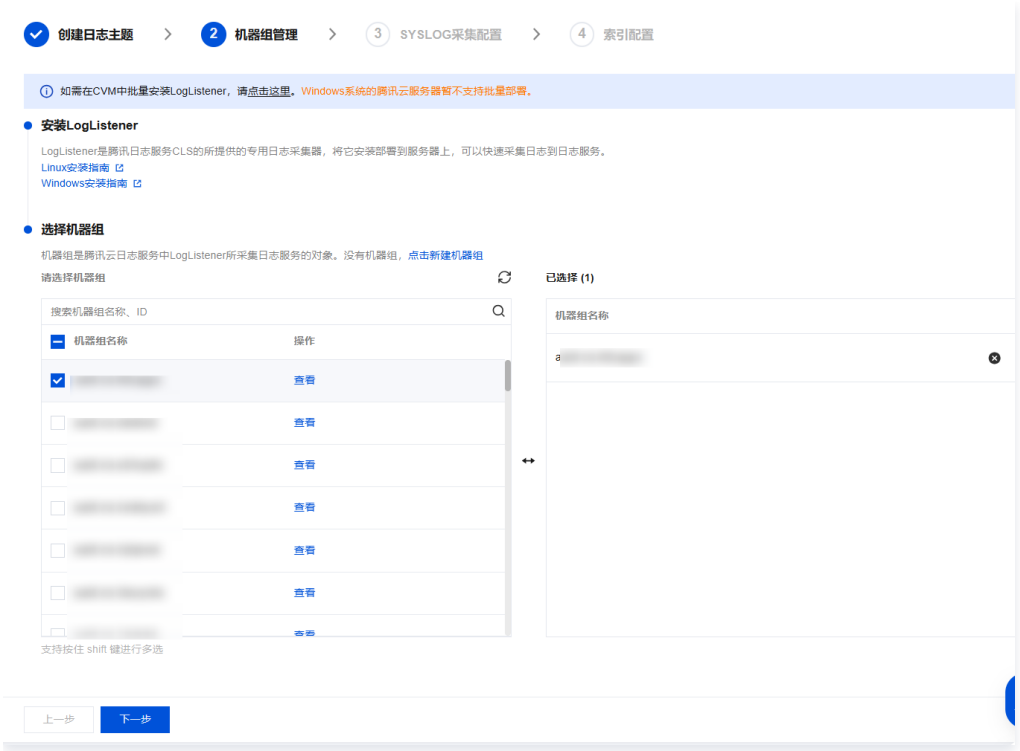
使用现有的日志主题

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，选择需要投递的日志主题，进入日志主题管理页面。
3. 选择**采集配置**页签，在 **Loglistener** 采集配置栏下单击**新增**，在列表中找到 **SYSLOG 采集**。



步骤2：机器组配置

在**机器组管理**页面，勾选需要与当前日志主题进行绑定的机器组，单击**下一步**，即可进入采集配置阶段，更多详情请参见 管理机器组 。



步骤3：syslog 采集配置

在 syslog 采集配置页面，依次配置如下信息：

配置项	类型	说明
采集规则名称	输入框	输入本条采集规则的名称
网络类型	单选框	指定 syslog 的传输协议，UDP/TCP
解析协议	单选框	指定解析日志所使用的协议，默认为空，表示不解析。其中： • rfc3164：指定使用 RFC3164协议解析日志 • rfc5424：指定使用 RFC5424协议解析日志 • auto：自动选择合适的解析协议
输出源	输入框	rsyslog 的配置文件中指定的 syslog 转发地址和端口，格式为[ip]:[port]。例如：127.0.0.1:9000
解析失败上传	开关	指定解析失败后的操作，打开时表示如果解析失败，按照输入的 key 返回日志全文。配置为 false ，表示解析失败时丢弃日志
解析失败日志的键名称(Key)	输入框	指定解析失败的 key 名



步骤4：索引配置

1. 在索引配置页面，配置如下信息。配置详情请参考 索引配置 。



> **注意：**
> - 检索必须开启索引配置，否则无法检索。
> - 索引规则编辑后仅对新写入的日志生效，已有数据不会更新。

2. 单击**提交**，完成导入配置。

查看 syslog 日志

当前日志主题下配置完成 syslog 采集后，单击**检索**，进入该主题**检索分析**页面查看 syslog。

日志字段解释

字段	说明

HOSTNAME	主机名，如果日志中未提供则获取当前主机名
program	协议中的 tag 字段
priority	协议中的 priority 字段
facility	协议中的 facility 字段
severity	协议中的 severity 字段
timestamp	日志对应的时间戳
content	日志内容，如果解析失败的话，此字段包含未解析日志的所有内容
SOURCE	当前主机的 IP 地址
client_ip	传输日志的客户端 IP 地址

版权所有：腾讯云计算（北京）有限责任公司

采集 Windows 事件日志

最近更新时间：2025-07-03 15:06:02

Windows 事件日志记录为应用程序和操作系统记录重要软件和硬件事件提供了一种标准的集中式方法。当软件、硬件或操作系统发生异常错误时，您可以通过 Windows 事件日志来检查错误发生的原因。本文将介绍如何通过 LogListener（Windows 版）采集 Windows 事件日志至 CLS 日志服务。

前提条件

目标 Windows 服务器已安装 LogListener。详情请参见 LogListener 安装指南（Windows 版）。

操作步骤

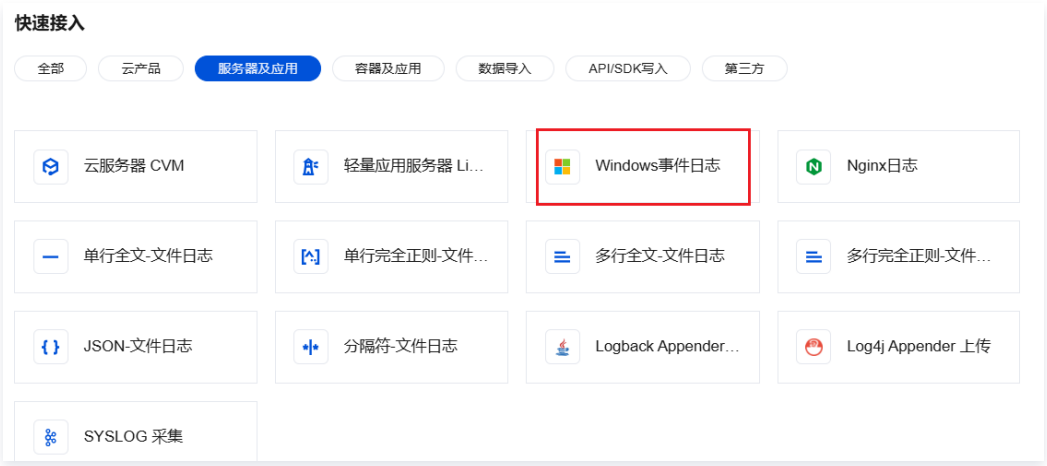
步骤1：选择日志主题

支持创建新的日志主题和使用现有的日志主题，您可按需选择。

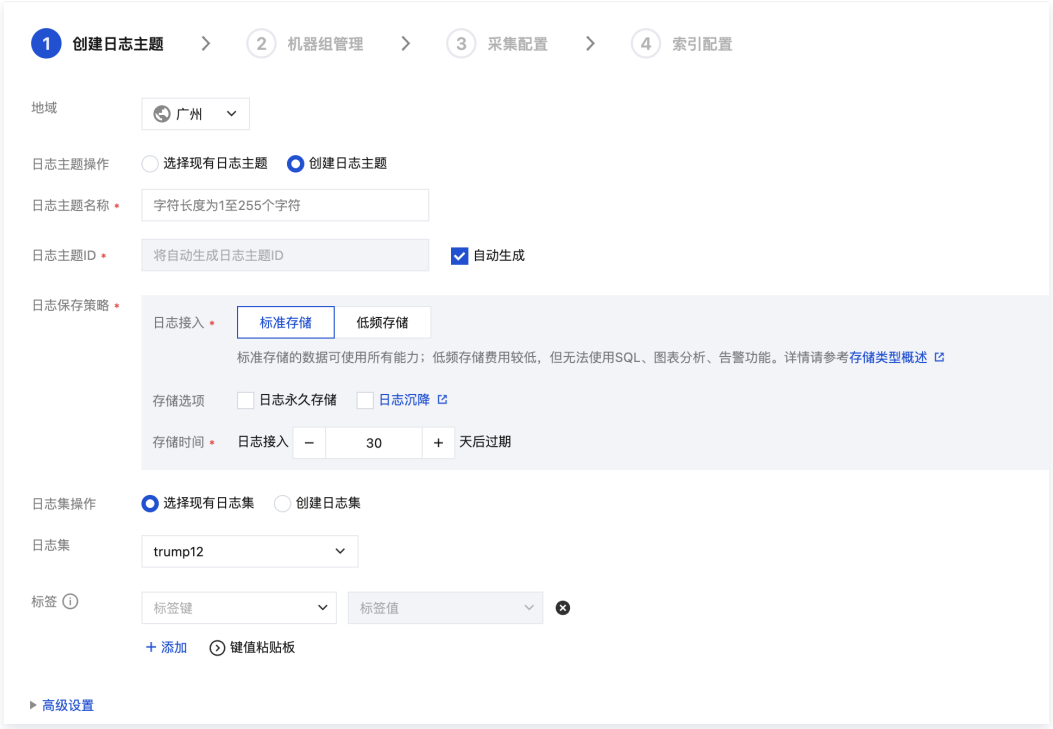
创建新的日志主题

如果您想选择新的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**概览**，进入概览页面。
3. 在**快速接入** > **服务器及应用**中，找到并单击 **Windows 事件日志**，进入采集配置流程。



4. 在**创建日志主题**页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步** 。



使用现有的日志主题

如果您想选择现有的日志主题，可执行如下操作：

1. **登录 日志服务控制台 。**
2. **在左侧导航栏中，选择日志主题，选择需要投递的日志主题，单击日志主题名称，进入日志主题管理页面。**
3. **选择采集配置页签，在 Windows 事件采集配置栏单击新增。**



步骤2：管理机器组

1. **完成日志主题新建或选择之后，进入机器组管理配置。**
2. **在机器组列表中选择需要采集的 Windows 机器组。**



如果您想选择新的机器组，可单击**新建机器组**，**系统环境**选择 **Windows**，并选择通过 IP 地址或者机器标识的方式关联目标 Windows 服务器。详情请参见 机器组 。完成创建后，在机器组列表中选中新建的机器组。



3. 选择完成后，单击**下一步**，进入**采集配置**流程。

步骤3：采集配置

在**采集配置**页面， 配置 Windows 事件采集规则。完成采集配置后，单击**下一步**。



配置事件采集规则

一个采集配置支持配置多个事件采集规则，一个事件采集规则包含以下配置项。

配置项	是否必填	说明
事件通道	是	目标采集的事件通道，支持选择： • Application（应用事件）：记录由应用程序生成的事件，例如软件崩溃、配置更改、错误消息等。 • System（系统事件）：记录操作系统组件的事件，例如驱动程序、系统服务、硬件问题等。 • Security（安全性事件）：记录与安全相关的事件，例如用户登录/注销、权限更改、审核策略更改等。 • Setup（配置事件）：记录系统设置和配置更改的事件。 • ALL（所有事件）。 **⚠ 注意：** 推荐一个服务器中的一种事件通道仅被一个采集配置采集。 若配置多个采集配置采集同一事件通道， 可能导致重复采集。
采集起始点	是	支持以下两种采集起始点： • 自定义时间：将从您指定的时间开始采集事件日志。 • 全量采集：将采集服务器上所有的事件日志。注意：若事件超出了 Windows 系统保留时长，其日志将不会采集。
自定义时间	是	当采集起始点选择为“自定义起始点”时， 需指定开始采集事件日志的时间。
事件 ID 过滤	否	支持正向过滤单个值（例：20）或范围（例：0-20），也支持反向过滤单个值（例：-20）。多个过滤项之间可由逗号隔开，例：1-200。-100表示采集1-200范围内除了100以外的事件日志。

步骤4：索引配置

1. **单击下一步，进入索引配置页面。**
2. **在索引配置页面，设置如下信息。配置详情请参考** 索引配置 **。**



注意：
检索必须开启索引配置，否则无法检索。

步骤5：检索分析

至此，您已完成 Windows 事件日志的采集配置。下一步，您可前往 检索分析页 查看日志。

日志字段解释

字段名	描述
computer_name	产生当前事件的节点名。
keywords	当前事件关联的关键字，用于事件分类。
level	当前事件的等级。
channel	当前事件的通道名。
event_data	和当前事件相关的数据。
message	当前事件关联的消息。
opcode	当前事件关联的操作码。
process.pid	当前事件的进程 ID。
type	获取当前事件使用的 API。
version	当前事件的版本号。
record_id	当前事件关联的记录编号。
event_id	当前事件的 ID。
task	当前事件关联的任务。
provider_guid	当前事件来源的全局事务 ID。

activity_id	当前事件所属活动的全局事务 ID，同一个活动的事件具有相同的全局事务 ID。
process.thread.id	当前事件的线程 ID。
provider_name	当前事件的来源。
raw_data	当前事件最原始的信息，XML 格式。

采集 Nginx 访问日志

最近更新时间：2024-09-10 09:41:21

操作场景

Nginx 作为常见的反向代理服务，在实际业务中承载着大量的服务请求。服务在运行过程中会产生大量访问日志，通常使用户面临着集群中日志分散，数据量庞大等问题，因此如何有效地收集管理这些日志数据对业务的运维及运营都有着非常重要的意义。本文以 Nginx 访问日志为例，介绍如何使用日志服务接入 Nginx 日志。

Nginx 日志格式

Nginx 访问日志（access.log）的格式可以通过 log_format 命令来定义，例如下列格式：

```
log_format main '$remote_addr - $remote_user [$time_local] '
 '"$request" $status $body_bytes_sent '
 '"$http_referer" "$http_user_agent"';
```

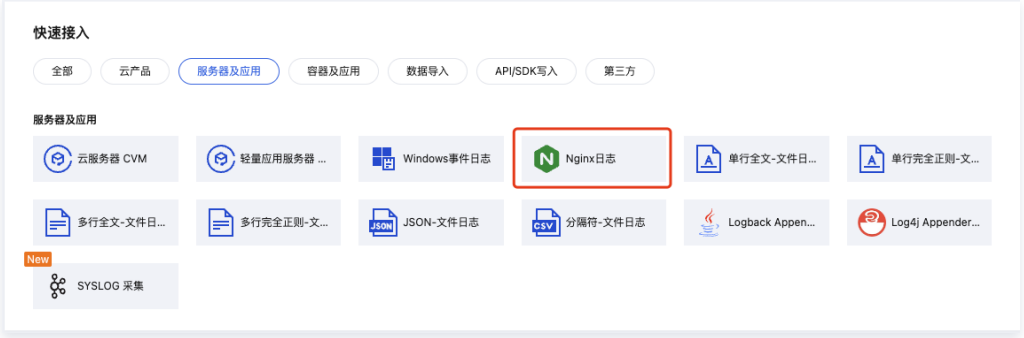
各字段含义如下：

字段名	含义
remote_addr	客户端 IP 地址
remote_user	客户端名称
time_local	服务器本地时间
request	HTTP 请求方法及 url 地址
status	HTTP 请求状态码
body_bytes_sent	发送给客户端的字节数
http_referer	访问来源的页面链接地址
http_user_agent	客户端浏览器信息

操作步骤

步骤一：创建/选择日志主题

- 如果您想选择新的日志主题，可执行如下操作：
  （1）登录 日志服务控制台 。
  （2）在左侧导航栏中，单击**概览**，进入概览页面。
  （3）在快速接入中，找到并单击 **Nginx 日志**，进入采集配置流程。



（4）在采集配置流程的**日志主题管理**页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步** 。



- **如果您想选择现有的日志主题，可执行如下操作：**

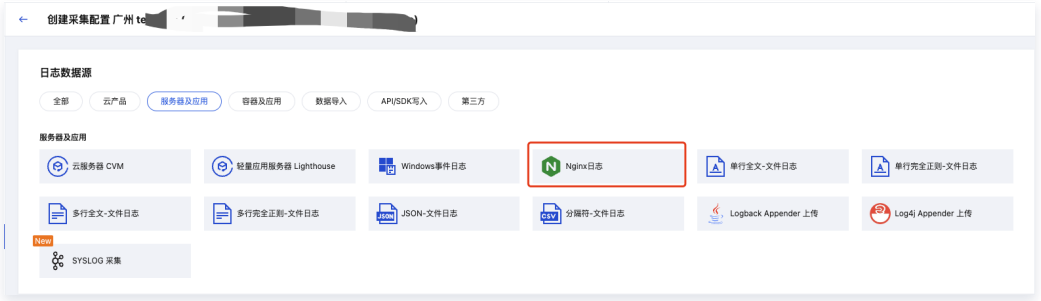
  （1）登录 日志服务控制台 。

  （2）在左侧导航栏中，单击日志主题，选择需要投递的日志主题，单击日志主题名称，进入日志主题管理页面。

  （3）选择采集配置页签，在 LogListener 采集配置栏下单击新增，进入采集配置流程。

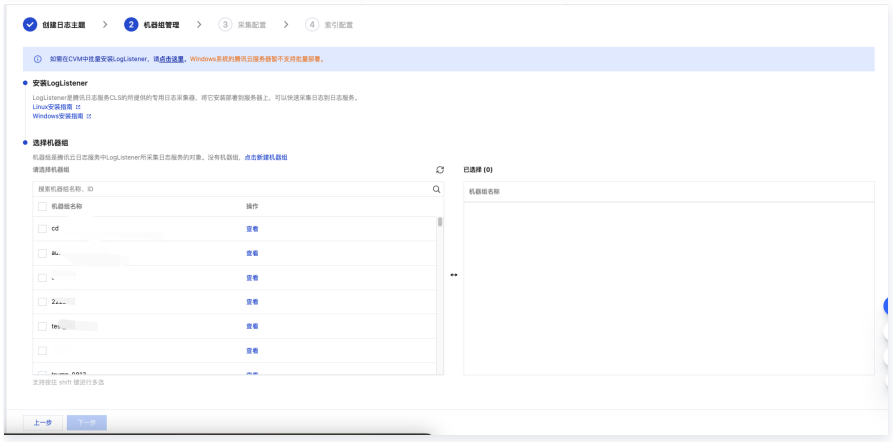


（4）在日志数据源选择页面中，找到并单击 Nginx 日志，进入采集配置流程。



步骤二：管理机器组

完成日志主题新建或选择之后，单击**下一步**进入**机器组管理**页面。



- 若您需要采集的目标服务器未安装 LogListener，可参见：
  - LogListener 安装指南（Linux 版）
  - LogListener 安装指南（Windows 版）
- 如果您想创建新的机器组，可执行如下操作：

  （1）单击**新建机器组**，并在机器组创建弹窗中，通过 IP 或者机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 机器组 。





（2）完成创建后，您可在列表中找到并选择您创建的机器组。



- 如果您想选择现有的机器组，可直接在列表中找到您的目标机器组：



步骤三：采集配置

完成机器组配置之后， 单击下一步进入采集配置流程 – 采集配置页面。

配置日志文件采集路径



在采集配置页面，填写采集规则名称，并根据日志采集路径格式填写采集路径。日志采集路径格式参考如下：

> **注意：**
> Linux日志路径必须以 `/` 开头。Windows文件路径必须以盘符开头，如 `C:\` 。

- Linux 系统中的日志路径： `/[目录前缀表达式]/**/[文件名表达式]` ，例如： `/data/log/**/*.log` 。
- Windows 系统中的日志路径 ： `[盘符]:\[目录前缀表达式]\**\[文件名表达式]` ，例如： `C:\Program Files\Tencent\...\*.log` 。

填写日志采集路径后，LogListener 会按照[目录前缀表达式]匹配所有符合规则的公共前缀路径，并监听这些目录（包含子层目录）下所有符合[文件名表达式]规则的日志文件。其参数详细说明如下：

字段	说明
目录前缀	日志文件前缀目录结构，仅支持通配符 * 和 ？ • * 表示匹配多个任意字符 • ？表示匹配单个任意字符 不支持填写逗号
**	表示当前目录以及所有子目录
文件名	日志文件名，仅支持通配符 * 和 ？ • * 表示匹配多个任意字符 • ？表示匹配单个任意字符 不支持填写逗号

常用的配置模式如下：

- **[公共目录前缀]/**/[公共文件名前缀]***
- **[公共目录前缀]/**/*[公共文件名后缀]**
- **[公共目录前缀]/**/[公共文件名前缀]*[公共文件名后缀]**
- **[公共目录前缀]/**/*[公共字符串]***

填写示例如下：

序号	目录前缀表达式	文件名表达式	说明
1.	/var/log/nginx	access.log	此例中，日志路径配置为 `/var/log/nginx/**/access.log` ，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `access.log` 命名的日志文件
2.	/var/log/nginx	*.log	此例中，日志路径配置为 `/var/log/nginx/**/*.log` ，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `.log` 结尾的日志文件
3.	/var/log/nginx	error*	此例中，日志路径配置为 `/var/log/nginx/**/error*` ，LogListener 将会监听 `/var/log/nginx` 前缀路径下所有子目录中以 `error` 开头命名的日志文件

> **注意**
> - Windows 系统环境暂不支持软链采集。
> - Loglistener 2.3.9及以上版本才可以添加多个采集路径。
> - 日志轮转场景下，建议配置采集路径为 `log/*.log`，并将日志轮转后的老文件命名为 `log/*.log.xxxx`。
> - 默认情况下，一个日志文件只能被一个日志主题采集。如果一个文件需要对应多个采集配置，且文件所在系统环境为 Linux，请给源文件添加一个软链接，并将其加到另一组采集配置中。

配置采集路径黑名单

开启采集路径黑名单，可在采集时忽略指定的目录前缀或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



采集黑名单分为两类过滤类型，且可以同时使用：

- 文件路径：采集路径下，需要忽略采集的完整文件路径，支持通配*或?，支持**路径模糊匹配。
- 目录路径：采集路径下，需要忽略采集的目录前缀，支持通配*或?，支持**路径模糊匹配。

> **注意：**
> - 需要 LogListener 2.3.9 及以上版本。
> - 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

配置采集策略

- 全量采集：Loglistener 采集文件时，从文件的开头开始读。
- 增量采集：Loglistener 采集文件时，只采集文件内新增的内容。

配置回溯采集

> **注意：**
> Windows 系统环境暂不支持。

当采集策略选择为**增量采集**时，您可进一步在回溯采集中指定当 LogListener 启动采集时，是否从最新的位置往前偏移指定字节的位置开始采集。

配置编码模式

- UTF-8：若您的日志文件编码模式为 UTF-8，请选择该选项。
- GBK：若您的日志文件编码模式为 GBK，请选择该选项。

配置提取模式

- 提取模式选择“Nginx 日志模板”，填写 Nginx 配置文件（通常为 /etc/nginx/nginx.conf 或 /usr/local/nginx/conf/nginx.conf）中的日志模板，通常以 log_format 开头，例如：

```
log_format main '$remote_addr - $remote_user [$time_local] '
 '"$request" $status $body_bytes_sent '
 '"$http_referer" "$http_user_agent"';
```

- CLS 将根据上面的 Nginx 日志模板自动生成日志提取正则表达式，例如：

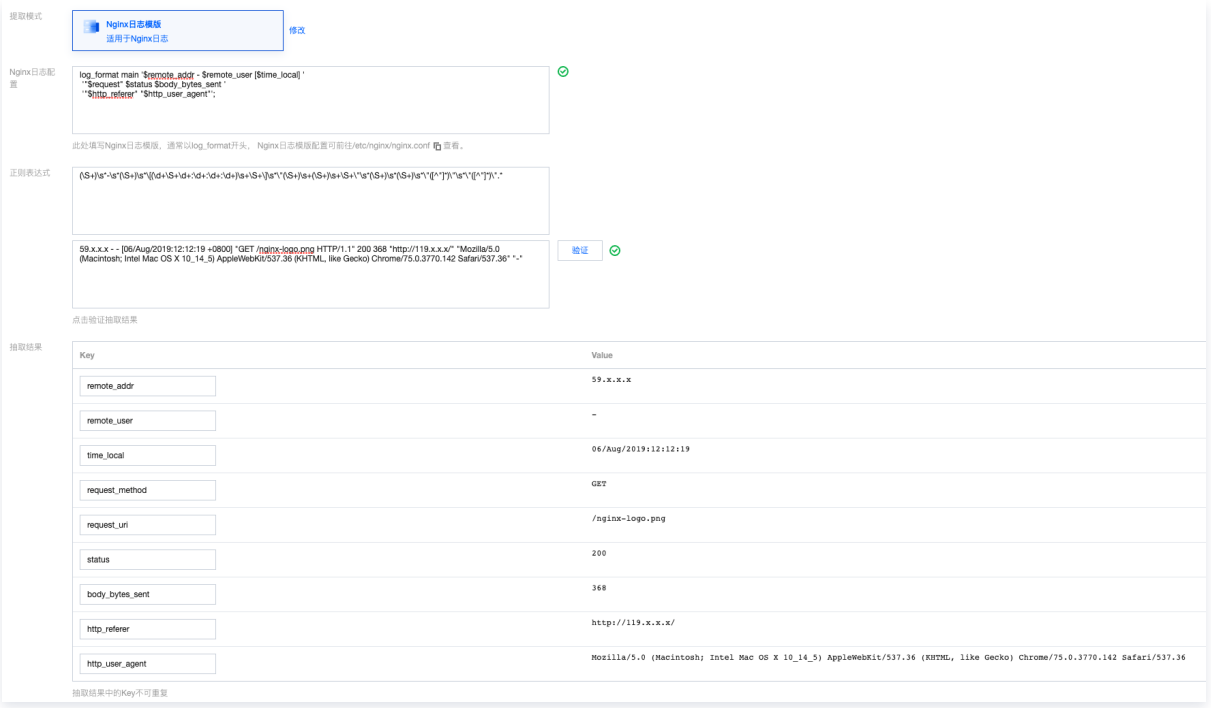
```
(\S+)\s*-\s*(\S+)\s*\[(\d+\S+\d+:\d+:\d+:\d+)\s+\S+\]\s*\"(\S+)\s+(\S+)\s+\S+\"\s*(\S+)\s*(\S+)\s*\"
```

```
([^"]*)\"\s*\"([^"]*)\".*
```

- 在样例日志中填写一条实际的 Nginx 日志，用来验证上述提取模式配置是否正确，例如：

```
59.x.x.x - - [06/Aug/2019:12:12:19 +0800] "GET /nginx-logo.png HTTP/1.1" 200 368 "http://119.x.x.x/"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/75.0.3770.142 Safari/537.36" "-"
```

- 提取模式验证成功后，可为每个字段设定名称或直接使用默认值：



配置自定义元数据

> **注意：**
> - Windows 系统环境暂不支持。
> - 需要 LogListener-2.8.7 及以上版本。

您可配置自定义元数据，可以用来区分日志。当前支持以下几种元数据配置。详情请参见 自定义元数据 。

- 机器组元数据：使用机器组元数据。
- 采集路径：通过正则提取采集路径中的值作为元数据。
- 自定义：自定义键值作为元数据。

配置过滤器条件

过滤器的目的是根据业务需要添加日志采集过滤规则，以帮助您筛选出有价值的日志数据。
单行全文模式下，默认使用 `__CONTENT__` 作为全文的键（key）名， 并支持以下过滤规则：

- 等于：仅采集指定字段值匹配指定字符的日志。 支持完全匹配，或正则匹配。
- 不等于：仅采集指定字段值不匹配指定字符的日志。支持完全匹配，或正则匹配。

例如，单行全文日志样例格式为 `Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64` ，您希望采集1月22号这一天的所有日志，则 key 处填写 `__CONTENT__` ，过滤规则配置选择**等于**，value 处填写 `Tue Jan 22.*` 。

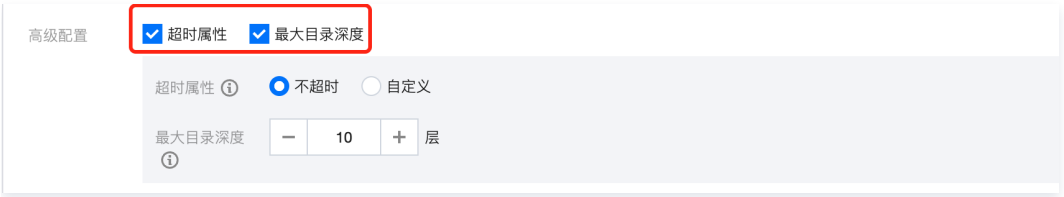
> **注意**
> - Windows 系统环境暂不支持。
> - 过滤规则 “不等于” 、“字段存在”以及“字段不存在” 仅在 LogListener 2.9.3及以上版本支持。

- 多条过滤条件之间关系是"与"逻辑，若同一 key 名配置多条过滤条件，规则会被覆盖。

高级配置

> **注意：**
> Windows 系统环境暂不支持。

通过勾选，选择您需要定义的高级配置：

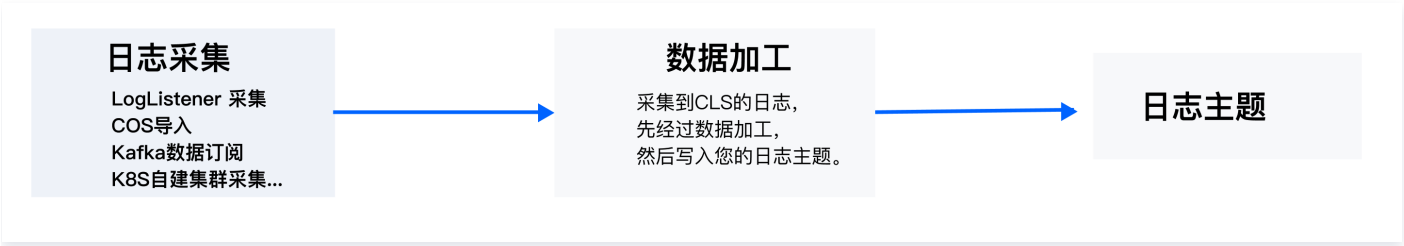


单行全文模式下，支持配置以下高级配置：

名称	描述	配置项
超时属性	该配置控制日志文件的超时时间。如果一个日志文件在指定时间内没有任何更新，则为超时。超时的日志文件 LogListener 将不再采集。当您的日志文件数量较大时，建议降低超时时间，避免 LogListener 性能浪费	• 不超时：日志文件用不超时 • 自定义：自定义日志文件的超时时间
最大目录深度	采集路径中的/**/代表查找所有子目录中的文件。但是，在查找过程中，如果不想搜索太深的目录，可以使用"最大目录深度"配置项来限制搜索深度	大于0的整数。0代表不进行子目录的下钻

数据加工

如您需要对采集到的 CLS 日志进一步处理，如结构化、脱敏、过滤，然后再写入日志主题，您可点击**采集配置**页面下方的**数据加工**，添加数据加工，最后 配置索引 。



> **说明：**
> - 数据加工相关操作请参考 创建加工任务 文档中【采集后立即加工】TAB 页面；
> - 数据加工脚本的编写请参考 数据加工函数总览，或者 加工实战案例；
> - 数据加工会产生费用，详情请参见 计费概述 。

步骤四：索引配置

1. 单击**下一步**，进入索引配置页面。
2. 在**索引配置**页面，设置如下信息。配置详情请参考 索引配置 。

> **注意**
> 检索必须开启索引配置，否则无法检索。

3. 单击**提交**，完成采集配置。

步骤五：检索分析 Nginx 日志

1. 在左侧导航栏中，单击**检索分析**，进入检索分析页面。
2. 单击顶部的“**日志集**”和“**日志主题**”的下拉框，选择待检索的日志主题，即可查询到 Nginx 访问日志内容。

版权所有：腾讯云计算（北京）有限责任公司

3. 更多检索分析示例语句可参见 Nginx 访问日志分析 。

订阅 MySQL Binlog

最近更新时间：2025-05-29 19:08:21

CLS 日志服务支持远端非侵入式订阅 MySQL 二进制日志（binlog），该日志以**事件形式**记录了对 MySQL 数据库执行更改的所有操作。适用于对增量的数据库改动进行实时查询、分析与审计的需求场景。本文将为您介绍如何订阅 MySQL Binlog。

前提条件

- 已创建 MySQL 数据库且开启 ROW 模式的 Binlog 功能。
- 创建一个具有 `REPLICATION SLAVE` 、`REPLICATION CLIENT` 和 `SELECT` 权限的 MySQL 账号。

操作步骤

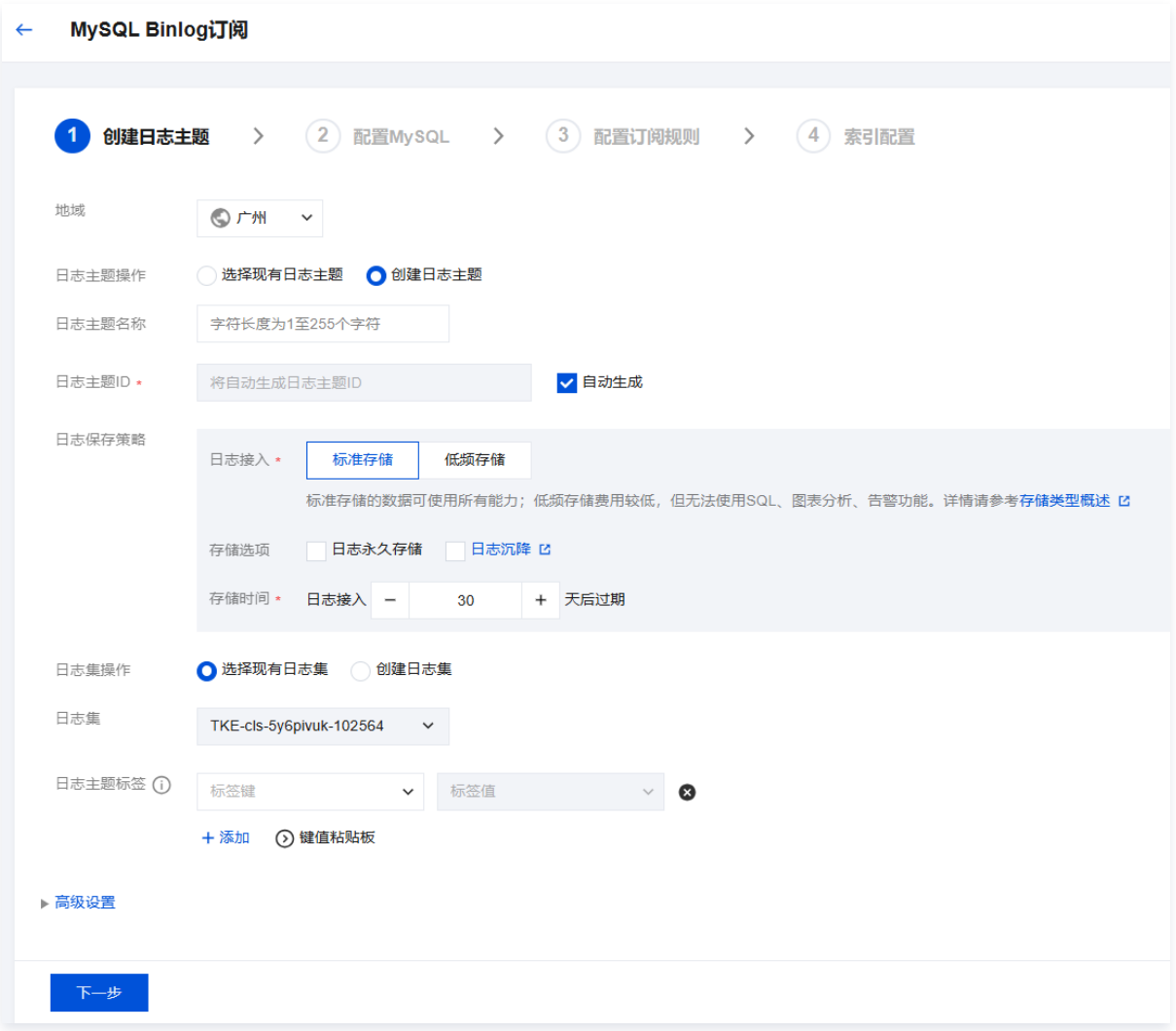
步骤1：选择日志主题

如果您想选择新的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**概览**，进入概览页面。
3. 在**快速接入**中，选择**数据导入**，单击 **MySQL Binlog 订阅**，进入订阅任务配置流程。

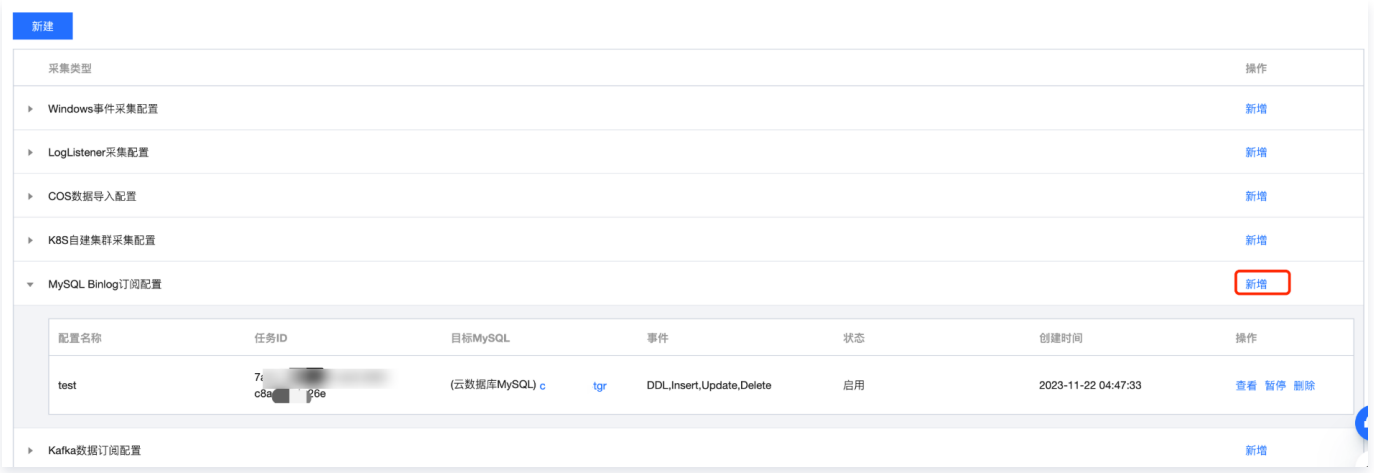


4. 在**创建日志主题**页面，根据您的实际需求，输入日志主题名称，配置日志保存时间等信息，然后点击**下一步** 。



如果您想选择现有的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，选择需要投递的日志主题，单击日志主题名称，进入日志主题详情页面。
3. 选择**采集配置**页签，在 MySQL Binlog 订阅配置栏下单击**新增**，进入订阅任务配置流程。



步骤2：配置 MySQL

在配置 MySQL 步骤中，配置如下参数：



参数	是否必填	说明
MySQL 类型	是	目标订阅的 MySQL 类型，当前支持： • 腾讯云数据库 MySQL • 腾讯云 TDSQL-C MySQL • 自建 MySQL
MySQL 实例	是	若 MySQL 类型为**腾讯云数据库 MySQL** 或**腾讯云 TDSQL-C MySQL**，可通过下拉选项选择需要订阅的目标云上 MySQL 实例。 ⚠ **注意：** 当前仅支持选择与日志主题相同地域的云上 MySQL 实例，若需订阅不同地域 MySQL 实例的 Binlog，**MySQL 类型**请选择**自建 MySQL**，并通过公网订阅 Binlog。
访问方式	是	若 MySQL 类型为**自建 MySQL**，支持选择通过内网地址或公网地址访问您的 MySQL。
所属网络	是	若 MySQL 类型为**自建 MySQL**，且访问方式为通过内网地址访问，您需指定 MySQL 实例所在的 VPC 网络。 ⚠ **注意：** 仅支持与目标日志主题所在地域相同的 VPC 网络。
网络服务类型	是	若 MySQL 类型为**自建 MySQL**，且访问方式为通过内网地址访问，需指定目标 MySQL 的网络服务类型： • 若您的 MySQL 需通过 CLB 访问，请选择 CLB。 • 若您的 MySQL 所在服务器可直接访问，请选择 CVM。
内网访问地址	是	若 MySQL 类型为自建 MySQL，基于您选择访问方式，填写 MySQL 的内网访问地址或公网访问地址。
MySQL 端口	是	若 MySQL 类型为自建 MySQL，填写 MySQL 的端口。
用户名	是	填写访问 MySQL 的用户名。 ⚠ **注意：** 账号需开启以下权限：`REPLICATION SLAVE`、`REPLICATION CLIENT` 和 `SELECT`。
密码	是	填写访问 MySQL 的密码。

步骤3：配置订阅规则

在配置订阅规则步骤中，配置如下参数：



参数	是否必填	说明
订阅规则名称	是	指定当前订阅规则的名称。
起始位置	是	指定 Binlog 订阅的起始位置。支持通过以下3种方式定义起始位置： • 最新位置：从最新位置开始采集 Binlog。 • 指定位置：从指定的位置开始采集 Binlog。 • 指定 GTID：从指定的 GTID（事务 ID）位置开始采集 Binlog。
Binlog 文件名	是	若起始位置选择为**指定位置**，需指定 Binlog 的文件名，例如：mysql-bin.000005。
开始的 Binlog 位置	是	若起始位置选择为**指定位置**，需指定在 Binlog 文件中开始采集的位置。例如：需要采集 mysql-bin.000005 中的第 N 条日志的起始下标，则指定位置为第 N-1条日志的 position+size（position 是 Binlog 日志在 Binlog 文件中的起始下标，size 是该条 Binlog 日志大小）。若需要从文件的开头进行采集，指定位置为0。 ⚠ **注意：** 请确保填写位置正好为一条 Binlog 日志的起始下标，否则可能导致订阅失败。
开始的 GTID	是	若起始位置选择为**指定 GTID**，需指定起始的 GTID（事务 ID）。
事件类型	是	需要采集的 Binlog 事件类型，支持以下4种事件类型： • DDL • Insert • Update • Delete
事件元数据	否	支持勾选以下6种 Binlog 相关元数据与日志一起上传： • log_name：Binlog 文件名 • position：Binlog 日志在 Binlog 文件中的起始下标 • size：Binlog 日志大小 • server_id：Slave ID • gtid：事务 ID • task_id：当前订阅任务的 ID ⚠ **注意：** DDL 事件的 Binlog 不支持 size 元数据。
时间戳来源	是	支持选择以下2种时间作为所采集 Binlog 日志的时间戳： • 采集时间：Binlog 日志被采集的时间。 • 事件事件：Binlog 日志对应的事件发生时间。

数据库表白名单	否	开启后可采集指定数据库以及数据库表，关闭则代表采集所有的数据库以及数据库表。支持配置多个白名单，每个白名单需配置以下信息： • 数据库名称：需要采集的 Binlog 所关联的数据库名称。 • 数据库表：需要采集的 Binlog 所关联的数据库表名，支持填写多个。 **注意：** DDL 事件的 Binlog 日志采集不受白名单限制。
数据库黑名单	否	开启后可忽略不采集指定数据库以及数据库表。支持配置多个黑名单，每个黑名单需配置以下信息： • 数据库名称：需要忽略不采集的 Binlog 所关联的数据库名称。 • 数据库表：需要忽略不采集的 Binlog 所关联的数据库表名，支持填写多个。 **注意：** DDL 事件的 Binlog 日志采集不受名单限制。
平铺时间数据	否	• 若开启，会将不同的事件拆分为多条日志，并在每条日志中对被操作的字段进行平铺。例如：在数据库操作中执行了两次操作，一次更新 a 和 b 字段，一次更新 d 和 e 字段，则会生成两条日志，分别的值为 a:xxx, old_a:xxx, b:xxx, old_b:xxx 和 d:xxx, old_d:xxx, e:xxx, old_e:xxx。 • 若关闭，会将事件数据以 数组+JSON 格式集中打包到 old_data 和 data 两个字段中。例如: 在数据库操作中执行了两次操作，一次更新 a 和 b 字段，一次更新 d 和 e 字段，日志值为 old_data:[{a:xxx,b:xxx}, {d:xxx,e:xxx}], data:[{a:xxx,b:xxx}, {d:xxx,e:xxx}]。
预览 Binlog	否	单击后将从数据库中获取第一条符合订阅类型的 Binlog 日志，展示相应的内容。

步骤4：索引配置

在索引配置页面，设置如下信息。



- **索引状态：确认是否开启。**

  注意：
  若要对日志进行检索，索引状态需未开启，否则无法检索日志。

- **全文索引：全文索引将日志全文整体切分为多个分词进行索引构建，检索时直接通过关键词进行检索（即全文检索），例如使用 `error` 检索所有出现过 error 的日志。**

配置项	功能描述

全文分词符	对原始日志进行分词的字符集合，仅支持英文符号，控制台默认分词符为 `@&?|#()='",;:<>[]{}/ \n\t\r\\` 。
大小写敏感	检索时是否对大小写敏感。例如日志为 Error，若大小写敏感，则使用 error 无法检索到该条日志。
包含中文	日志中包含中文且需要对中文进行检索时可开启该功能。例如日志为 `"用户登录接口超时"` ，若未开启该功能，搜索 `"超时"` 无法检索到该日志，只有完整地搜索 `"用户登录接口超时"` 才能检索到该日志，开启该功能后便可通过搜索 `"超时"` 检索到该日志。

- 键值索引：键值索引将原始日志按字段（即 key:value）分别切分为多个分词进行索引构建，检索时可基于键值方式进行检索（即键值检索），例如使用 `type:ddl AND db:test` 检索事件为 DDL 且事件关联的数据库名为 test 的日志。

配置项	功能描述
字段名称	字段名称，单个日志主题键值索引最多可添加300个字段。仅支持字母、数字、下划线和-./@，且不能以下划线开头。 Binlog 日志可配置键值索引的字段请参考 Binlog 日志字段说明 。
字段类型	字段的数据类型，包括 text、long、double 三种类型。 • text 类型支持使用通配符进行模糊检索，不支持范围比较。 • long、double 类型支持范围比较，不支持模糊检索。
分词符	对字段值进行分词的字符集合，仅支持英文符号，控制台默认分词符为 `@&?|#()='",;:<>[]{}/ \n\t\r\\` 。
包含中文	字段中包含中文且需要对中文进行检索时可开启该功能。例如日志为 `"message:用户登录接口超时"` ，若未开启该功能，使用 `message:"超时"` 无法检索到该日志，只有使用 `message:"用户登录接口超时"` 才能检索到该日志，开启该功能后便可使用 `message:"超时"` 检索该日志。
开启统计	开启后可对该字段使用 SQL 进行统计分析。text 类型字段开启统计时，如果值过长，仅前32766个字符参与统计运算。 开启统计不会额外产生费用，建议您开启。
大小写敏感	检索时是否对大小写敏感，例如日志为 `"level:Error"` ，若大小写敏感，则使用 `level:error` 无法检索到该条日志。

- 索引相关的更多详情，请参见 索引配置 。

步骤5：检索分析

至此，您已完成 MySQL Binlog 订阅的配置，下一步即可开始使用日志的检索与分析，详情请参见 检索分析 。

日志字段说明

固定字段

字段	描述
db	数据库名称
table	表名称
query	query
type	事件类型
after	当 type 为 insert、update、delete 时，通过 json 包裹所有受到影响的字段以及结果值
before	当 type 为 update 时，通过 json 包裹所有受影响的字段以及影响产生前的值
errorCode	当 type 为 DDL 时，展示执行 errorcode
executionTime	当 type 为 DDL 时，展示执行耗时

元数据字段

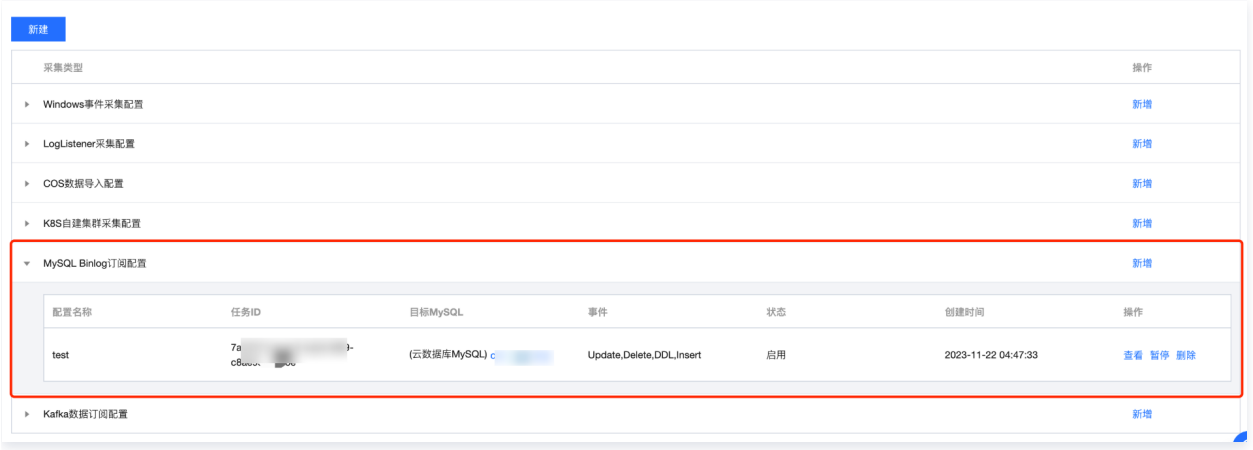
> **说明：**
> 需在 Binlog 订阅规则中指定后才会上传。

字段	描述
log_name	Binlog 文件名
position	Binlog 日志在 Binlog 文件中的起始下标
size	Binlog 日志大小
server_id	Slave ID
gtid	事务 ID
task_id	订阅规则任务 ID

其他操作

管理订阅任务

完成 MySQL Binlog 订阅任务配置后，您可在日志主题详情页的**采集配置**中查看并管理所有已创建的 MySQL Binlog 订阅任务。



使用 Kafka 协议上传日志

最近更新时间：2025-06-27 11:06:31

日志服务（Cloud Log Service，CLS）目前已支持使用 Kafka Producer SDK 和其他 Kafka 相关 agent 上传日志到 CLS。

使用场景

日志应用中使用 Kafka 作为消息管道是非常普遍的场景。如通过机器上的开源采集客户端或者使用 producer 直接写入日志，再通过消费管道提供给下游如 spark、flink 等进行消费。CLS 具备完整的 Kafka 数据管道上下行能力，以下主要介绍哪些场景适合您使用 Kafka 协议上传日志，更多 Kafka 协议消费场景请参见 Kafka 协议实时消费 。

- **场景1**：您已有基于开源采集的自建系统，不希望有复杂的二次改造，您可以通过修改配置文件将日志上传到 CLS。
  例如，您之前使用 ELK 搭建日志系统的客户，现在只需要通过修改 Filebeat 或者 Logstash 的配置文件，将 Output 配置（详情请参见 filebeat 配置 ）到 CLS，即可非常方便简洁地将日志上传。
- **场景2**：您希望通过 Kafka producer SDK 来采集日志并上传，不必再安装采集 Agent。
  CLS 支持您使用各类 Kafka producer SDK 采集日志，并通过 Kafka 协议上传到 CLS。详情请参见本文提供的 SDK 调用示例 。

相关限制

- 支持 Kafka 协议版本为：0.11.0.X、1.0.X、1.1.X、2.0.X、2.1.X、2.2.X、2.3.X、2.4.X、2.5.X、2.6.X、2.7.X、2.8.X、3.X.X。
- 支持压缩方式：gzip、snappy、lz4。
- 当前使用 SASL_PLAINTEXT 认证。
- 使用 Kafka 协议上传需要配置 RealtimeProducer 权限，详情请参见 CLS 访问策略模板 。

配置方式

使用 Kafka 协议上传日志至 CLS Kafka 生产者端时，需要配置下 CLS Kafka 访问信息。

参数	说明
鉴权机制	当前支持 SASL_PLAINTEXT。
hosts	CLS Kafka 地址，根据目标写入日志主题所在地域配置。请参见 CLS Kafka 地址 。
topic	CLS Kafka topic 名称，配置为日志主题 ID。例如：76c63473-c496-466b-XXXX-XXXXXXXXXXXX。
username	CLS Kafka 访问用户名，配置为日志集 ID。例如：0f8e4b82-8adb-47b1-XXXX-XXXXXXXXXXXX。
password	• CLS Kafka 访问密码，格式为 `${secret_id}#${secret_key}` 。例如：XXXXXXXXXXXXXXX#YYYYYYYY。密钥信息获取请前往 密钥获取 。请确保密钥关联的账号具有相应的 Kafka 协议上传日志权限 。 • 若要匿名写入，格式为 `topic_id#${日志主题 ID}` 。例如：topic_id#76c63473-c496-466b-XXXX-XXXX。 **注意：** 若使用匿名写入，目标日志主题需开启**匿名访问**，并在**匿名操作**选择 **Kafka 协议上传日志**。详情请参见 管理日志主题 。 
header	定义在通过 Kafka 协议上传时日志的解析行为。 • json_remove_escape：对日志进行 JSON 解析时是否进行去转义，取值为 true、false，若不指定默认为 false。 • time_key：日志中的指定时间字段，表示选择日志中的指定字段作为日志采集时间。

- time_format：当配置了 time_key，需额外配置指定字段的时间解析格式，详情请参见 配置时间格式 。

CLS Kafka 地址

通过 Kafka 协议上传日志时，基于访问方式（内网/外网）以及目标日志主题所在地域，CLS Kafka 地址配置为如下。

说明：

- 地域（region）格式请参见 可用域名 - Kafka上传日志 。
- 访问方式：
  - 内网访问：数据发起端为腾讯云服务器且服务器所在地域与目标日志主题一致。
  - 外网访问：数据发起端为非腾讯云服务器或者服务器所在地域与目标日志主题不一致。

访问方式	CLS Kafka 地址
内网	${region}-producer.cls.tencentyun.com:9095
外网	${region}-producer.cls.tencentcs.com:9096

示例

Beat 调用示例

filebeat/winlogbeat 配置

注意：
不同的 filebeat 版本，请使用不同的配置内容。

filebeat版本 > 8.18

```
output.kafka:
  enabled: true
  hosts: ["${region}-producer.cls.tencentyun.com:9095"] # TODO 服务地址；外网端口9096，内网端口9095
  topic: "${ClsTopicID}" # TODO 日志主题 ID
  version: "1.0.0"
  compression: "${compress}"    # TODO 配置压缩方式，支持 gzip、snappy、lz4，例如"lz4"
  username: "${ClslogsetID}" # TODO 日志集 ID
  # 若要匿名写入，password: "topic_id#${日志主题 ID}"
  password: "${secret_id}#${secret_key}"
```

filebeat版本 ⩽ 8.18

```
output.kafka:
  enabled: true
  hosts: ["${region}-producer.cls.tencentyun.com:9095"] # TODO 服务地址；外网端口9096，内网端口9095
  topic: "${ClsTopicID}" # TODO 日志主题 ID
  version: "0.11.0.2"
  compression: "${compress}"    # TODO 配置压缩方式，支持gzip、snappy、lz4，例如"lz4"
  username: "${ClslogsetID}" # TODO 日志集 ID
  # 若要匿名写入，password: "topic_id#${日志主题 ID}"
  password: "${secret_id}#${secret_key}"
```

若出现报错 kafka: client has run out of available brokers to talk to。建议将 version 的版本提升至1.0.0。

Logstash 调用示例

logstash 配置

```
output {
  kafka {
    topic_id => "${ClstopicID}"
    bootstrap_servers => "${region}-producer.cls.tencentyun.com:${port}"
    sasl_mechanism => "PLAIN"
    security_protocol => "SASL_PLAINTEXT"
    compression_type => "${compress}"
    # 若要匿名写入，password='topic_id#${日志主题 ID}'
    sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginModule required
username='${ClslogsetID}' password='${secret_id}#${secret_key}';"
    codec => json
  }
}
```

FluentTD 调用示例

> **注意：**
> 该 demo 基于 fluentd-1.15.3版本验证；依赖的 ruby 版本 ruby=2.7.6。

FluentTD 配置

```
<match *>
  @type rdkafka2

  # brokers setting
  # TODO 域名参考 https://cloud.tencent.com/document/product/614/18940，注意内网端口9095，公网端口9096
  brokers "${domain}:${port}" # 例如 gz-producer.cls.tencentyun.com:9095

  # topic settings
  # TODO 替换日志主题 ID
  topic "${topic_id}"

  # sasl
  rdkafka_options {
    "sasl.mechanism": "PLAIN",
    "security.protocol": "sasl_plaintext",
    # TODO topic 所属日志集 ID
    "sasl.username": "${logset_id}",
    # TODO topic 所属 uin 的密钥，格式为 ${secret_id}#${secret_key}；若要匿名写入，格式为 topic_id#${日志主题 ID}
    "sasl.password": "${secret_id}#${secret_key}"
  }

  required_acks 1
  compression_codec gzip

  <format>
    @type json
  </format>

  <buffer tag>
    flush_at_shutdown true
    flush_mode interval
    flush_interval 1s
    chunk_limit_size 3MB
    chunk_full_threshold 1
```

```
    total_limit_size 1024MB
    overflow_action block
  </buffer>
</match>
```

FluentBit 调用示例

FluentBit 配置

```
[OUTPUT]
    Name kafka
    Match *
    # TODO 域名参考 https://cloud.tencent.com/document/product/614/18940，注意内网端口9095，公网端口9096
    Brokers ${domain}:${port} # 例如 gz-producer.cls.tencentyun.com:9095
    # TODO 替换日志主题 ID
    Topics ${topic_id}
    # TODO 请求消息的最大大小，最大不能超过5M
    rdkafka.message.max.bytes 5242880
    rdkafka.sasl.mechanisms PLAIN
    rdkafka.security.protocol sasl_plaintext
    # TODO 根据使用场景选择 acks 的值
    rdkafka.acks 1
    # TODO 配置压缩方式
    rdkafka.compression.codec lz4
    # TODO topic 所属日志集 ID
    rdkafka.sasl.username ${logset_id}
    # TODO topic 所属 uin 的密钥，格式为 ${secret_id}#${secret_key}；若要匿名写入，格式为 topic_id#${日志主题 ID}
    rdkafka.sasl.password ${secret_id}#${secret_key}
```

SDK 调用示例

Golang SDK 调用示例

以下以 sarama.V1_1_0_0为例，其他版本按照类似规则配置：

```
import (
    "fmt"
    "github.com/Shopify/sarama"
)

func main() {
    config := sarama.NewConfig()

    config.Net.SASL.Mechanism = "PLAIN"
    config.Net.SASL.Version = int16(1)
    config.Net.SASL.Enable = true
    // TODO 日志集 ID
    config.Net.SASL.User = "${logsetID}"
    // TODO 格式为 ${secret_id}#${secret_key}，若要匿名写入，格式为 topic_id#${日志主题 ID}
    config.Net.SASL.Password = "${secret_id}#${secret_key}"
    config.Producer.Return.Successes = true
    // TODO 根据使用场景选择 acks 的值
    config.Producer.RequiredAcks = ${acks}
    config.Version = sarama.V1_1_0_0
    // TODO 配置压缩方式
    config.Producer.Compression = ${compress}

    // TODO 服务地址；外网端口9096，内网端口9095
    producer, err := sarama.NewSyncProducer([]string{"${region}-producer.cls.tencentyun.com:9095"},
config)
```

```
    if err != nil {
        panic(err)
    }

    msg := &sarama.ProducerMessage{
        Topic: "${topicID}", // TODO 日志主题 ID
        Value: sarama.StringEncoder("goland sdk sender demo"),
    }
    // 发送消息
    for i := 0; i <= 5; i++ {
        partition, offset, err := producer.SendMessage(msg)
        if err != nil {
            panic(err)
        }
        fmt.Printf("send response; partition:%d, offset:%d\n", partition, offset)
    }

    _ = producer.Close()

}
```

Python SDK 调用示例

```
from kafka import KafkaProducer

if __name__ == '__main__':
    produce = KafkaProducer(
        # TODO 服务地址；外网端口9096，内网端口9095
        bootstrap_servers=["${region}-producer.cls.tencentyun.com:9095"],
        security_protocol='SASL_PLAINTEXT',
        sasl_mechanism='PLAIN',
        # TODO 日志集 ID
        sasl_plain_username='${logsetID}',
        # TODO 格式为 ${secret_id}#${secret_key}，若要匿名写入，格式为 topic_id#${日志主题 ID}
        sasl_plain_password='${secret_id}#${secret_key}',
        api_version=(0, 11, 0),
        # TODO 配置压缩方式
        compression_type="${compress_type}",
    )

    for i in range(0, 5):
        # 发送消息 TODO 日志主题 ID
        future = produce.send(topic="${topicID}", value=b'python sdk sender demo')
        result = future.get(timeout=10)
        print(result)
```

Java SDK 调用示例

maven 依赖：

```
<dependencies>
  <!--https://mvnrepository.com/artifact/org.apache.kafka/kafka-clients-->
  <dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>0.11.0.2</version>
  </dependency>
</dependencies>
```

代码示例：

```
import org.apache.kafka.clients.producer.*;

import java.util.Properties;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.Future;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class ProducerDemo {
    public static void main(String[] args) throws InterruptedException, ExecutionException,
TimeoutException {
        // 0.配置一系列参数
        Properties props = new Properties();
        // TODO 使用时
        props.put("bootstrap.servers", "${region}-producer.cls.tencentyun.com:9095");
        // TODO 以下值根据业务场景设置
        props.put("acks", ${acks});
        props.put("retries", ${retries});
        props.put("batch.size", ${batch.size});
        props.put("linger.ms", ${linger.ms});
        props.put("buffer.memory", ${buffer.memory});
        props.put(ProducerConfig.COMPRESSION_TYPE_CONFIG, "${compress_type}"); // TODO 配置压缩方式
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        props.put("security.protocol", "SASL_PLAINTEXT");
        props.put("sasl.mechanism", "PLAIN");
        // TODO 用户名为 logsetID；密码为 secret_id 和 secret_key 的组合，格式为 ${secret_id}#${secret_key},
        // 若要匿名写入，密码为 topic_id#${日志主题 ID}
        props.put("sasl.jaas.config",
                "org.apache.kafka.common.security.plain.PlainLoginModule required username='${logsetID}'
password='${secret_id}#${secret_key}';");

        // 1.创建一个生产者对象
        Producer<String, String> producer = new KafkaProducer<String, String>(props);

        // 2.调用 send 方法 TODO 日志主题 ID
        Future<RecordMetadata> meta = producer.send(new ProducerRecord<String, String>("${topicID}",
${message}));
        RecordMetadata recordMetadata = meta.get(${timeout}, TimeUnit.MILLISECONDS);
        System.out.println("offset = " + recordMetadata.offset());

        // 3.关闭生产者
        producer.close();
    }
}
```

C SDK 调用示例

```
// https://github.com/edenhill/librdkafka - master
#include <iostream>
#include <librdkafka/rdkafka.h>
#include <string>
#include <unistd.h>

#define BOOTSTRAP_SERVER "${region}-producer.cls.tencentyun.com:${port}"
// USERNAME 为日志集 ID
```

```
#define USERNAME "${logsetID}"
// PASSWORD 格式为 ${secret_id}#${secret_key}, 若要匿名写入，格式为 topic_id#${日志主题 ID}
#define PASSWORD "${secret_id}#${secret_key}"
// 日志主题 ID
#define TOPIC "${topicID}"
#define ACKS "${acks}"
// 配置压缩方式
#define COMPRESS_TYPE "${compress_type}"

static void dr_msg_cb(rd_kafka_t *rk, const rd_kafka_message_t *rkmessage, void *opaque) {
    if (rkmessage->err) {
        fprintf(stdout, "%% Message delivery failed : %s\n", rd_kafka_err2str(rkmessage->err));
    } else {
        fprintf(stdout, "%% Message delivery successful %zu:%d\n", rkmessage->len, rkmessage->partition);
    }
}

int main(int argc, char **argv) {
    // 1. 初始化配置
    rd_kafka_conf_t *conf = rd_kafka_conf_new();

    rd_kafka_conf_set_dr_msg_cb(conf, dr_msg_cb);

    char errstr[512];
    if (rd_kafka_conf_set(conf, "bootstrap.servers", BOOTSTRAP_SERVER, errstr, sizeof(errstr)) !=
RD_KAFKA_CONF_OK) {
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\n", errstr);
        return -1;
    }

    if (rd_kafka_conf_set(conf, "acks", ACKS, errstr, sizeof(errstr)) != RD_KAFKA_CONF_OK) {
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\n", errstr);
        return -1;
    }

    if (rd_kafka_conf_set(conf, "compression.codec", COMPRESS_TYPE, errstr, sizeof(errstr)) !=
RD_KAFKA_CONF_OK) {
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\n", errstr);
        return -1;
    }

    // 设置认证方式
    if (rd_kafka_conf_set(conf, "security.protocol", "sasl_plaintext", errstr, sizeof(errstr)) !=
RD_KAFKA_CONF_OK) {
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\n", errstr);
        return -1;
    }
    if (rd_kafka_conf_set(conf, "sasl.mechanisms", "PLAIN", errstr, sizeof(errstr)) != RD_KAFKA_CONF_OK) {
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\n", errstr);
        return -1;
    }
    if (rd_kafka_conf_set(conf, "sasl.username", USERNAME, errstr, sizeof(errstr)) != RD_KAFKA_CONF_OK) {
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\n", errstr);
        return -1;
```

```
    }
    if (rd_kafka_conf_set(conf, "sasl.password", PASSWORD, errstr, sizeof(errstr)) != RD_KAFKA_CONF_OK) {
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\n", errstr);
        return -1;
    }

    // 2. 创建 handler
    rd_kafka_t *rk = rd_kafka_new(RD_KAFKA_PRODUCER, conf, errstr, sizeof(errstr));
    if (!rk) {
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "create produce handler failed: %s\n", errstr);
        return -1;
    }

    // 3. 发送数据
    std::string value = "test lib kafka ---- ";
    for (int i = 0; i < 100; ++i) {
        retry:
        rd_kafka_resp_err_t err = rd_kafka_producev(
                rk, RD_KAFKA_V_TOPIC(TOPIC),
                RD_KAFKA_V_MSGFLAGS(RD_KAFKA_MSG_F_COPY),
                RD_KAFKA_V_VALUE((void *) value.c_str(), value.size()),
                RD_KAFKA_V_OPAQUE(nullptr), RD_KAFKA_V_END);

        if (err) {
            fprintf(stdout, "Failed to produce to topic : %s, error : %s", TOPIC, rd_kafka_err2str(err));
            if (err == RD_KAFKA_RESP_ERR__QUEUE_FULL) {
                rd_kafka_poll(rk, 1000);
                goto retry;
            }
        } else {
            fprintf(stdout, "send message to topic successful : %s\n", TOPIC);
        }

        rd_kafka_poll(rk, 0);
    }

    std::cout << "message flush final" << std::endl;
    rd_kafka_flush(rk, 10 * 1000);

    if (rd_kafka_outq_len(rk) > 0) {
        fprintf(stdout, "%d message were not deliverer\n", rd_kafka_outq_len(rk));
    }

    rd_kafka_destroy(rk);

    return 0;
}
```

C# SDK 调用示例

```
/*
 * 该 demo 只提供了最简单的使用方法，具体生产还需要结合调用放来实现
 * 在使用过程中，Demo 中留的 todo 项需要替换使用
 *
 * 注意:
 *  1. 该 Demo 基于 Confluent.Kafka/1.8.2版本验证通过
 *  2. MessageMaxBytes 最大值不能超过5M
 *  3. 该 Demo 使用同步的方式生产，在使用时也可根据业务场景调整为异步的方式
```

```
 *  4. 其他参数在使用过程中可以根据业务参考文档自己调整：
https://docs.confluent.io/platform/current/clients/confluent-kafka-
dotnet/_site/api/Confluent.Kafka.ProducerConfig.html
 *
 * Confluent.Kafka 参考文档：https://docs.confluent.io/platform/current/clients/confluent-kafka-
dotnet/_site/api/Confluent.Kafka.html
 */


using Confluent.Kafka;

namespace Producer
{
    class Producer
    {
        private static void Main(string[] args)
        {
            var config = new ProducerConfig
            {
                // TODO 域名参考 https://cloud.tencent.com/document/product/614/18940
               //  Kafka 填写，注意内网端口9095，公网端口9096
                BootstrapServers = "${domain}:${port}",
                SaslMechanism = SaslMechanism.Plain,
                // TODO topic 所属日志集 ID
                SaslUsername = "${logsetID}",
                // TODO topic 所属 uin 的密钥，格式为 ${secret_id}#${secret_key},
                // 若要匿名写入，格式为 topic_id#${日志主题 ID}
                SaslPassword = "${secret_id}#${secret_key}",
                SecurityProtocol = SecurityProtocol.SaslPlaintext,
                // TODO 根据实际使用场景赋值。可取值：Acks.None、Acks.Leader、Acks.All
                Acks          = Acks.None,
                // TODO 请求消息的最大大小，最大不能超过5M
                MessageMaxBytes = 5242880
            };

            // deliveryHandler
            Action<DeliveryReport<Null, string>> handler =
                r => Console.WriteLine(!r.Error.IsError ? $"Delivered message to {r.TopicPartitionOffset}"
: $"Delivery Error: {r.Error.Reason}");


            using (var produce = new ProducerBuilder<Null, string>(config).Build())
            {
                try
                {
                    // TODO 测试验证代码
                    for (var i = 0; i < 100; i++)
                    {
                        // TODO 替换日志主题 ID
                        produce.Produce("${topicID}", new Message<Null, string> { Value = "C# demo value"
}, handler);
                    }
                    produce.Flush(TimeSpan.FromSeconds(10));

                }
                catch (ProduceException<Null, string> pe)
                {
                    Console.WriteLine($"send message receiver error : {pe.Error.Reason}");
                }
            }
        }
```

```
        }
    }
```

版权所有：腾讯云计算（北京）有限责任公司

使用 Kafka 数据订阅采集日志

最近更新时间：2025-05-26 14:49:42

日志服务（Cloud Log Service，CLS）目前已支持主动订阅自建 Kafka 或腾讯云 CKafka 生产的日志。本文将介绍如何通过 Kafka 数据订阅将日志接入日志服务。

前提条件

- 已有可用的自建 Kafka 集群或 腾讯云 CKafka 集群 。
- Kafka 0.10.2.0以上版本。
- 已创建日志集于日志主题。具体操作请参见 日志集 和 日志主题 。

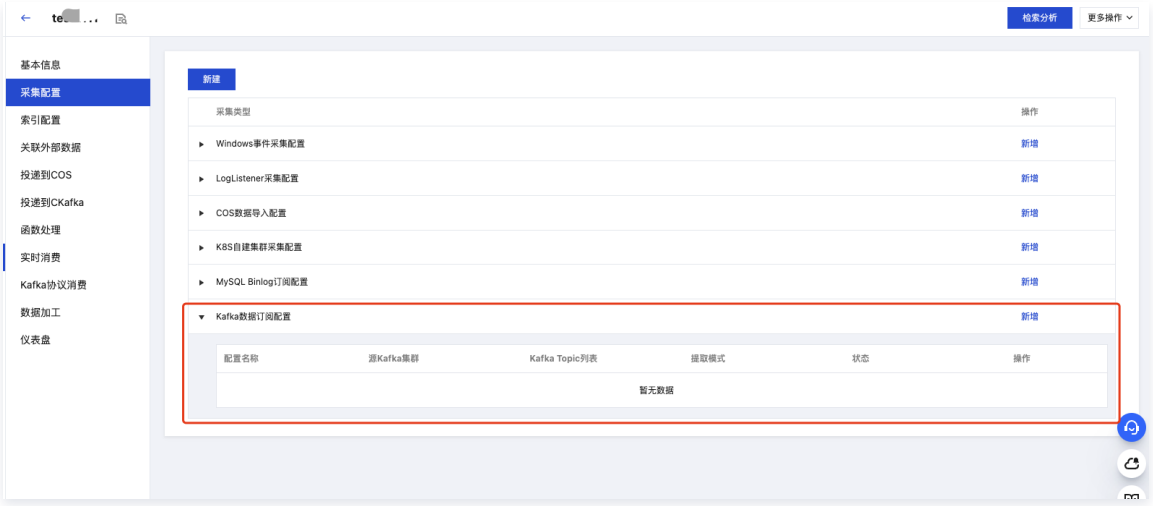
操作步骤

步骤1：登录控制台

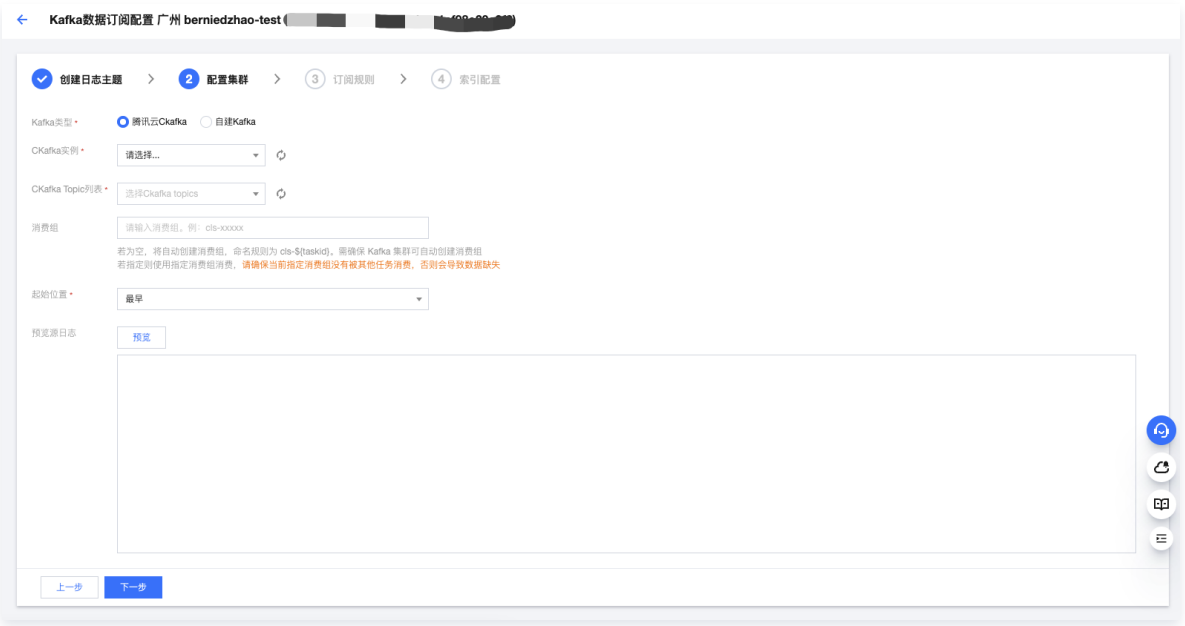
1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，进入**日志主题管理**页面。

步骤2：创建 Kafka 数据订阅任务

1. 在**日志主题管理页**中，找到目标日志主题，单击日志主题名称，进入**日志主题详情页**。
2. 在日志主题详情页中，选择**采集配置**页签，找到 **Kafka 数据订阅配置**。



3. 单击**新增**后，创建 Kafka 数据订阅任务。



步骤3：配置 Kafka 数据订阅任务

1. 在**配置集群**步骤中，首先选择目标 Kafka 类型，支持选择**腾讯 CKafka** 与**自建 Kafka**。



2. 基于选中 Kafka 类型，参考如下说明配置对应参数：

腾讯云 Ckafka

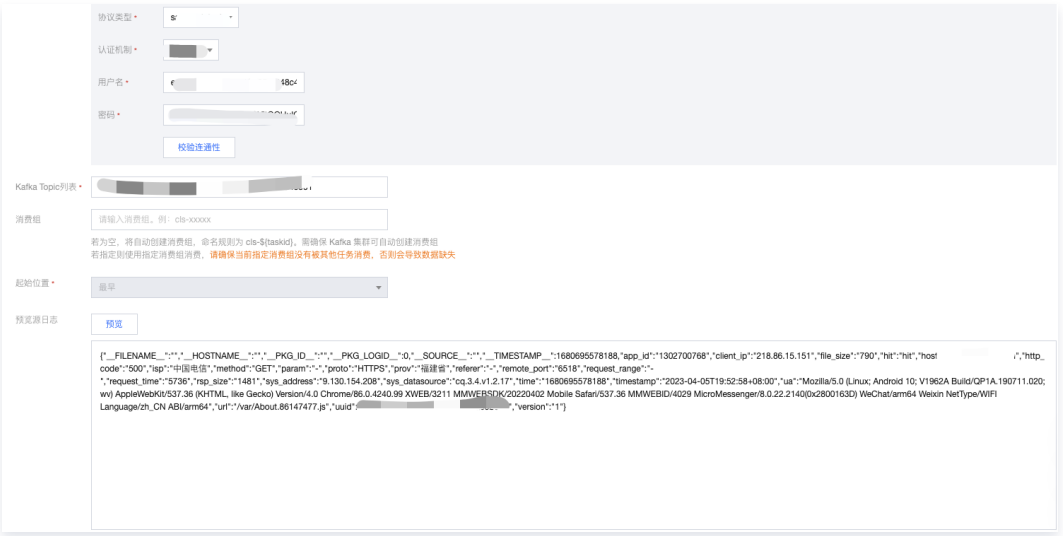
参数	是否必填	说明
Ckafka 实例	是	选择目标 Ckafka 实例。
Kafka Topic 列表	是	单选或多选 Kafka Topic。
消费组	否	若为空，将自动创建消费组，命名规则为 cls-${taskid}。若不为空，则使用指定的消费组消费。 **注意：** • 若为空，确保 Kafka 集群可自动创建消费组。 • 若不空，请确保当前指定消费组没有被其他任务消费，否则会导致数据缺失。
起始位置	是	• 最早：从最早的 offset 开始消费。 • 最晚：从最新的 offset 开始消费。 **注意：**起始位置仅支持创建订阅任务时配置，后续不可修改。

自建 Kafka

参数	是否必填	说明
访问方式	是	支持选择通过**内网地址**或**公网地址**访问您的自建 Kafka 集群。

网络服务类型	是	若访问方式为通过**内网地址**访问，需指定目标自建 Kafka 集群的网络服务类型： • CVM • CLB • 云联网（内测中，如需使用请 提交工单 我们） • 专线网关（内测中，如需使用请 提交工单 我们） **说明：** 不同网络服务类型的差别与使用方式请参见 自建 Kafka 内网接入配置说明 。
所属网络	是	当网络服务类型选择为 CVM 或 CLB 时，需选择 CVM 或 CLB 所在的 VPC 实例。
服务地址	是	填写目标 Kafka 的 IP 地址或域名，以及访问端口。 例如：192.168.0.1:9095或 test.cls.tencentyun.com:9095 **注意：** 若使用 Kafka 协议跨地域/跨账号消费其他日志主题的日志，请使用目标日志主题的公网 Kafka 协议消费域名，详情请参见 通过 Kafka 数据订阅跨账号同步日志 。
私网域名解析	否	部署在 CVM 上的 Kafka Broker 之间采用内部域名通信时，您需要在此处指定每个 Broker 对应的 CVM 域名和IP地址。详细配置场景请参见 自建 Kafka 内网接入配置说明 。
是否需要鉴权	是	目标 Kafka 集群是否需要鉴权才能访问。
协议类型	是	若目标 Kafka 集群需要鉴权才能访问，则需选择鉴权协议类型： • plaintext • sasl_plaintext • sasl_ssl • ssl
认证机制	是	若目标 Kafka 集群需要鉴权才能访问，且协议类型为 sasl_plaintext 或 sasl_ssl，则需选择鉴权认证机制： • PLAIN • SCRAM-SHA-256 • SCRAM-SHA-512
用户名/密码	是	若目标 Kafka 集群需要鉴权才能访问，且协议类型为 sasl_plaintext 或 sasl_ssl，则需填写访问目标 Kafka 集群所需的用户名与密码。
客户端 SSL 认证	是	若目标 Kafka 集群访问协议类型为 sasl_ssl 或 ssl，且访问需携带客户端 CA 证书，则需开启该配置，并选择已有证书或前往 SSL 证书服务 上传 CA 证书。
服务端 SSL 认证	是	若目标 Kafka 集群访问协议类型为 sasl_ssl 或 ssl，且访问需携带服务端证书，则需开启该配置，并选择已有证书或前往 SSL 证书服务 上传服务端证书。
Kafka Topic 列表	是	填写单个或多个 Kafka Topic；多个 Topic 通过逗号隔开。
消费组	否	若为空，将自动创建消费组，命名规则为 cls-${taskid}。若不为空，则使用指定的消费组消费。 **注意：** • 若为空，确保 Kafka 集群可自动创建消费组。 • 若不空，请确保当前指定消费组没有被其他任务消费，否则会导致数据缺失。
起始位置	是	• 最早：从最早的 offset 开始消费。 • 最晚：从最新的 offset 开始消费。 **注意**：起始位置仅支持创建订阅任务时配置，后续不可修改。

3. 完成集群配置之后，可单击**预览**，如下图所示：



4. 确认集群配置无误后，单击下一步进入**订阅规则**配置步骤。
5. 在**订阅规则**配置步骤中，配置如下参数：

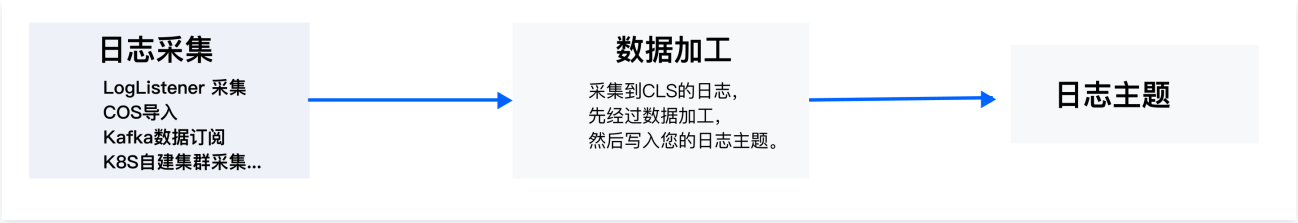
参数	是否必填	说明
配置名称	是	Kafka 数据订阅配置名称。
数据提取模式	是	可选择 JSON、单行全文、单行完全正则三种提取模式，详情请参见 数据提取模式 。
日志样例	是	若数据提取模式选择为单行完全正则，需手动填写或自动获取一条日志样例，用于验证正则表达式，并提取键值。
正则表达式	是	若数据提取模式选择为单行完全正则，需手动填写或自动生成一条正则表达式，系统会根据您填写的正则表达式验证并提取键值。自动生成正则表达式的具体操作，详情请参见 正则表达式自动生成 。
抽取结果	是	若数据提取模式选择为单行完全正则，需配置或修改基于正则表达式提取出来的字段的字段名。
手动验证	否	若数据提取模式选择为单行完全正则，可选择额外填写单条或多条日志样例来验证正则表达式的正确性。
上传解析失败日志	是	若数据提取模式选择为 JSON 或单行完全正则，若开启上传解析失败日志，在日志解析失败时，LogListener 会上传解析失败的日志；若关闭，则会丢弃解析失败的日志。
解析失败日志的键名称	是	若开启了上传解析失败日志，您可指定一个字段名作为 Key，解析失败的日志将作为您指定字段的 Value 进行上传。
编码格式	是	基于您的日志，您可选择以下两种编码格式： • UTF-8 • GBK
使用默认时间	是	开启后将使用系统当前时间或 Kafka 消息时间戳作为日志的时间戳；关闭则将使用日志中的时间字段作为日志时间戳。
默认时间来源	是	当开启了使用默认时间，您可以选择以下两默认事件作为日志时间戳： • 系统当前时间 • Kafka 消息时间戳
时间字段	是	当关闭了使用默认时间，且数据提取模式为 JSON 或正则时，您可以指定日志中代表时间的字段名，该字段的值将会用作日志的时间戳。
时间提取正则	是	当关闭了使用默认时间，且数据提取模式单行全文时，您可通过正则表达式定义日志中代表时间的字段。 注意：若正则匹配到多个字段，将使用第一个。 例子：日志原文为：message with time 2022-08-08 14:20:20，则您可以设置提取时间正则为 \d\d\d\d-\d\d-\d\d\d\d:\d\d:\d\d
时间字段格式	是	当关闭了使用默认时间，并确认了日志中的时间字段，需进一步指定时间格式，用于解析时间字段的值。详情请参见 配置时间格式 。

时间字段时区	是	当关闭了使用默认时间，并确认了日志中的时间字段与格式，需选择以下两种不同的时区标准： • UTC（协调世界时） • GMT（格林威治标准时间）
解析失败使用时间	是	当关闭了使用默认时间，若时间提取正则或时间字段格式解析失败，支持用户选择以下两种默认时间代替作为日志时间戳： • 系统当前时间 • Kafka 消息时间戳
过滤器	否	过滤器的目的是根据业务需要添加日志采集过滤规则，以帮助您筛选出有价值的日志数据。支持以下过滤规则： • 等于：仅采集指定字段值匹配指定字符的日志。支持完全匹配，或正则匹配。 • 不等于：仅采集指定字段值不匹配指定字符的日志。支持完全匹配，或正则匹配。 • 字段存在：仅采集指定字段存在的日志。 • 字段不存在：仅采集指定字段不存在的日志。 例如，您希望原始 JSON 格式日志内容中 response_code 为400或500的所有日志数据被采集，那么 key 处配置 response_code，过滤规则选择**等于**，value 处配置400|500。 **注意：** 多条过滤条件之间关系是"与"逻辑，若同一 key 名配置多条过滤条件，规则会被覆盖。
Kafka 元数据	否	支持勾选以下4种 Kafka 相关元数据与日志一起上传： • kafka_topic • kafka_partition • kafka_offset • kafka_timestamp **注意：** 若日志原文中存在与以上元数据同名的字段，将会被覆盖。

6. 完成订阅规则配置后，您可单击**预览**，预览导出结果。如下图所示：



如您需要对采集到的 CLS 日志进一步处理，如结构化、脱敏、过滤，然后再写入日志主题，您可单击页面下方的**数据加工**，添加数据加工，最后配置索引。



说明：

- 数据加工相关操作请参见 创建加工任务 文档中的 前置数据加工 TAB 页面。
- 数据加工脚本的编写请参见 数据加工函数总览，或者 加工实战案例。
- 数据加工会产生费用，详情请参见 计费概述。

7. 预览无误后，单击**下一步**，进入**索引配置**步骤。
8. 在**索引配置**页面，设置如下信息。



- **索引状态**：确认是否开启。

  > **注意：**
  > 若要对日志进行检索，索引状态需未开启，否则无法检索日志。

- **全文索引**：全文索引将日志全文整体切分为多个分词进行索引构建，检索时直接通过关键词进行检索（即全文检索），例如使用 `error` 检索所有出现过 error 的日志。

配置项	功能描述
全文分词符	对原始日志进行分词的字符集合，仅支持英文符号，控制台默认分词符为 `@&?|#()='",;:<>[]{}/ \n\t\r\\` 。
大小写敏感	检索时是否对大小写敏感。例如日志为 Error，若大小写敏感，则使用 error 无法检索到该条日志。
是否包含中文	日志中包含中文且需要对中文进行检索时可开启该功能。例如日志为“用户登录接口超时”，若未开启该功能，搜索 `"超时"` 无法检索到该日志，只有完整地搜索 `"用户登录接口超时"` 才能检索到该日志，开启该功能后便可通过搜索“超时”检索到该日志。

- **键值索引**：键值索引将原始日志按字段（即 key:value）分别切分为多个分词进行索引构建，检索时基于键值方式进行检索（即键值检索），例如使用 `level:error AND timeCost:>1000` 检索日志级别（level）为 error 且耗时（timeCost）大于1000ms 的日志。某些日志中还包含一类特殊的“元数据字段”，这类字段配置索引的方式与普通字段一致。

配置项	功能描述
字段名称	字段名称，单个日志主题键值索引最多可添加300个字段。 仅支持字母、数字、下划线和-./@，且不能以下划线开头。

字段类型	字段的数据类型，包括 text、long、double 三种类型。 • text 类型支持使用通配符进行模糊检索，不支持范围比较。 • long、double 类型支持范围比较，不支持模糊检索。
分词符	对字段值进行分词的字符集合，仅支持英文符号，控制台默认分词符为 `@&?|#()='",;:<>[]{}/ \n\t\r\\` 。
包含中文	字段中包含中文且需要对中文进行检索时可开启该功能。例如日志为“message:用户登录接口超时”，若未开启该功能，使用 `message:"超时"` 无法检索到该日志，只有使用 `message:"用户登录接口超时"` 才能检索到该日志，开启该功能后便可使用 `message:"超时"` 检索该日志。
开启统计	开启后可对该字段使用 SQL 进行统计分析，text 类型字段开启统计时，如果值过长，仅前32766个字符参与统计运算。 开启统计不会额外产生费用，建议您开启。
大小写敏感	检索时是否对大小写敏感例如日志为“level:Error”，若大小写敏感，则使用 `level:error` 无法检索到该条日志。

说明：
索引相关的更多详情，请参见 索引配置 。

9. 完成索引配置后，单击**提交**，即可完成 Kafka 数据订阅任务的创建。

步骤4：查看 Kafka 数据订阅任务

完成 Kafka 数据订阅任务后，您可在日志主题详情页的**采集配置**页签中找到所有已创建的 Kafka 数据订阅任务。



步骤5：检索分析日志

完成 Kafka 数据订阅任务后，即可开始使用日志的检索与分析，以及更高阶的仪表盘告警等功能了。

- 检索分析
- 监控告警
- 仪表盘
- 数据加工

规格与限制

规格与限制请参见 Kafka 数据订阅规格与限制 。

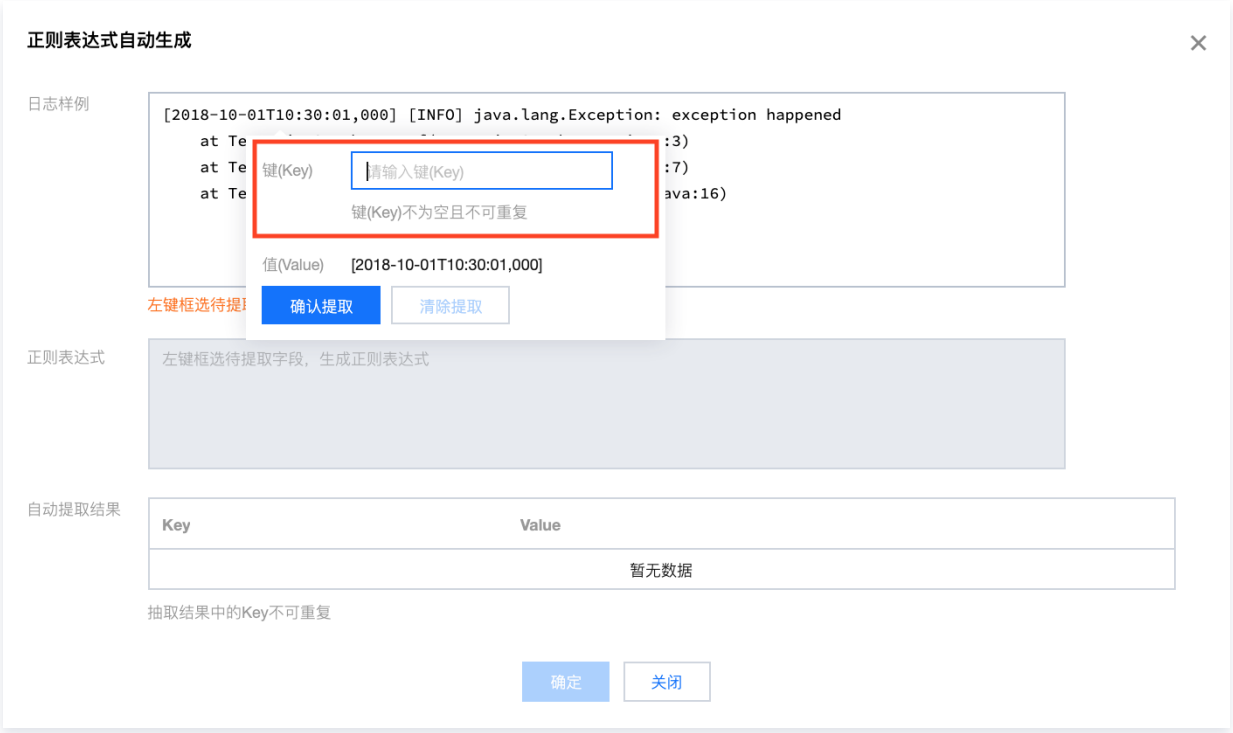
最佳实践

使用 Kafka 数据订阅进行跨账号跨日志主题数据迁移 。

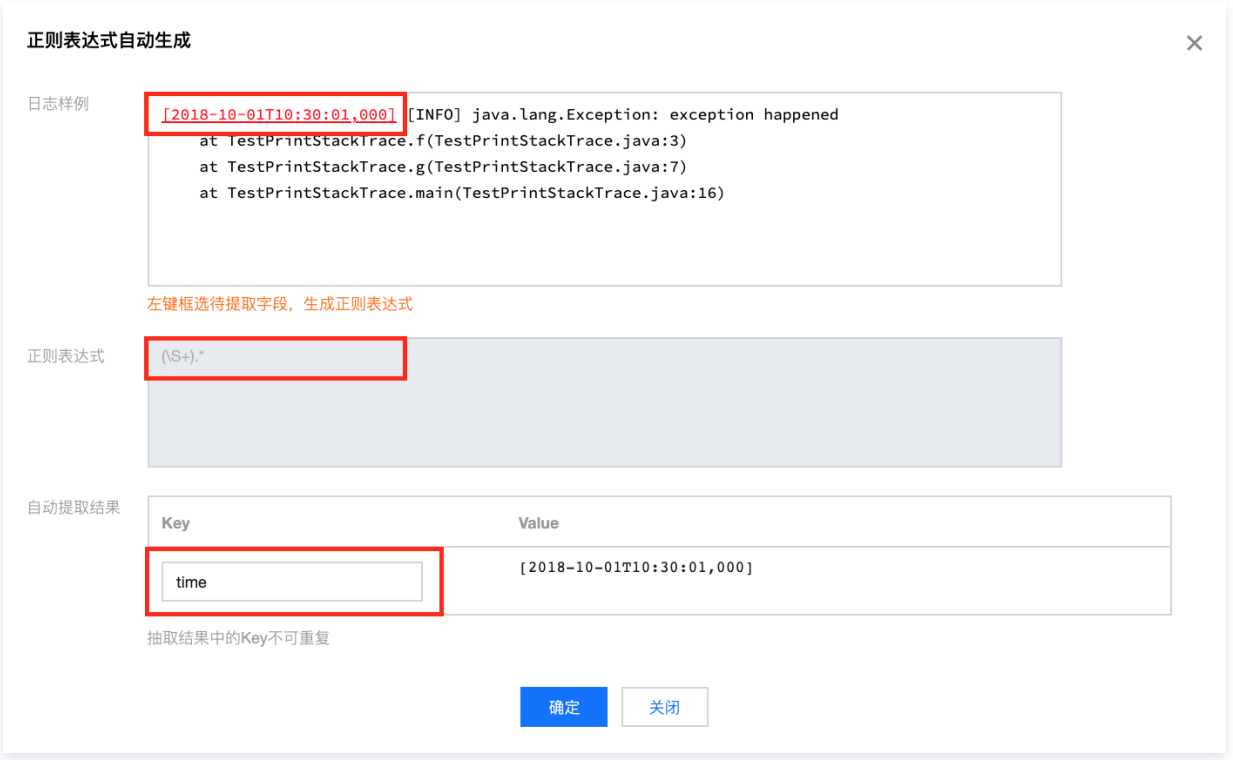
附录

正则表达式自动生成

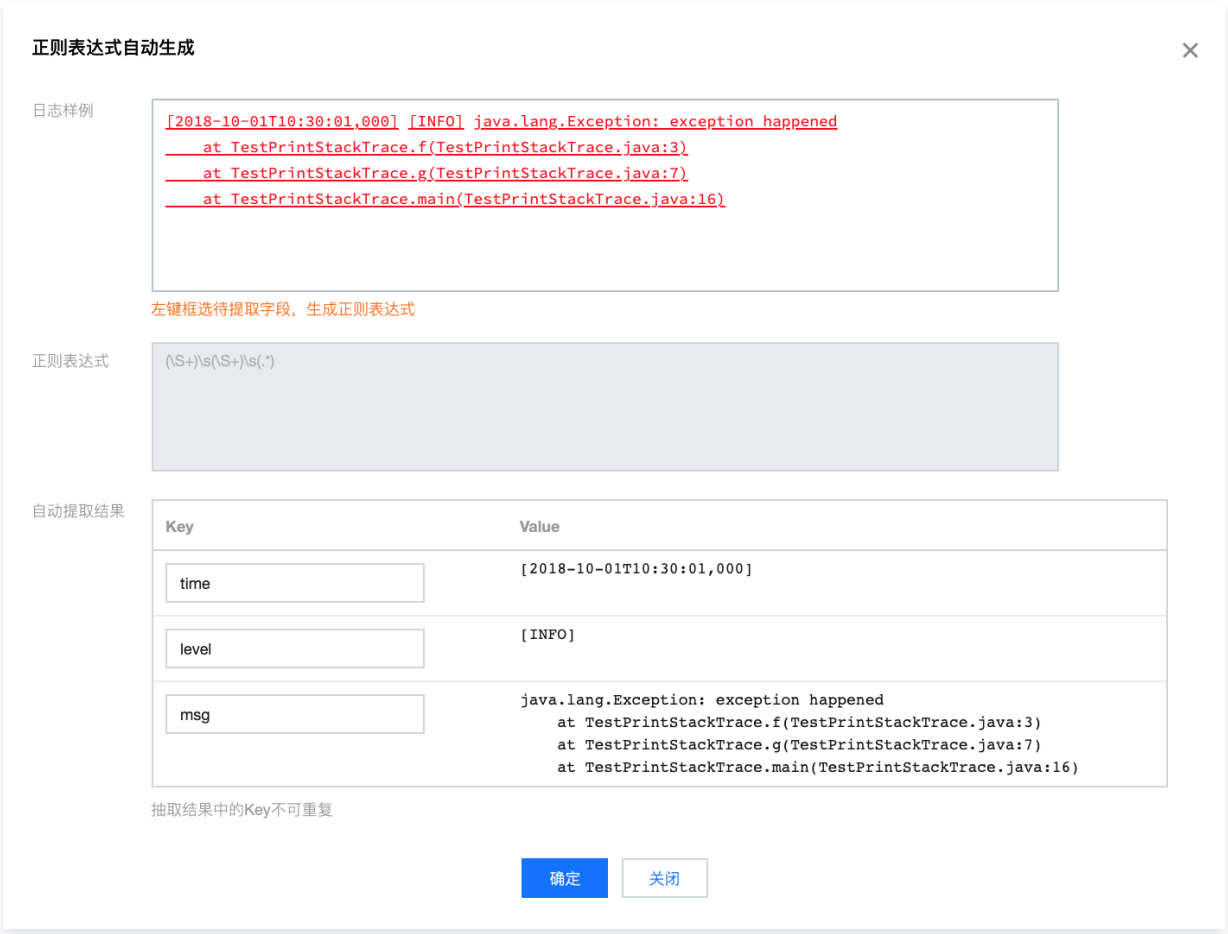
1. 在弹出的**正则表达式自动生成**模态视图中，根据实际的检索分析需求，选中需要提取 key-value 的日志内容，并在弹出的文本框中，输入键（key）名，单击**确认提取**。如下图所示：



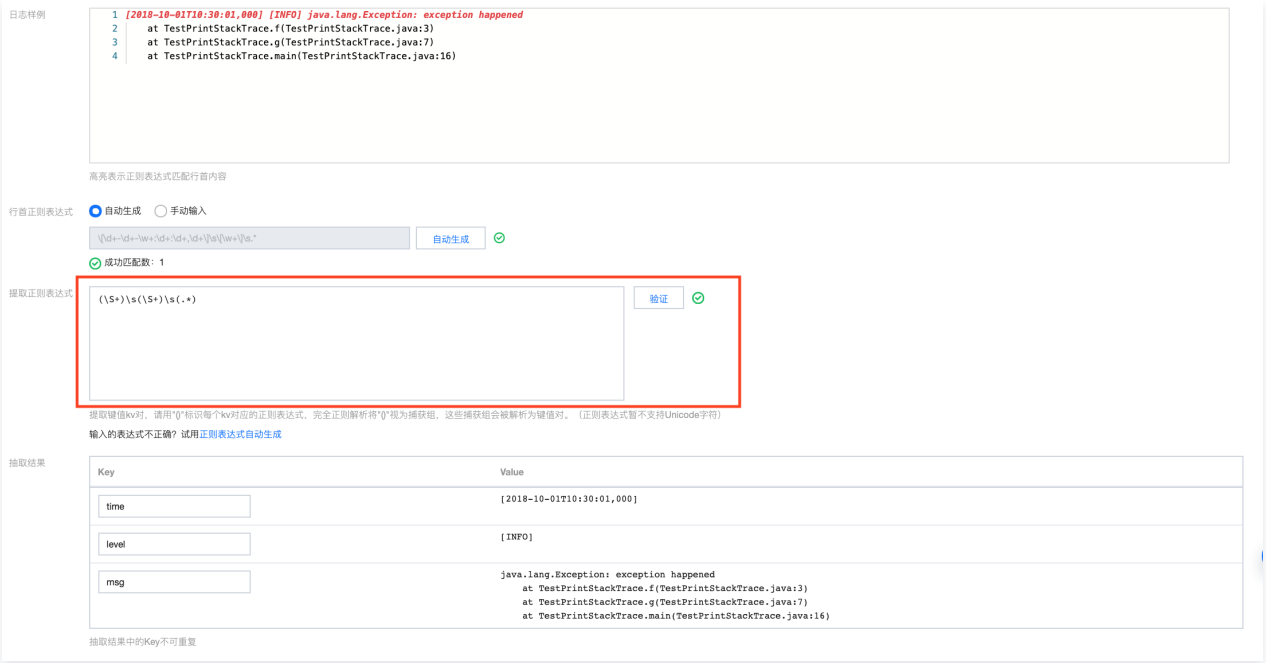
系统将自动对该部分内容提取一个正则表达式，自动提取结果会出现在 key-value 表格中。如下图所示：



2. 重复步骤1，直到提取完所有的 key-value 对。如下图所示：



3. 单击**确定**，系统将根据提取好的 key-value 对自动生成完整的正则表达式。如下图所示：



数据提取模式

Kafka 数据订阅提供多种解析方式，如下表所示：

JSON

假设您的一条 JSON 日志原始数据为：

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34
+0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost":"unix:/tmp/php-
cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","request":"POST
/event/dispatch HTTP/1.1","xff":"-","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0
(Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

单行全文

单行全文日志是指一行日志内容为一条完整的日志。日志服务在采集的时候，将使用换行符 \n 来作为一条日志的结束符。为了统一结构化管理，每条日志都会存在一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条日志原始数据为：

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

采集到日志服务的数据为：

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

单行完全正则

单行完全正则格式通常用来处理结构化的日志，指将一条完整日志按正则方式提取多个 key-value 的日志解析模式。
假设一条日志原始数据为：

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703
"http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum"
"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0"  0.354 0.354
```

配置的正则表达式为如下：

```
(\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"
([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
```

采集到日志服务的数据为：

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

自建 Kafka 内网接入配置说明

CVM 类型接入

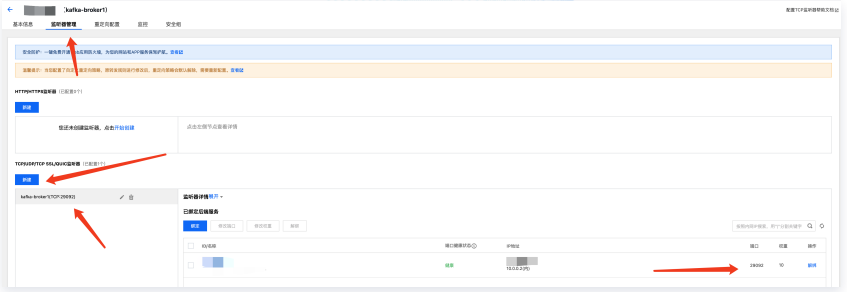
- 如果您的 kafka broker 节点均部署在同一个 VPC 下的 CVM 上，可以使用该种方式进行接入。
- 单个 CVM 上的 broker 节点配置信息如下。例如：CVM 节点在 VPC 内的 IP 地址为10.0.0.2，broker 端口为9092。其它节点采用相同配置并替换掉 IP 即可。

```
listener.security.protocol.map=CVM:PLAINTEXT
listeners=CVM://10.0.0.2:9092
advertised.listeners=CVM://10.0.0.2:9092
```

- 在 配置订阅任务 时，选择 自建 Kafka，**内网地址**访问方式，**CVM** 网络服务类型，那么 VPC 选择该 CVM 对应的 VPC ID，内网服务地址填写10.0.0.2:9092即可。

CLB 类型接入

- 如果您的 kafka broker 节点均部署在同一个 VPC 下的 CVM/TKE 节点，可以使用该种方式进行接入。
- 需要先按每个 broker 一对一 创建对应的 CLB 实例，并 配置 TCP 类型的监听器，监听端口最好与 kafka broker 的端口一致，便于管理。



- 单个 CVM/TKE 上 broker 节点配置信息如下所示。例：CVM/TKE 在 VPC 内 IP 地址为10.0.0.2，节点对应的 CLB 地址为10.0.0.12，broker 端口为29092。其它节点同样配置并替换掉 IP 即可。

```
listener.security.protocol.map=CLB:PLAINTEXT
listeners=CLB://10.0.0.2:29092
advertised.listeners=CLB://10.0.0.12:29092
```

- 在 配置订阅任务 时，选择 自建 Kafka，**内网地址**访问方式，**CLB** 网络服务类型，那么 VPC 选择该 CLB 对应的 VPC ID，内网服务地址填写10.0.0.12:29092即可。
- 在 TKE 场景下，可能会在 broker 配置中使用私网域名作为监听地址，此时需要使用私有域名解析能力进行处理。详请请参见 私网域名解析 。

私网域名解析

1. 有些场景下，期望使用私网域名进行 kafka 的访问，例如 TKE 场景，节点 IP 是不固定的，需要配置域名进行访问，如下配置：

```
listener.security.protocol.map=DOMAIN:PLAINTEXT
listeners=DOMAIN://10.0.0.2:9092
advertised.listeners=DOMAIN://broker1.cls.tencent.com:9092
```

2. 为了便于此类场景直接接入 CLS，CLS 提供了私网域名解析功能来做域名解析映射。
   例如这里使用 CLB 来关联后端 rs，CLB 地址为10.0.0.12，那么就需要在 自建 Kafka 订阅任务配置 中增加一条私网域名解析，即：
   - 域名：broker1.cls.tencent.com
   - IP：10.0.0.12
3. 如果多个 broker 节点都是使用这种访问策略，则均需添加私网域名解析。
4. 配置了私网域名解析后，自建 Kafka 订阅任务配置 中内网服务地址则只需填写 broker1.cls.tencent.com:9092 即可。

匿名日志采集

最近更新时间：2024-07-01 16:46:11

CLS 日志服务支持免密通过 JS、HTTP 和 Kafka 协议将日志上传至 CLS。

使用场景

- 用户使用的浏览器、操作系统、分辨率等信息。
- 用户浏览行为记录（例如：用户在网站上的点击行为、购买行为等信息）。
- 用户在 App 中的停留时间、是否活跃等信息。

操作步骤

步骤一：日志主题开通匿名访问

> **注意：**
> 使用**匿名采集**则表示目标投递日志主题打开匿名访问，没有经过有效鉴权，可能产生脏数据。

1. 登录 日志服务 控制台。
2. 在左侧导航栏中，单击**日志主题**，在日志主题管理页中，单击**创建日志主题**，或找到已有日志主题并单击**编辑**，即可创建/编辑日志主题。
3. 在日志主题创建或编辑弹窗中，找到**匿名访问**配置项， 并开启开关。



4. 开启开关后，在**匿名操作**中选择日志匿名上传方式。 当前支持的匿名上传操作为：**JS/HTTP 日志上传、Kafka 协议上传。**



步骤二：上传日志

开通匿名访问后，基于步骤一中日志主题配置的匿名操作，您可以通过以下方法上传日志到日志主题中。

方法1：通过浏览器 JavaScript SDK 上传日志

详细信息您可参见 通过浏览器 JavaScript SDK 上传日志 。

方法2：通过小程序 JavaScript SDK上传日志

详细信息您可参见 通过小程序 JavaScript SDK 上传日志 。

方法3：通过 HTTP GET 请求上传日志

```
# 支持https
curl --request GET 'http://${region}.cls.tencentcs.com/track?topic_id=${topic_id}&key1=val1&key2=val2'
```

参数	是否必填	说明
${region}	是	日志主题所在地域。更多信息，请参见 可用地域。例子：ap-guangzhou
${topic_id}	是	日志主题 ID
key1=val1&key2=val2	是	您要上传到日志服务的键值对（Key-Value），支持设置多个键值对，由&隔开。请确保长度小于16KB

方法4：通过 HTTP POST 请求上传日志

GET 请求上传日志时，单个请求只能写入一条日志。如果请求的数据量比较大，可以使用 POST 方法上传数据。

接口说明

- 适用于在网页或者客户端采集日志的场景。
- 支持将多条日志合并为一次请求。
- 该接口不支持同时写入多个 Topic 的日志数据。

请求头

该接口使用公共请求头，无特殊请求头，详情可参见 公共请求参数 文档。

请求语法

```
# 支持https
POST http://${region}.cls.tencentcs.com/tracklog?topic_id=${topic_id} HTTP/1.1
```

参数	是否必填	说明
${region}	是	日志主题的地域。更多信息，请参见 可用地域。例子：ap-guangzhou
${topic_id}	是	日志主题 ID

示例

```
POST /tracklog?topic_id={topic_id} HTTP/1.1
Host:ap-guangzhou.cls.tencentcs.com
Content-Type:application/json
# 请勿修改JSON结构， 仅需修改TODO即可
{
    "logs": [{
        # TODO：定义需要上传的key-value键值
        "contents": {
            "key1": "value1",
            "key2": "value2"
        },
        # TODO：定义上传时间戳
        "time": 123456789
```

```
    }],
    # TODO: 定义来源IP
    "source": "127.0.0.1"
}
```

方法5：通过 Kafka 协议上传日志

详细信息您可参见 使用 Kafka 协议上传日志 。

> **说明：**
> 通过 Kafka 协议匿名上传，在生产端配置中， password 需以 `topic_id#${日志主题 ID}` 的格式写入，例如：topic_id#76c63473-c496-466b-XXXX-XXXX。

使用 Logback Appender 上传日志

最近更新时间：2025-03-14 15:53:22

简介

日志服务（Cloud Log Service，CLS）目前已支持使用 Logback Appender SDK 上传日志到 CLS。

背景信息

Logback 是 Apache 的一个开源项目。通过使用 Logback，您可以控制日志信息输送的目的地是控制台、文件、GUI 组件，甚至是套接口服务器、NT 的事件记录器、UNIX Syslog 守护进程等。此外，您还可以通过一个 logback 配置文件来灵活地进行配置，而不需要修改应用的代码。

功能优势

- 日志不落盘：产生数据通过网络发给服务端。
- 无需改造：对已使用 Logback 应用，只需简单配置即可采集。
- 异步非阻塞：高并发设计，后台异步发送，适合高并发写入。
- 资源可控制：可以通过参数控制 producer 用于缓存待发送数据的内存大小，同时还可以配置用于执行数据发送任务的线程数量。
- 自动重试： 对可重试的异常，支持配置重试次数。
- 优雅关闭： 退出前会将日志全量进行发送。
- 感知日志上报结果："运行过程中产生的异常通过 AddError 输出"。

工程引入和配置

maven 工程中引入依赖

```
<dependency>
    <groupId>com.tencentcloudapi.cls</groupId>
    <artifactId>tencentcloud-cls-logback-appender</artifactId>
    <version>1.0.8</version>
</dependency>
```

修改 logback 配置文件

```
  <appender name="LoghubAppender" class="com.tencentcloudapi.cls.LoghubAppender">
        <!--必选项-->
        <!--域名配置-- 详见https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D>
        <endpoint>${region}.cls.tencentcs.com</endpoint>
        <!--密钥信息-- 前往https://console.cloud.tencent.com/cam/capi获取>
        <!--请确保密钥关联的账号具有相应的SDK日志上传权限-- 查看权限策略前往
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE >
        <accessKeyId>${SecretID}</accessKeyId>
        <accessKeySecret>${SecretKey}</accessKeySecret>
        <!--日志主题ID-->
        <topicId>${topicId}</topicId>

        <!-- 可选项 详见 '参数说明'-->
        <totalSizeInBytes>104857600</totalSizeInBytes>
        <maxBlockMs>0</maxBlockMs>
        <sendThreadCount>8</sendThreadCount>
        <batchSizeThresholdInBytes>524288</batchSizeThresholdInBytes>
        <batchCountThreshold>4096</batchCountThreshold>
        <lingerMs>2000</lingerMs>
        <retries>10</retries>
        <maxReservedAttempts>11</maxReservedAttempts>
        <baseRetryBackoffMs>100</baseRetryBackoffMs>
        <maxRetryBackoffMs>50000</maxRetryBackoffMs>
```

```
        <includeMessage>false</includeMessage>

        <!-- 可选项 设置时间格式 -->
        <timeFormat>yyyy-MM-dd'T'HH:mm:ssZ</timeFormat>
        <timeZone>Asia/Shanghai</timeZone>
        <customFields>{"${自定义key}":"${自定义value}"}</customFields>
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger - %msg</pattern>
        </encoder>
        <mdcFields>${MDC字段}</mdcFields>
    </appender>
```

参数说明

参数	描述	示例
totalSizeInBytes	单个 producer 实例能缓存的日志大小上限，默认为100MB	totalSizeInBytes=104857600
maxBlockMs	如果 producer 可用空间不足，调用者在 send 方法上的最大阻塞时间，默认为60秒。为了不阻塞打印日志的线程，强烈建议将该值设置成0	maxBlockMs=0
sendThreadCount	执行日志发送任务的线程池大小，默认为可用处理器个数	sendThreadCount=8
batchSizeThresholdInBytes	当一个 ProducerBatch 中缓存的日志大小大于等于 batchSizeThresholdInBytes 时，该 batch 将被发送，默认为512 KB，最大可设置成5MB	batchSizeThresholdInBytes=524288
batchCountThreshold	当一个 ProducerBatch 中缓存的日志条数大于等于 batchCountThreshold 时，该 batch 将被发送，默认为4096，最大可设置成40960	batchCountThreshold=4096
lingerMs	一个 ProducerBatch 从创建到可发送的逗留时间，默认为2秒，最小可设置成100毫秒	lingerMs=2000
retries	• 如果某个 ProducerBatch 首次发送失败，能够对其重试的次数，默认为10次 • 如果 retries 小于等于0，该 ProducerBatch 首次发送失败后将直接进入失败队列	retries=10
maxReservedAttempts	该参数越大能让您追溯更多的信息，但同时也会消耗更多的内存	maxReservedAttempts=11
baseRetryBackoffMs	首次重试的退避时间，默认为100毫秒。Producer 采用指数退避算法，第 N 次重试的计划等待时间为 baseRetryBackoffMs * 2^(N-1)	baseRetryBackoffMs=100
maxRetryBackoffMs	重试的最大退避时间，默认为50秒	maxRetryBackoffMs=50000
timeFormat	设置时间格式	• 精确到秒：yyyy-MM-dd'T'HH:mm:ssZ • 精确到毫秒：yyyy-MM-dd'T'HH:mm:ss.SSSZ **注意：** 仅 Logback 版本为1.0.4及以上支持毫秒时间。
mdcFields	指定要在 Appender 中包含的 MDC（Mapped Diagnostic Context）字段。这些字段将作为日志的一部分上传至 CLS。	-
customFields	定义自定义字段，这些字段将作为日志的一部分上传至 CLS	{"app":"my_application"}

includeMessage	当 encoder 不为空时，是否要包含 message 字段，默认为 true	true

使用 Log4j Appender 上传日志

最近更新时间：2024-09-20 10:26:11

简介

日志服务（Cloud Log Service，CLS）目前已支持使用 Log4j Appender 上传日志到 CLS。

背景信息

Log4j 是 Apache 的一个开源项目。通过使用 Log4j，我们可以控制日志信息输送的目的地是控制台、文件、GUI 组件，甚至是套接口服务器、NT 的事件记录器、UNIX Syslog 守护进程等；我们可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们可以更加细致地控制日志的生成过程。此外，我们还可以通过一个配置文件来灵活地进行配置，而不需要修改应用的代码。
Log4j 由三个重要的组件构成：

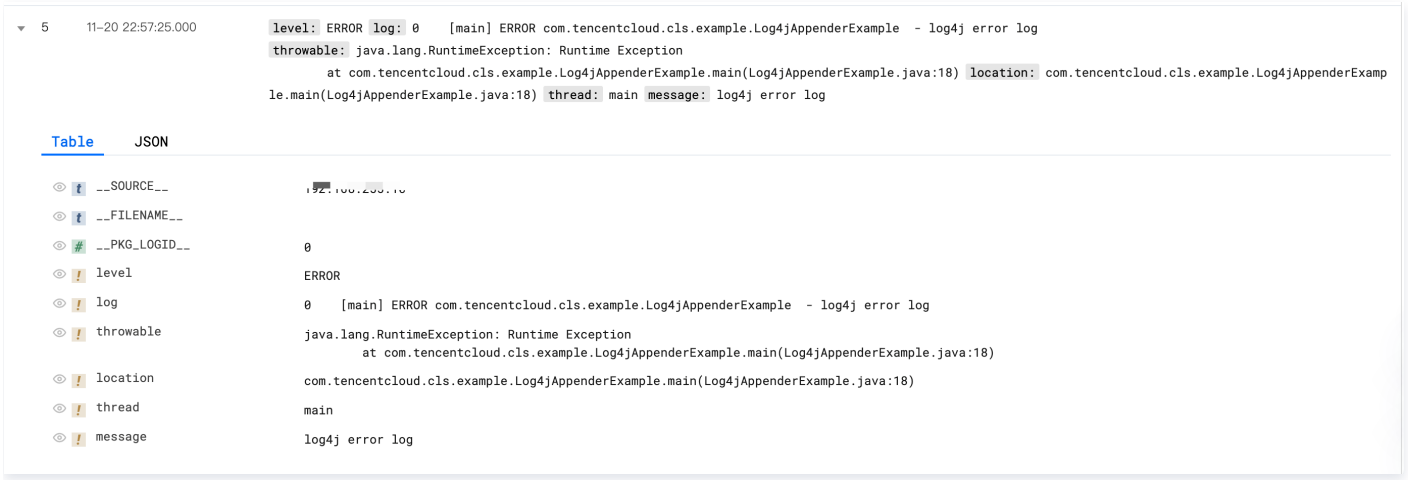
- **日志信息的优先级**
  日志信息的优先级从高到低分别为 ERROR、WARN、INFO和DEBUG，分别用来指定这条日志信息的重要程度。
- **日志信息的输出目的地**
  日志信息的输出目的地指定了日志将打印到控制台还是文件中。
- **日志信息的输出格式**
  输出格式控制了日志信息的显示内容。

功能优势

- 日志不落盘：产生数据通过网络发给服务端。
- 无需改造：对已使用 Log4J 应用，只需简单配置即可采集。
- 异步非阻塞：高并发设计，后台异步发送，适合高并发写入。
- 资源可控制：可以通过参数控制 producer 用于缓存待发送数据的内存大小，同时还可以配置用于执行数据发送任务的线程数量。
- 自动重试：对可重试的异常，支持配置重试次数。
- 优雅关闭：退出前会将日志全量进行发送。
- 感知日志上报结果：通过 org.apache.log4j.helpers.LogLog 记录运行过程中产生的异常，在默认情况下 LogLog 会将信息输出到控制台。

使用 Tencent CLS Log4j Appender

通过 Tencent CLS Log4j Appender，您可以控制日志的输出目的地为腾讯云日志服务。写入日志服务中的日志的样式如下：



字段	简介
__SOURCE__	来源 IP
__FILENAME__	文件名称
level	日志级别
location	日志打印语句的代码位置
message	日志内容
throwable	日志异常信息（只有记录了异常信息，这个字段才会出现）

thread	线程名称
time	日志打印时间（可以通过 timeFormat 或 timeZone 配置 time 字段呈现的格式和时区）
log	自定义日志格式

工程引入和配置

maven 工程中引入依赖

```
<dependency>
    <groupId>com.tencentcloudapi.cls</groupId>
    <artifactId>tencentcloud-cls-log4j-appender</artifactId>
    <version>1.0.3</version>
</dependency>
```

修改 Log4J 配置文件

```
#loghubAppender
log4j.appender.loghubAppender=com.tencentcloudapi.cls.LoghubAppender
#日志服务的 http 地址，必选参数
log4j.appender.loghubAppender.endpoint=ap-guangzhou.cls.tencentcs.com
#用户身份标识，必选参数
#请确保用户身份标识（密钥）关联的账号具有相应的SDK日志上传权限，查看权限策略前往
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE >
log4j.appender.loghubAppender.accessKeyId=
log4j.appender.loghubAppender.accessKeySecret=
#设置log字段的格式，必选参数
log4j.appender.loghubAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.loghubAppender.layout.ConversionPattern=%-4r [%t] %-5p %c %x - %m%n
#指定日志主题，必选参数
log4j.appender.loghubAppender.topicID =
#指定日志来源，可选参数
log4j.appender.loghubAppender.source =
#单个 producer 实例能缓存的日志大小上限，默认为 100MB。
log4j.appender.loghubAppender.totalSizeInBytes=104857600
#如果 producer 可用空间不足，调用者在 send 方法上的最大阻塞时间，默认为 60 秒。为了不阻塞打印日志的线程，强烈建议将该值设置
成 0。
log4j.appender.loghubAppender.maxBlockMs=0
#执行日志发送任务的线程池大小，默认为可用处理器个数。
log4j.appender.loghubAppender.sendThreadCount=8
#当一个 ProducerBatch 中缓存的日志大小大于等于 batchSizeThresholdInBytes 时，该 batch 将被发送，默认为 512 KB，最大
可设置成 5MB。
log4j.appender.loghubAppender.batchSizeThresholdInBytes=524288
#当一个 ProducerBatch 中缓存的日志条数大于等于 batchCountThreshold 时，该 batch 将被发送，默认为 4096，最大可设置成
40960。
log4j.appender.loghubAppender.batchCountThreshold=4096
#一个 ProducerBatch 从创建到可发送的逗留时间，默认为 2 秒，最小可设置成 100 毫秒。
log4j.appender.loghubAppender.lingerMs=2000
#如果某个 ProducerBatch 首次发送失败，能够对其重试的次数，默认为 10 次。
#如果 retries 小于等于 0，该 ProducerBatch 首次发送失败后将直接进入失败队列。
log4j.appender.loghubAppender.retries=10
#该参数越大能让您追溯更多的信息，但同时也会消耗更多的内存。
log4j.appender.loghubAppender.maxReservedAttempts=11
#首次重试的退避时间，默认为 100 毫秒。
#Producer 采用指数退避算法，第 N 次重试的计划等待时间为 baseRetryBackoffMs * 2^(N-1)。
log4j.appender.loghubAppender.baseRetryBackoffMs=100
#重试的最大退避时间，默认为 50 秒。
log4j.appender.loghubAppender.maxRetryBackoffMs=50000
```

```
#设置时间格式，可选参数
log4j.appender.loghubAppender.timeFormat=yyyy-MM-dd'T'HH:mm:ssZ
#设置时区为东八区，可选参数
log4j.appender.loghubAppender.timeZone=Asia/Shanghai
#输出 DEBUG 级别及以上的消息
log4j.appender.loghubAppender.Threshold=DEBUG
```

Log4j Appender SDK

- Log4j 1.x版本：请使用 tencentcloud-cls-log4j-appender 。
- Log4j 2.x版本：请使用 tencentcloud-cls-log4j2-appender 。

使用 SDK 上传日志

最近更新时间：2025-01-09 14:12:32

简介

为了能让您更高效地使用日志服务（Cloud Log Service，CLS），日志服务专门为日志上传量身打造了多个语言版本的 SDK（Software Development Kit），您可以根据业务需求选择语言版本使用。

使用前须知

- SDK 对日志服务的数据接入接口做了统一封装，降低了使用上传日志的难度，您可以使用对应语言的 SDK 进行日志上传。
- 实现日志服务日志的 ProtoBuffer 格式封装，让您在写入日志时不需要关心 ProtoBuffer 格式的具体细节。
- 实现日志服务 API 中定义的压缩方法，让您不用关心压缩实现的细节。部分语言的 SDK 支持启用压缩模式写入日志（默认为使用压缩方式）。
- 提供统一的异步发送、资源控制、自动重试、优雅关闭、感知上报等功能，使上报日志功能更完善。

SDK 列表

下表列举了日志服务不同语言 SDK 的代码仓库：

SDK 语言	代码仓库	上传日志实践
Python	tencentcloud-cls-sdk-python	Python SDK 上传日志
Java	tencentcloud-cls-sdk-java	Java SDK 上传日志
C++	tencentcloud-cls-sdk-c++	C++ SDK 上传日志
C	tencentcloud-cls-sdk-c	C SDK 上传日志
Go	tencentcloud-cls-sdk-go	GO SDK 上传日志
NodeJS	tencentcloud-cls-sdk-js	NodeJS SDK 上传日志
HarmonyOS NEXT	tencntcloud-cls-sdk-ohos	HarmonyOS SDK 上传日志
Android	tencentcloud-cls-sdk-android	Android SDK 上传日志
iOS	tencentcloud-cls-sdk-ios	IOS SDK 上传日志
PHP	tencentcloud-cls-sdk-php	PHP SDK 上传日志
Flutter	tencentcloud_cls_sdk_dart	Flutter SDK 上传日志
浏览器 JavaScript	tencentcloud-cls-sdk-js-web	浏览器 JavaScript SDK上传日志
小程序 JavaScript	tencentcloud-cls-sdk-js-mini	小程序 JavaScript SDK 上传日志
.net sdk	tencentcloud-cls-sdk-dotnet	–

使用 API 上传日志

最近更新时间：2024-10-14 16:10:41

功能描述

本接口用于将日志写入到指定的日志主题。系统根据当前日志主题下所有可读写的分区，遵循负载均衡原则自动分配写入的目标分区。

示例

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf

<LogGroupList 的 PB 格式打包内容>
```

内外网域名

日志服务请求域名区分内外网形式：

- 内网域名格式形如 `${region}.cls.tencentyun.com` ，内网域名仅对同地域访问生效，即云服务器或云产品通过内网域名访问相同地域的日志服务。
- 外网域名格式形如 `${region}.cls.tencentcs.com` ，访问源端接入 Internet 网后，正常情况均能访问日志服务外网域名。

`region` 字段是日志服务区域简称，例如北京区域填写 `ap-beijing` ，完整区域列表格式参见 地域列表 。

```
ap-beijing - 北京
ap-shanghai - 上海
ap-guangzhou - 广州
ap-chengdu - 成都
...
```

此外，日志服务还为用户提供以下两种不同的日志上传模式：

压缩上传模式

使用压缩上传模式时，用户日志将在采集端先以 lz4 格式压缩，之后再将压缩完成后的日志数据上传到日志存储端。通过使用压缩上传模式，用户可以减少日志上传流量（日志写流量），从而节省费用。

示例

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf
x-cls-compress-type:lz4

<LogGroupList 的 PB 格式压缩包内容>
```

非压缩上传模式

使用非压缩上传模式时，用户日志将以原始大小上传，相比于压缩上传模式，非压缩模式将会产生更高的日志写流量费用。

示例

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
```

```
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf

<LogGroupList 的 PB 格式打包内容>
```

签名方法

> **注意：**
> 仅支持 v1 签名方法。

签名方法请参见 签名方法 。

请求

请求行

```
POST /structuredlog
```

请求参数

字段名	类型	位置	是否必选	含义
topic_id	string	query	是	日志主题 ID ，表示数据上传的目标日志主题，可在 日志主题列表 查看日志主题 ID
logGroupList	message	pb	是	logGroup 列表，封装好的日志组列表内容，建议 logGroup 数量不要超过5个

LogGroup 说明：

字段名	是否必选	含义
logs	是	日志数组，表示由多个 Log 组成的集合，一个 Log 表示一条日志，一个 LogGroup 中 Log 个数不能超过10000
contextFlow	否	保持上下文的 UID，该字段目前暂无效用
filename	否	日志文件名
source	否	日志来源，一般使用机器 IP 作为标识
logTags	否	日志的标签列表

Log 说明：

字段名	是否必选	含义
time	是	日志时间（Unix 格式时间戳），支持秒、毫秒，建议采用毫秒
contents	否	key-value 格式的日志内容，表示一条日志里的多个 key-value 组合

Content 说明：

字段名	是否必选	含义
key	是	单条日志里某个字段组的 key 值，不能以 _ 开头
value	是	单条日志某个字段组的 value 值，单条日志 value 不能超过1MB，LogGroup 中所有 value 总和不能超过5MB > **注意：** value 赋值必须为 string 类型。如果期望在日志检索分析时， value 被当作其他类型，例如 long 或者 double，可在为该字段配置键值索引时指定 value 类， 详情请参见 索引配置 。

LogTag 说明：

字段名	是否必选	含义
key	是	自定义的标签 key
value	是	自定义的标签 key 对应的 value 值

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码 。

PB 编译示例

> **说明：**
> 目前 protoc 官方支持 Java、C++、Python 等语言的编译，详情请参见 protoc 。
> 本示例将说明如何使用官方 protoc 编译工具将 PB 描述文件 编译生成为 C++ 语言可调用的上传日志接口。

1. 安装 Protocol Buffer

1. 下载 Protocol Buffer 。示例版本为 protobuf 2.6.1，环境为 Centos 7.3 系统。

```
wget https://main.qcloudimg.com/raw/d7810aaf8b3073fbbc9d4049c21532aa/protobuf-2.6.1.tar.gz
```

2. 解压 `protobuf-2.6.1.tar.gz` 压缩包至 `/usr/local` 目录并进入该目录，执行以下命令：

```
tar -zxvf protobuf-2.6.1.tar.gz -C /usr/local/ && cd /usr/local/protobuf-2.6.1
```

3. 开始编译和安装，配置环境变量，执行命令如下：

```
./configure
make && make install
export PATH=$PATH:/usr/local/protobuf-2.6.1/bin
```

4. 编译成功后，您可以使用以下命令查看版本， 如果返回 liprotoc x.x.x则表示编译成功：

```
protoc --version
```

2. 创建 PB 描述文件

PB 描述文件是通信双方约定的数据交换格式，上传日志时须将规定的协议格式编译成对应语言版本的调用接口，然后添加到工程代码里，详情请参见 protoc 。

以日志服务所规定的 PB 数据格式内容为准， 在本地创建 PB 消息描述文件 cls.proto。

> **注意：**
> PB 描述文件内容不可更改，且文件名须以 `.proto` 结尾。

cls.proto 内容（PB 描述文件）如下：

```
package cls;

message Log
{
    message Content
    {
        required string key   = 1; // 每组字段的 key
        required string value = 2; // 每组字段的 value
    }
    required int64   time     = 1; // 时间戳，UNIX时间格式
    repeated Content contents = 2; // 一条日志里的多个kv组合
}

message LogTag
{
    required string key       = 1;
    required string value     = 2;
}

message LogGroup
{
    repeated Log    logs        = 1; // 多条日志合成的日志数组
    optional string contextFlow = 2; // 目前暂无效用
    optional string filename    = 3; // 日志文件名
    optional string source      = 4; // 日志来源，一般使用机器IP
    repeated LogTag logTags     = 5;
}

message LogGroupList
{
    repeated LogGroup logGroupList = 1; // 日志组列表
}
```

3. 编译生成

此例中，使用 proto 编译器生成 C++ 语言的文件，在 cls.proto 文件的同一目录下，执行如下编译命令：

```
protoc --cpp_out=./ ./cls.proto
```

> **说明：**
> - `--cpp_out=./` ：表示编译成 cpp 格式并输出当前目录下。
> - `./cls.proto`：表示位于当前目录下的 cls.proto 描述文件。

编译成功后，会输出对应语言的代码文件。此例会生成 cls.pb.h 头文件和 cls.pb.cc 代码实现文件，如下图所示：

```
[root@VM_0_8_centos protobuf-2.6.1]# ls
cls.pb.cc cls.pb.h cls.proto
```

4. 调用

将生成的 cls.pb.h 头文件引入代码中，调用接口进行数据格式封装。

IOS/Android 移动端网络探测接入指南

最近更新时间：2024-04-24 17:25:02

CLS 日志服务移动端 SDK 支持收集客户端本地网络及访问服务端网络链路信息进行上报。

使用场景

您可根据以下不同需求场景触发移动端 SDK 上报服务端网络链路信息：

- 客户端本地主动触发（用户无法登录时单击**网络探测**进行数据收集上报，同时提供 uuid 进行保障进行 troubleshooting）
- 游戏内事件触发（如登录时、开始战斗时等进行收集上报，用于统计大盘数据，也可用于玩家排障时的数据支撑）
- 服务端触发客户端上报（使用场景与客户端主动触发类似，只是时效性较差）

网络探测方式

主要用以下的几种方式进行网络探测：

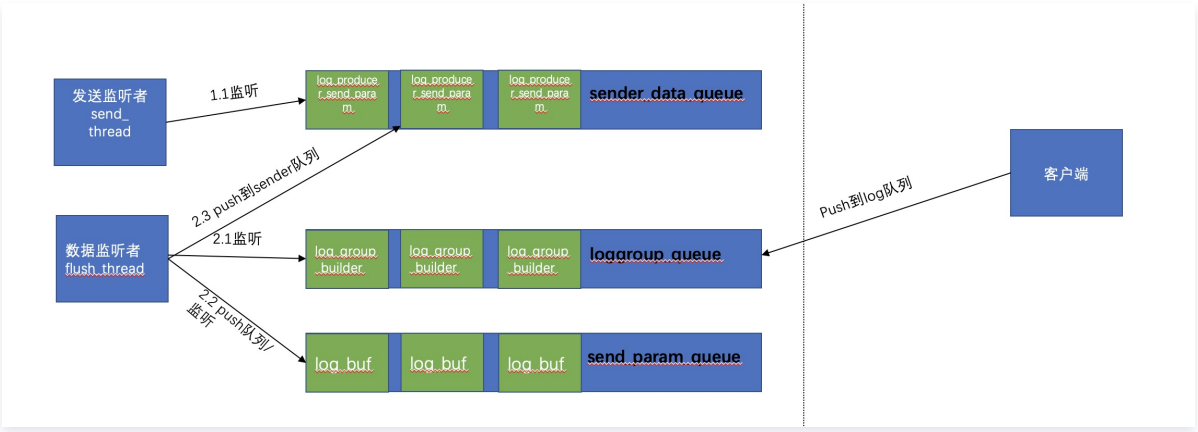
- Ping（ICMP）
- TCP/UDP ping（发送相应协议）
- Traceroute（探测客户端到服务端链路以及链路上的丢包）

IOS/Android SDK 上报网络链路日志

功能特点

- 异步写入，客户端线程无阻塞。
- 聚合&压缩

  上传支持按超时时间、日志数、日志 size 聚合数据发送。支持 lz4 压缩。
- 缓存
  - 支持缓存上限可设置
  - 超过上限后日志写入失败
- 核心上报架构



使用方法

- IOS 移动端 SDK 上报网络链路日志 (含 Demo 及使用方法)

  GitHub - TencentCloud/tencentcloud-cls-sdk-ios: 腾讯云 CLS IOS SDK
- Android 移动端 SDK 上报网络链路日志 (含 Demo 及使用方法)

  GitHub - TencentCloud/tencentcloud-cls-sdk-android: 腾讯云 CLS Android SDK

日志字段说明

公共字段

字段	含义	示例
access	本次探测使用的网络	Wi-Fi
access_subtype	同时接入多个网络时有，本次探测使用的其他网络	● Wi-Fi ● Android：3G/4G/5G

		● iOS：cellula
app_version	App 版本号	1.0.0
device_model	设备模式	-
me	移动用户身份标识	-
resolution	屏幕分辨率	2476*1440
local_time	本地时间	2023-02-01 20:58:00:332
root	是否为 root 用户	false
app_id	App 包名	-
brand	设备厂商信息	google
os	操作系统	Android
utdid	设备标识	-
os_version	操作系统版本	13
reserve6	探测具体内容	-
reserves	探测协议	● ping ● tcpping ● Traceroute
app_name	App 名称	test
imei	移动设备标识	-
local_timestamp	本地时间戳	1675256280332

PING 探测

字段	含义	示例
method	探测方式	ping
host_ip	通过域名解析到的 IP 地址	192.0.65.112
host	域名	www.tencentcloud.com
max	最大延迟，单位：ms	100.11
min	最小延迟，单位：ms	0.00
avg	平均延迟，单位：ms	74.51
stddev	延迟标准差	20.00
loss	PING 包丢失次数	1
count	探测次数，每次发一个 PING 包	10
size	PING 包字节数	64
responseNum	PING 包回应个数	9
interval	PING 包时间间隔，单位：ms	200
timestamp	本地时间戳。	1675256419

TCPPING 探测

字段	含义	示例
method	探测方式	TCPPING
host_ip	通过域名解析到的 IP 地址	192.0.65.112
host	域名	www.tencentcloud.com
max	最大延迟，单位：ms	100.11
min	最小延迟，单位：ms	0.00
avg	平均延迟，单位：ms	74.51
stddev	延迟标准差	20.00
loss	PING 包丢失次数	1
count	探测次数，每次发一个 PING 包	10
size	PING 包字节数	64
sum	本次探测总时间，单位：ms	219.66
port	TCP 端口	88
timestamp	本地时间戳	1675256419

Traceroute 探测

字段	含义	示例
method	探测方式	TRACEROUTE
host_ip	通过域名解析到的 IP 地址	192.0.65.112
host	域名	www.tencentcloud.com
command_status	探测请求状态	success
timestamp	本地时间戳	1675256419
traceroute_node_results	TRACEROUTE 探测节点返回结果	列表内容见下文详细字段描述

traceroute_node_results

字段	含义	示例
targetIp	某一跳的 IP 地址	43.152.65.112
hop	某一跳的跳数，探测发起端为第0跳，越接近目的端跳数越大	1
avg_delay	平均延迟	102
loss	探测丢包次数	33
is_final_route	是否是最终路径	true
single_node_list	某单个节点的返回结果	列表见下文详细字段

single_node_results

字段	含义	示例
targetIp	某一跳的 IP 地址	43.152.65.112

版权所有：腾讯云计算（北京）有限责任公司

hop	某一跳的跳数，探测发起端为第0跳，越接近目的端跳数越大	1
delay	探测延迟	102
is_final_route	是否是最终路径	true
status	本次探测请求状态	CMD_STATUS_FAILED/CMD_STATUS_SUCCESSFUL

FluentBit 日志上传

最近更新时间：2024-10-10 17:23:31

概述

CLS 支持通过以下两种方式将 FluentBit 中的日志上传至 CLS。

- Fluent-bit Go 插件
- Kafka 协议上传

操作步骤

FluentBit Go 插件

> **注意：**
> 该插件依赖 Go 环境。

1. 前往 fluent-bit-go cls 日志服务插件，获取 FluentBit CLS 日志服务插件。
2. 执行以下命令 `Build` FluentBit CLS 日志服务插件。

```
go build -buildmode=c-shared -o fluent-bit-go.so
```

3. 执行以下命令 `Run` FluentBit CLS 日志服务插件。

```
fluent-bit -c example/fluent.conf -e fluent-bit-go.so
```

4. 配置 FluentBit 的配置文件 `example/fluent.conf` 。

```
[OUTPUT]
        Name              fluent-bit-go-cls
        Match             *
        # TODO: 请配置以下参数
        TopicID           YOUR_TOPIC_ID
        CLSEndPoint       YOUR_ENDPOINT
        AccessKeyID       YOUR_PROJECT_SK
        AccessKeySecret   YOUR_PROJECT_AK
```

参数描述

参数名	描述
TopicID	目标上传的日志主题ID
CLSEndPoint	目标上传的日志主题所在地域的域名。 请参见 可用地域 例子： • 广州内网域名：ap-guangzhou.cls.tencentyun.com • 广州公网域名：ap-guangzhou.cls.tencentcs.com
AccessKeyID	云 API 密钥 的一部分，SecretId 用于标识 API 调用者身份
AccessKeySecret	云 API 密钥 的一部分，SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥

> **说明：**
> 请确保云 API 密钥关联的账号具有相应的 Fluent-bit 日志上传最小权限 。

Kafka 协议上传 FluentBit 中的日志

请参见 Kafka 协议上传 Fluent-bit 配置 。

Logstash 日志上传

最近更新时间：2024-10-29 09:51:32

概述

CLS 日志服务支持通过以下两种 Logstash 插件将 Logstash 采集的日志上传至 CLS 日志服务：

- Logstash CLS Output 插件
- Logstash Kafka Output 插件

本文将介绍如何通过这两种 Logstash Output 插件将日志上传至 CLS 日志服务。

通过 CLS Output 插件上传

> **注意：**
> 请确保云 API 密钥关联的账号具有相应的 Logstash 日志上传最小权限 。

1. 执行以下命令安装 Logstash CLS Output 插件。

```
logstash-plugin install logstash-output-cls
```

2. 执行以下运行 Logstash CLS Output 插件。

```
logstash -f logstash-sample.conf
```

3. 配置 Logstash 的配置文件 `example/fluent.conf` 。

```
input {
}

## 若需要指定日志中的时间字段作为日志时间戳， 可配置如下时间解析配置
filter {
  date {
    match => ["produce_log_time","yyyy-MM-dd HH:mm:ss.SSS"]
    target => "@timestamp"
  }
}

output {
  cls{
     endpoint  => "[CLS数据接入域名]"
     topic_id  => "[日志主题id]"
     access_key_id     => ""
     access_key_secret => ""
  }
}
```

参数描述

参数名	类型	是否必填	默认值	描述
endpoint	string	是	-	目标上传的日志主题所在地域的域名。请参见 可用地域 。 示例： • 广州内网域名：ap-guangzhou.cls.tencentyun.com • 广州公网域名：ap-guangzhou.cls.tencentcs.com
topic_id	string	是	-	目标上传的日志主题 ID。
source	string	否	本地网卡 IP	日志的来源 IP。
access_key_id	string	是	-	云 API 密钥 的一部分，SecretId 用于标识 API 调用者身份。
access_key_secr	string	是	-	云 API 密钥 的一部分，SecretKey 是用于加密签名字符串和服务器端验

et				证签名字符串的密钥。
max_buffer_items	int	否	4000	日志上传按照批次上传。该参数控制每一批次最多可以有多少条日志。
max_buffer_bytes	int	否	2097152	日志上传按照批次上传。该参数控制每一批次的日志总大小最多为多少。单位为Bytes。
max_buffer_seconds	int	否	3	日志上传按照批次上传。 该参数控制每一批次从创建到可发送的最大逗留时间。
total_size_in_bytes	int	否	104857600	实例能缓存的日志大小上限。 单位为Bytes。
max_send_retry	int	否	10	日志上传失败时， 最大重试次数。
send_retry_interval	int	否	200	日志上传失败时，下一次重试的时间间隔。
to_json	boolean	否	true	是否对采集的日志进行 Json 解析。
time_key	string	否	@timestamp	日志时间戳的来源字段 key 名。

通过 Kafka Output 插件上传

请参见 Kafka 协议上传 Logstash 配置 。

数据导入
导入 COS 数据

最近更新时间：2024-09-06 15:41:11

概述

日志服务（Cloud Log Service，CLS）投递功能打通产品生态上游链路，将腾讯云对象存储（Cloud Object Storage，COS）中的数据导入到日志服务，实现日志数据的查询分析、加工等操作，挖掘日志数据价值，您只需要在日志服务控制台进行简单的配置即可完成数据导入。

前提条件

- 开通日志服务，创建日志集与日志主题，并成功采集到日志数据。
- 开通对象存储，并确保待导入的文件已经上传到 COS Bucket 中，更多信息请参见 上传对象 。
- 设置当前操作账号拥有访问 COS 的权限，即已授权日志服务使用 CLS_QcsRole 角色访问您的 COS 资源。更多信息请参见 投递 COS 权限配置 。

配置流程

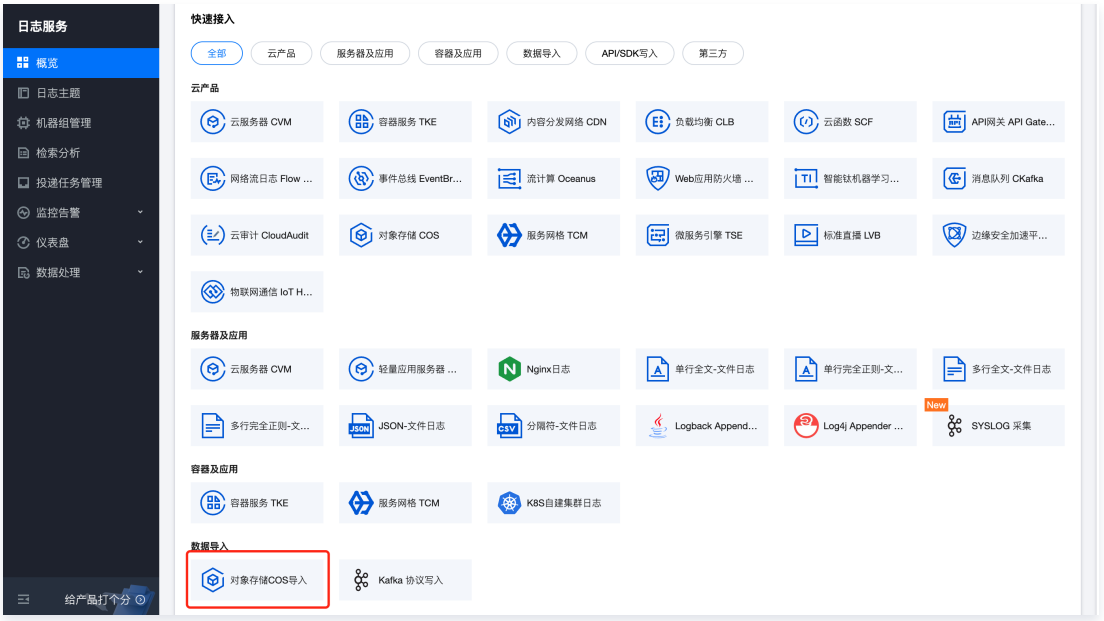
1. **选择日志主题**：可以选择现有的日志主题（topic）或者创建新的日志主题（topic），用于存储从 COS 导入到 CLS 的数据。
2. **导入数据源配置**：需要配置想要导入的 COS 对象路径，以及压缩方式（gzip/lzop/snappy/无压缩）。
3. **解析配置**：需要配置导入文件的解析格式，目前支持单行全文/多行全文/单行正则/多行正则/JSON/分隔符。
4. **索引配置**：需要配置目前 topic 的索引配置，且检索必须开启索引配置。如果选择已存在的日志主题，则新索引配置只对修改后的数据生效。

操作步骤

步骤1：选择日志主题

- 如果您想选择新的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**概览**，进入概览页面。
3. 在快速接入页面中 ，找到数据导入栏，单击**对象存储 COS 数据导入**。



4. 在创建日志主题页面步骤，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步**。

- 如果您想选择现有的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，选择需要投递的日志主题，进入日志主题管理页面。
3. 选择**采集配置**页签，在 **COS 数据导入配置**栏下单击**新增**。

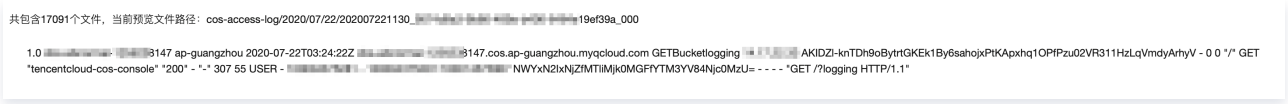


步骤2：数据源配置

1. 在数据源配置页面，依次配置如下信息：

配置项	说明	规则	是否必填
任务名称	配置导入任务的名称。	字母、数字、_和-	是
存储桶地域	配置需要导入的文件所在存储桶的地域位置。如果文件所在地域和导入日志主题所在地域不同，则会因跨地域访问而产生外网费用。	列表选择	是
COS 存储桶	选择需要导入的文件所在存储桶。下拉框会列出当前选择地域下的所有存储桶供您选择。	列表选择	是
文件前缀	输入待导入的 COS 文件所在文件夹的前缀，用于准确定位待导入的文件夹，支持输入文件前缀 csv/，或完整文件路径 csv/object.gz。 **注意：** 不支持采集存储类型为“归档存储”或“深度归档存储”的文件。	输入框	是
压缩模式	选择待导入的 COS 文件的压缩格式，日志服务根据对应格式进行解压并读取数据，支持 gzip/lzop/snappy/bzip2/zip/zstd/deflate/lz4(frame)/不压缩。	列表选择	是

2. 单击**预览**，系统会选择符合配置规则的一条路径数据进行展示，并提示输入的文件前缀下共有多少文件。如下图所示：



3. 确认预览无误后，单击**下一步**。

步骤3：解析配置

1. 在解析配置页面，配置如下主要信息：
   - 提取模式：可选择以下几种提取模式。

单行全文

单行全文日志是指一行日志内容为一条完整的日志。日志服务在采集的时候，将使用换行符 \n 来作为一条日志的结束符。为了统一结构化管理，每条日志都会存在一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条日志原始数据为：

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

采集到日志服务的数据为：

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

单行-完全正则

单行完全正则格式通常用来处理结构化的日志，指将一条完整日志按正则方式提取多个 key-value 的日志解析模式。
假设一条日志原始数据为：

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703
"http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum"
"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0"  0.354 0.354
```

配置的正则表达式为如下：

```
(\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"
([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
```

采集到日志服务的数据为：

```
ody_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

多行全文

多行全文日志是指一条完整的日志数据可能跨占多行（例如 Java stacktrace）。在此情况下，为了能让日志系统明确区分开每条日志，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。
多行全文也会设置一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条多行日志原始数据为：

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
    at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
    at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

首行正则表达式为如下：

```
\d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2},\d{3}\s.+
```

采集到日志服务的数据为：

```
__CONTENT__:2019-12-15 17:13:06,043 [main] ERROR
com.test.logging.FooFactory:\njava.lang.NullPointerException\n    at
com.test.logging.FooFactory.createFoo(FooFactory.java:15)\n    at
com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

多行-完全正则

假设您的一条日志原始数据为：

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

行首正则表达式为：

```
\[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*
```

配置的自定义正则表达式为：

```
\[(\d+-\d+-\w+:\d+:\d+,\d+)\]\s\[(\w+)\]\s(.*)
```

系统根据 `()` 捕获组提取对应的 key-value 后，您可以自定义每组的 key 名称如下所示：

```
time: 2018-10-01T10:30:01,000`
level: INFO`
msg: java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

JSON

假设您的一条 JSON 日志原始数据为：

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34
+0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost":"unix:/tmp/php-
cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","request":"POST
/event/dispatch HTTP/1.1","xff":"-","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0
(Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
```

```
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

分隔符

假设您的一条日志原始数据为：

```
10.20.20.10 - ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/1.1 ::: 127.0.0.1
::: 200 ::: 647 ::: 35 ::: http://127.0.0.1/
```

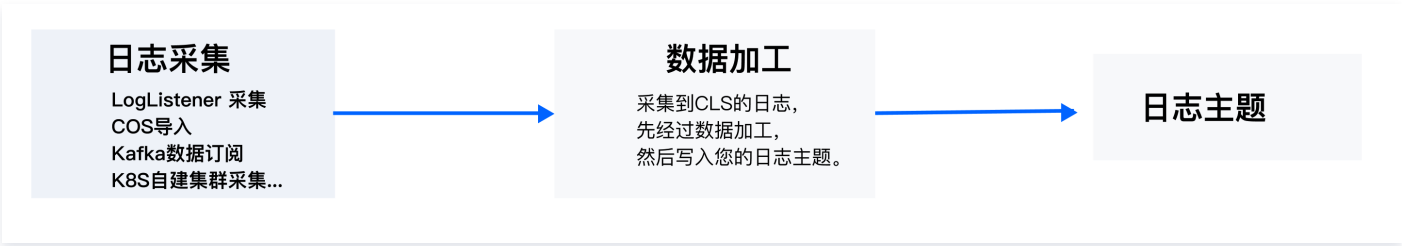
当日志解析的分隔符指定为 `:::`，该条日志会被分割成八个字段，并为这八个字段定义唯一的 key，如下所示：

```
IP: 10.20.20.10 -
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

- **过滤器：**
  - **过滤器可以帮助您根据业务需要添加日志采集过滤规则，为您筛选出有价值的日志数据。过滤规则为 Perl 正则表达式，所创建的过滤规则为命中规则，即匹配上正则表达式的日志才会被采集上报。**
  - **分隔符格式日志需要根据所自定义的键值对来配置过滤规则。例如，样例日志使用分隔符模式解析后，您希望 status 字段为 400 或 500 的所有日志数据被采集，那么 key 处配置 status，过滤规则处配置 400|500。**
- **使用采集时间：开启后可根据采集时间标记每条日志时间，或关闭此项指定某一字段作为日志时间。**

> **说明**
> - **日志时间单位为：秒，若时间格式填写错误日志时间将以采集时间为准。**
> - **日志的时间属性有两种方式来定义：采集时间和原始时间戳。**
> - **采集时间：日志的时间属性由日志服务 CLS 从 COS 导入时间决定。**
> - **原始时间戳：日志的时间属性由原始日志中时间戳决定。**
> - **日志的原始时间戳作为日志时间属性。**
> - **关闭采集时间状态，在时间键和时间格式解析处，填写原始时间戳的时间键以及对应的时间解析格式。时间解析格式详情参见 配置时间格式 。**

如您需要对采集到的 CLS 日志进一步处理，如结构化、脱敏、过滤，然后再写入日志主题，您可点击页面下方的数据加工，添加数据加工，最后配置索引。



说明：

- 数据加工相关操作请参考 创建加工任务 文档中【前置数据加工】TAB 页面；
- 数据加工脚本的编写请参考 数据加工函数总览，或者 加工实战案例；
- 数据加工会产生费用，详情请参见 计费概述。

2. 单击**下一步**。

步骤4：索引配置

1. 在索引配置页面，配置如下信息。配置详情请参考 索引配置。



注意

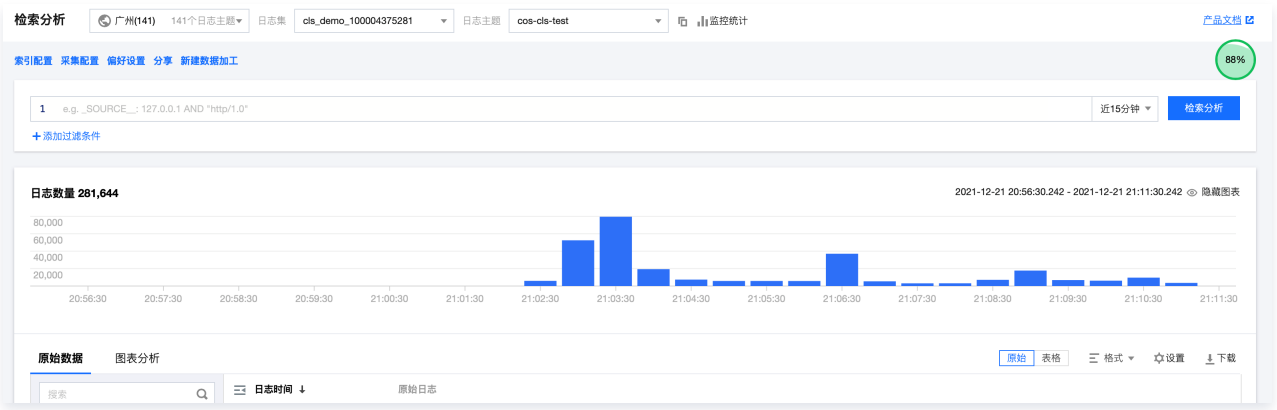
- 检索必须开启索引配置，否则无法检索。
- 索引规则编辑后仅对新写入的日志生效，已有数据不会更新。

2. 单击**提交**，完成导入配置。

相关操作

导入进度查询

1. 当前日志主题下存在 COS 导入任务时，单击**检索分析**，进入**检索分析**页面查看当前导入任务的进度。



2. 在检索分析页面右上角的进度悬浮球显示目前完成的导入进度。单击悬浮球，右侧浮窗会展开导入任务详情。



3. 在任务详情页面，单击**查看详情**，即可跳转采集配置页查询导入的详细配置。

规格限制

COS 数据导入规格限制

采集云产品日志

最近更新时间：2025-06-27 11:06:31

日志服务（Cloud Log Service，CLS）支持采集多个云产品的日志数据，例如：SCF 运行日志、CDN 实时日志、TKE 容器日志、CLB 访问日志等。用户可以使用 CLS 服务进行操作记录查看，运营数据统计分析，运行状况监控告警等多种日志数据处理功能。
当前 CLS 支持采集的云产品日志如下。

弹性计算

云产品名称	日志类型	采集指引	预置仪表盘
云服务器 CVM	业务日志	CLS 控制台配置，配置指引	-
容器服务 TKE	事件日志	CLS 控制台配置，配置指引	• 事件日志总览仪表盘，查看演示 Demo • 事件日志异常事件聚合检索仪表盘，查看演示 Demo
	审计日志	CLS 控制台配置，配置指引	• 审计日志总览仪表盘，查看演示 Demo • 审计日志节点操作概览仪表盘，查看演示 Demo • 审计日志 K8S 对象操作概览仪表盘，查看演示 Demo • 审计日志聚合检索仪表盘，查看演示 Demo
	业务日志	CLS 控制台配置，配置指引	-
云函数 SCF	• 用户日志 • 平台日志	SCF 控制台配置，配置指引	SCF 函数调用日志分析仪表盘
服务网格 TCM	访问日志	TCM 控制台配置，配置指引	TCM 访问日志分析仪表盘

存储服务

云产品名称	日志类型	采集指引	预置仪表盘
对象存储 COS	访问日志	COS 控制台配置，配置指引	COS 访问日志分析仪表盘，查看演示 Demo
	客户端 SDK 的运行日志	COS 控制台配置， 配置指引	COS SDK 访问日志分析仪表盘
数据加速器 GooseFS	运行日志	GooseFS 控制台配置，配置指引	-
CFS 文件存储	审计日志	CFS 控制台配置，配置指引	CFS 审计日志仪表盘

数据库

云产品名称	日志类型	采集指引	预置仪表盘
TDSQL-C MySQL 版	慢日志	TDSQL-C MySQL 控制台配置，配置指引	-
	Binlog	CLS 控制台配置，配置指引	-
	审计日志	TDSQL-C MySQL 控制台配置，配置指引	-
云数据库 MySQL	• 慢日志 • 错误日志	云数据库 MySQL 控制台配置，配置指引	云数据库 MySQL 慢查询分析仪表盘
	Binlog	CLS 控制台配置，配置指引	-
	审计日志	云数据库 MySQL 控制台配置，配置指引	-
云数据库 PostgreSQL	• 错误日志 • 慢日志	云数据库 PostgreSQL 控制台配置，配置指引	-

版权所有：腾讯云计算（北京）有限责任公司

云数据库 MongoDB	• 慢日志 • 错误日志 • 操作日志	云数据库 MongoDB 控制台配置，配置指引	–

安全

云产品名称	日志类型	采集指引	预置仪表盘
Web 应用防火墙 WAF	访问日志	WAF 控制台配置，配置指引	访问流量分析仪表盘，查看演示 Demo
	攻击日志	WAF 控制台配置，配置指引	攻击日志分析仪表盘，查看演示 Demo
操作审计 CloudAudit	操作审计日志	CloudAudit 控制台配置，配置指引	审计事件分析仪表盘，查看演示 Demo
容器安全服务 TCSS	• 容器 Bash 日志 • 容器启动审计日志 • K8s API 审计日志	TCSS 控制台配置，配置指引	–
云安全中心	根据所选的日志来源不同则日志类型不同	云安全中心控制台配置，配置指引	–
云防火墙 CFW	操作日志	云防火墙 CFW 控制台配置，配置指引	–
主机安全	• 主机资产日志 • 客户端上报日志 • 告警日志	主机安全控制台配置，配置指引	–
数据安全审计	审计日志	数据安全审计控制台配置，配置指引	–
堡垒机	审计日志	堡垒机审计日志控制台配置，配置指引	–
DDoS 防护	• 慢日志 • 错误日志	DDoS 日志控制台配置，配置指引	–

网络

云产品名称	日志类型	采集指引	预置仪表盘
负载均衡 CLB	访问日志	CLB 控制台配置，配置指引	访问日志仪表盘，查看演示 Demo
	健康日志	CLB 控制台配置，配置指引	–
网络流日志 FlowLog	• NAT 网关流日志 • 弹性网卡流日志 • 云联网跨地域流日志	FL 控制台配置，配置指引	• CCN 流日志高级分析仪表盘，查看演示 Demo • ENI 流日志高级分析仪表盘，查看演示 Demo
API 网关 APIGW	访问日志	API 网关 控制台配置，配置指引	接口质量分析仪表盘，查看演示 Demo
内容分发网络 CDN	访问日志	CDN 控制台配置，配置指引	• 质量监控分析仪表盘，查看演示 Demo • 用户行为分析仪表盘，查看演示 Demo
全站加速网络 ECDN	访问日志	CDN 控制台配置，配置指引	–
边缘安全加速平台 EdgeOne	访问日志	EdgeOne 控制台配置，配置指引	• 访问日志分析仪表盘，查看演示 Demo • 防盗刷流量分析仪表盘，查看演示 Demo

消息队列

云产品名称	日志类型	采集指引	预置仪表盘
消息队列 CKafka	Kafka 消息	• CKafka 控制台配置，配置指引	–

版权所有：腾讯云计算（北京）有限责任公司

		• CLS 控制台配置，配置指引	

物联网

云产品名称	日志类型	采集指引	预置仪表盘
物联网通信 IoT Hub	• 行为日志 • 内容日志 • 设备日志	IoT Hub 控制台配置，配置指引	-

大数据

云产品名称	日志类型	采集指引	预置仪表盘
流计算 Oceanus	作业运行日志	Oceanus 控制台配置，配置指引	-
腾讯云数据仓库 TCHouse-C	• 运行日志 • 错误日志	TCHouse-C 服务控制台配置，配置指引	-
腾讯云数据仓库 TCHouse-D	• FE 日志 • BE 日志 • Broker 日志	TCHouse-D 服务控制台配置，配置指引	-
弹性 MapReduce	• 运行日志 • 审计日志	弹性 MapReduce 控制台配置，配置指引	-

AI 平台产品

云产品名称	日志类型	采集指引	预置仪表盘
腾讯云 TI 平台 TI-ONE	• 任务日志 • Notebook 日志 • 训练日志	TI-ONE 控制台配置，配置指引	-

视频服务

云产品名称	日志类型	采集指引	预置仪表盘
云点播 VOD	访问日志	VOD 控制台配置，配置指引	• 质量监控分析仪表盘 • 用户行为分析仪表盘
云直播	• 播放日志 • 推流日志	云直播控制台配置，配置指引	• 播放日志分析仪表盘，查看演示 Demo • 推流日志分析仪表盘，查看演示 Demo

域名管理

云产品名称	日志类型	采集指引	预置仪表盘
Private DNS	访问日志	Private DNS 控制台配置，配置指引	-

微服务工具与平台

云产品名称	日志类型	采集指引	预置仪表盘
弹性微服务 TEM	业务日志	TEM 控制台配置，配置指引	-
北极星网络 TSE	治理中心运行日志	TSE 控制台配置，配置指引	-

版权所有：腾讯云计算（北京）有限责任公司

云原生网关 TSE	• 访问日志 • 错误日志	TSE 控制台配置，配置指引	云原生网关访问分析仪表盘，查看演示 Demo
TSF 微服务平台	业务日志	TSF 微服务平台，配置指引	-

云托管

云产品名称	日志类型	采集指引	预置仪表盘
云开发 CloudBase	业务日志	CloudBase 控制台配置，配置指引	-

全球加速日志上传

最近更新时间：2025-03-20 16:26:21

应用场景

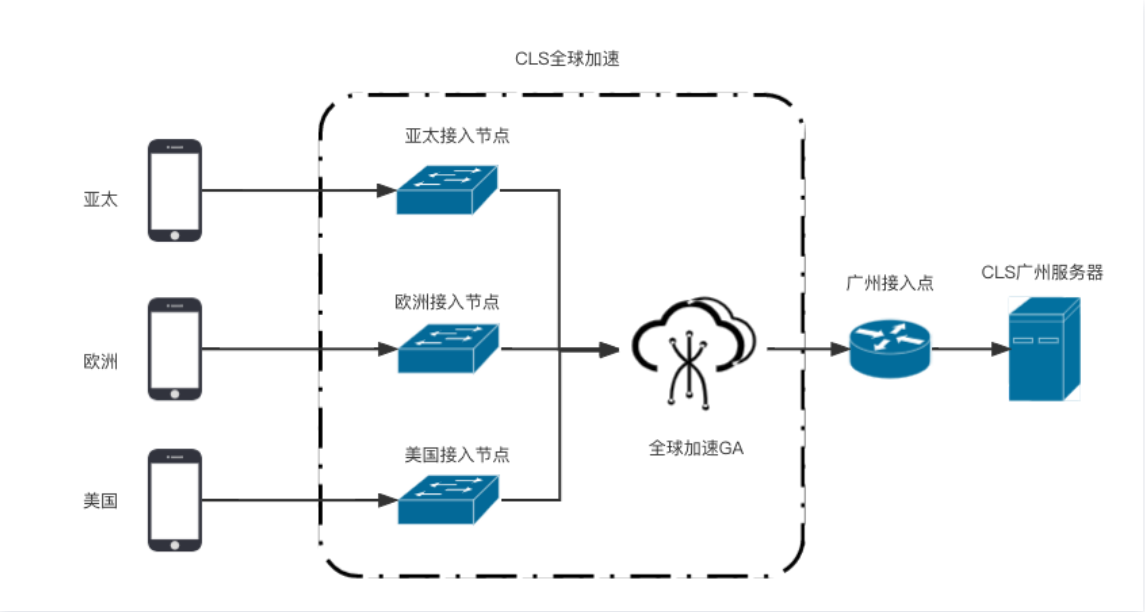
腾讯云 CLS 全球加速解决方案用于解决在网络环境比较差的情况下，用户能稳定上传日志的问题，该方案能够加速通过任何网络环境上传日志到 CLS，对于跨国公司和移动互联网企业来说，此解决方案能够为网站、移动应用和应用服务提供安全可靠的方式快速连接，实现网络服务全球范围就近接入和跨地域部署。

业务痛点

- 跨地公网域上传日志网络耗时高，丢包严重。
- 移动 App 客户支持差，无法提供高效的日志上传服务。

全球加速方案

全球跨地域接入方案如下图所示（境外 -> 广州）



工作原理：就近接入 -> 腾讯云内网 -> 目标地域接入节点 - 目标地域 CLS 服务器

方案亮点

- 加速延迟敏感型客户
- 提供企业级稳定网络接入
- 提供安全的网络环境
- 接入方便快捷，易于维护

接入方式

该功能可能涉及跨境数据传输， 需 提交工单 申请使用该功能。 工单中，请描述以下信息：

- 日志源地域
- 日志目标上传地域
- 预估最大带宽需求
- 账号信息，即 APPID

待工单审批通过后，我们将提供一个您账号专用的日志上传加速域名。通过该域名，您可以通过以下方式加速跨地域日志上传。

LogListener 采集日志上传加速

首次安装 LogListener

您可参见 LogListener 安装指南 下载并安装 LogListener。在初始化 LogListener 步骤，以安装路径 `/usr/local/` 为例，在 `/usr/local/loglistener/tools` 路径下，以 root 权限执行 LogListener 初始化命令，初始化命令如下：

```
./loglistener.sh init -secretid AKID****************************** -secretkey
whHwQfjdLnzzCE1jIf09xxxxxxxxxxx -domain asccelerate-xxxxx-ap-xxxxx -IP xxx.xxx.xxx.xxx
```

说明：
初始化命令中 **-secretid**、**-secretkey**、**-domain** 为必填的参数，更多其他参数请见如下 参数说明 。

参数说明

参数名	是否必填	类型描述
secretid	是	云 API 密钥 的一部分，SecretId 用于标识 API 调用者身份。
secretkey	是	云 API 密钥 的一部分，SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥。
domain	是	填写日志上传加速域名。
IP	否	机器的 IP 标识。若不填写，loglistener 会自动获取本机的 IP 地址。
label	否	机器标识，填写后该机器将与同样带有所填机器标识的机器组关联。多个标识按逗号分隔。若配置了机器标识， 该机器将只能通过机器标识与机器组关联，无法通过 IP 进行关联；若不配置，机器组只能通过 IP 关联该机器。

为已有 LogListener 开启全球加速

1. 登录您的机器。
2. 以 LogListener 安装路径 `/usr/local/` 为例，在 `/usr/local/loglistener/etc` 路径下，执行以下命令，打开 LogListener 配置文件 loglistener.conf。

```
vim loglistener.conf
```

3. 找到 proxy-host 配置项，将该配置的值修改为您的全球加速域名。

```
<root>
    proxy_host = ap-chongqing.cls.tencentcs.com
    proxy_port = 80
    secret_id = [redacted]
    secret_key = [redacted]
    # This switch use to enable/disable secret_id/secret_key store encrypted, default false. (true/false)
    encryption =
    group_ip = 10.0.0.11
    # define Loglistener by labels, separated by ,
    group_label =
    instance_id = loglistener-a27458ce-7a2a-747e-9b8e-35811ea25931
    pos_file = data/pos.dat
    checkpoint_cache_file = data/checkpoint_cache
    checkpoint_window_size = 1024
    # Max file break points. Warn: modify this value, history breaskpoints will lost.
    max_file_breakpoints = 8192
    mmap_version_file = data/mmap_version_file.dat
    file_cache = 0
    # The concurrency of the requests.
    max_connection = 20
    # Max memory loglistener would use.
    max_mem = 2097152000
    # Max bytes send per second, 0 for unlimited.
    max_send_rate = 0
    # Max cpu usage limit with single core, set to 50 means 50% usage limit, 0 for unlimited.
    cpu_usage_thres = 0
```

4. 执行以下命令，重启 LogListener。

```
systemctl restart loglistenerd
```

API/SDK 上传加速

仅需 API/SDK 的接口中 host 的入参替换为您的加速域名， 即可实现 API/SDK 加速上传。

计费规则

使用全球加速上传日志将按照日志压缩后产生的写流量进行计费。计费详情请参考 产品定价 。

注意：

- 使用全球加速功能才会产生该流量，普通外网访问不产生。
- 使用全球加速功能时不再重复产生日志写流量。

版权所有：腾讯云计算（北京）有限责任公司

从 Elasticsearch 导入数据

最近更新时间：2025-07-04 18:56:22

本文将介绍如何将Elasticsearch 中的数据导入至日志服务，包括历史数据的一次性导入和新增数据的持续导入，以便实现后续的查询分析、投递消费等操作。

前提条件

- 已有可用的自建 ES集群或腾讯云 ES 集群。
- ES需要为 6.x以上版本。
- 已创建日志集和日志主题。具体操作请参见 日志集 和 日志主题 。

操作步骤

步骤1：选择日志主题

创建新的日志主题

如果您想选择新的日志主题，可执行如下操作：

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**概览**，进入概览页面。
3. 在**快速接入** > 数据导入中，找到并单击 **ES 数据导入**，进入采集配置流程。



4. 在**创建日志主题**页面，根据实际需求，输入日志主题名称，配置日志保存时间等信息，单击**下一步**。



使用现有的日志主题

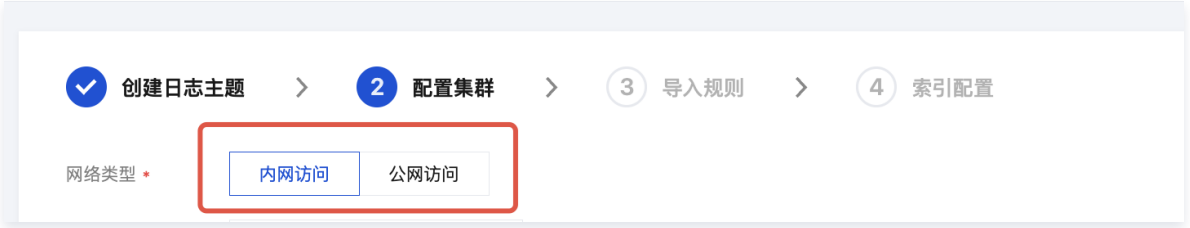
如果您想选择现有的日志主题，可执行如下操作：

1. **登录** 日志服务控制台 **。**
2. **在左侧导航栏中，选择日志主题，选择需要投递的日志主题，单击日志主题名称，进入日志主题管理页面。**
3. **选择采集配置页签，在 ES 数据导入 栏单击新增。**



步骤2：配置 ES 集群基本信息

1. **访问方式：支持选择通过内网地址或公网地址访问您的 ES 集群。**



2. **基于选择的访问方式，参考如下说明配置对应参数：**

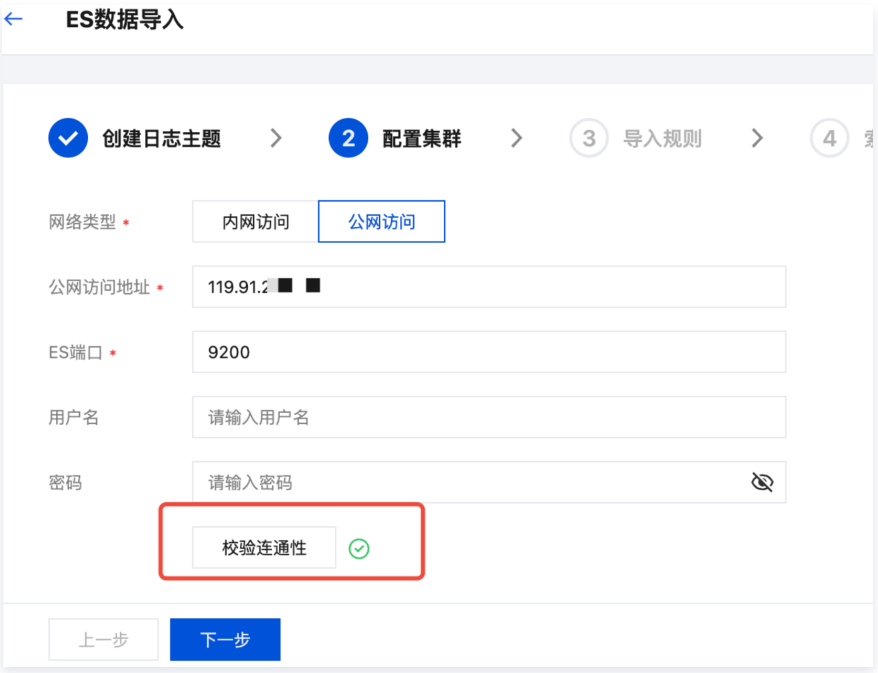
公网访问

参数	是否必填	说明
公网访问地址	是	Elasticsearch 集群的服务地址，填写 IP 或者域名
ES 端口	是	Elasticsearch 集群访问端口，一般为 9200
用户名	否	Elasticsearch 用户名。 仅在 Elasticsearch 集群开启用户认证时需要设置。
密码	否	Elasticsearch 用户密码。 仅在 Elasticsearch 集群开启用户认证时需要设置。

内网访问

参数	是否必填	说明
网络服务类型	是	若访问方式为通过**内网地址**访问，需指定目标 ES 集群的网络服务类型： • CVM • CLB
所属网络	是	当网络服务类型选择为 CVM 或 CLB 时，需选择 CVM 或 CLB 所在的 VPC 实例。
内网访问地址	是	Elasticsearch 集群的服务地址，填写 IP 或者域名。
ES端口	是	Elasticsearch 集群访问端口，一般为 9200。
用户名	否	Elasticsearch 用户名。 仅在 Elasticsearch 集群开启用户认证时需要设置。
密码	否	Elasticsearch 用户密码。 仅在 Elasticsearch 集群开启用户认证时需要设置。

3. 完成集群配置后，可点击校验连通性。若提示成功则表示能够访问 ES 集群。



步骤3：配置 ES 集群导入规则

1. 配置导入规则。



具体配置内容见下方的表格。

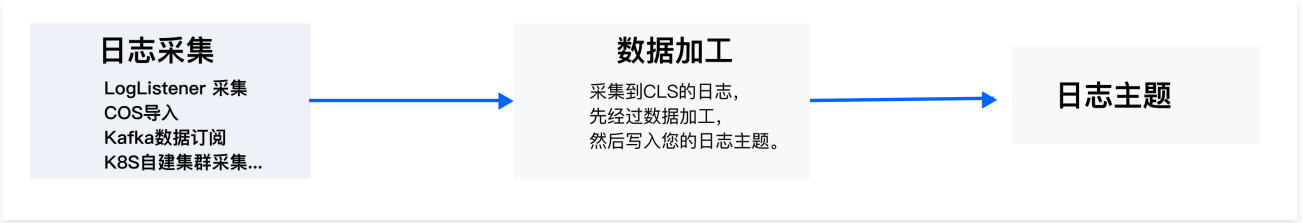
参数	说明
导入规则	导入配置的名称。
索引列表	待导入的索引，多个索引之间使用英文逗号（,）隔开，例如 index1,index2,index3.... 最大支持200个索引。
ES 查询语句	用于过滤数据的查询语句，符合查询条件的数据才会导入到日志服务中。指定为 * 或置空表示导入时不过滤数据。 查询语句必须符合 Elasticsearch 的 query_string 格式，详细信息请参考 Query string query 。
导入模式	支持选择导入历史数据和导入新增数据 • 导入历史数据：数据导入完成后，任务就会完成。 • 导入新增数据：导入任务会一直运行。 如果选择**导入新增数据**，必须指定时间字段。
日志时间来源	支持选择**日志采集时间**和**指定日志字段**。 • 日志采集时间：日志导入到 CLS 的时间作为日志的时间戳 • 指定日志字段：指定日志中代表时间的字段名，该字段的值将会用作日志的时间戳。

	⚠ **注意：** 使用采集时间作为时间字段场景下，ES集群需开启对_id字段的排序a
日志时间字段	日志时间来源为**指定日志字段**时才需要填写该内容，指定日志中代表时间的字段名，该字段的值将会用作日志的时间戳。 ⚠ **注意：** 指定的时间字段类型需要为 keyword 类型，指定为 text/nested/object/binary 不支持排序会导致数据导入。
时间解析格式	确认了日志中的时间字段，需进一步指定时间格式，用于解析时间字段的值。详情请参见 配置时间格式 。
时间字段时区	确认了日志中的时间字段与格式，需选择以下两种不同的时区标准： • UTC（协调世界时） • GMT（格林威治标准时间）
导入时间范围	指定需导入的日志的时间范围。只有设置了时间字段后，此配置才有效。
起始时间	只有当导入模式为导入新增数据，才会有该选项。指定导入数据的起始时间
数据最大延迟	用于指定数据产生到写入 ES 的最大延迟时间。 默认600s，最大3600s。 • 仅导入模式为自动导入新增数据后，此配置才有效。 • 如果设置的值比真实延迟小，将导致有些数据无法从 ES 导入到日志服务。
检查新数据周期	检查 ES 中新增数据的周期，默认值为300s，最小值60s。

2. 配置好内容后，单击**预览**可查看基于当前配置的数据导入结果。如果预览到的数据不符合预期，请修改配置后重试。



如您需要对采集到的 CLS 日志进一步处理，如结构化、脱敏、过滤，然后再写入日志主题，您可单击页面下方的**数据加工**，添加数据加工，最后配置索引。



说明：

- 数据加工相关操作请参见 创建加工任务 文档中的 前置数据加工 TAB 页面。
- 数据加工脚本的编写请参见 数据加工函数总览，或者 加工实战案例 。
- 数据加工会产生费用，详情请参见 计费概述 。

步骤4：检查索引配置

在**索引配置**页面，设置如下信息，配置详情请参考 索引配置 。



注意：
检索必须开启索引配置，否则无法检索。

步骤5：查看 ES 导入任务

完成 ES 导入任务后，您可在日志主题详情页的**采集配置**页签中找到所有已创建的 ES 导入任务。



步骤6：检索分析日志

完成 ES 导入任务后，即可开始使用日志的检索与分析，以及更高阶的仪表盘告警等功能了。

- 检索分析
- 监控告警
- 仪表盘
- 数据加工

日志字段说明

元数据字段

字段	描述
__TAG__.es_url	日志来源的 ES 集群的 URL 地址。
__TAG__.es_index	日志来源的索引信息。

其中日志所属的 ES 文档 ID（_id 字段），不作为元数据字段，作为日志字段在日志中呈现。

规格与限制

限制	描述
单条日志大小	支持导入的单条日志最大为1M。超过限制的部分会丢弃。
导入任务数量	1个主题支持最多100个 ES 导入任务。
导入索引数量	1个任务最大支持200个 ES 索引的导入。

版权所有：腾讯云计算（北京）有限责任公司

指标采集
指标上报

最近更新时间：2024-08-15 14:35:22

指标主题支持 Prometheus Remote Write 协议，可使用兼容该协议的各类采集器采集指标并上报至指标主题，例如 vmagent 及 telegraf。通过这些采集器，您可以将各种类型的监控指标上报至指标主题，例如：

- 服务器监控指标，例如 CPU、内存、磁盘使用率等。
- 容器集群监控指标，例如 Pods 内存使用量、apiserver 请求延迟等。
- 中间件监控指标，例如 MySQL、Kafka、Nginx 等中间件的性能指标。
- 应用系统监控指标，例如接口吞吐率、响应时间、错误请求数等。

指标主题也可以作为本地 Prometheus 的远程存储，以 Remote Write 的方式将本地 Prometheus 中的指标上报至指标主题。

上报地址

网络类型	地址
外网	https://`${region}`.cls.tencentcs.com/prometheus/`${topicId}`/api/v1/write
内网	https://`${region}`.cls.tencentyun.com/prometheus/`${topicId}`/api/v1/write

其中：

- `${region}` 需替换为指标主题所在的地域，例如 ap-beijing，更多地域简称参见 可用地域，目前仅支持北京、上海、广州及南京地域。
- `${topicId}` 需替换为指标主题 ID，例如 0e69453c-0727-4c9c-xxxx-ea51b10d2aba，可在 指标主题列表 中查询。

鉴权方式

指标上报采用 Basic Auth 进行鉴权，将 API 密钥 中的 SecretId 及 SecretKey 分别作为 username 及 password。

- username: `${SecretId}`
- password: `${SecretKey}`

建议您为采集器单独创建一个子账号，使用该账号的 SecretId 及 SecretKey。该账号仅授予如下权限即可，以保障账号安全，配置方式详见 子账号授权 。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:MetricsRemoteWrite"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

配置示例

Telegraf

在 Telegraf 配置文件 中添加如下的 output 配置：

```
[[outputs.http]]
  ## 上报地址，请替换其中的${region}及${topicId}，本示例采用外网地址，如果网络条件允许建议使用内网地址
  ## 内网地址 url = "https://${region}.cls.tencentyun.com/prometheus/${topicId}/api/v1/write"
  url = "https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v1/write"
```

版权所有：腾讯云计算（北京）有限责任公司

```
  ## 鉴权信息，请替换其中的${SecretId}及${SecretKey}
  username = "${SecretId}"
  password = "${SecretKey}"

  ## Telegraf output 数据格式配置，请勿修改
  data_format = "prometheusremotewrite"
  [outputs.http.headers]
     Content-Type = "application/x-protobuf"
     Content-Encoding = "snappy"
     X-Prometheus-Remote-Write-Version = "0.1.0"
```

Vmagent

启动 Vmagent 时使用如下参数配置 remoteWrite：

```
./vmagent-prod \
  -remoteWrite.url=https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v1/write \
  -remoteWrite.basicAuth.username=${SecretId} \
  -remoteWrite.basicAuth.password=${SecretKey}
```

说明：

- 其中 remoteWrite.url 为外网地址，如果网络条件允许建议使用内网地址。
- 内网地址为 `https://${region}.cls.tencentyun.com/prometheus/${topicId}/api/v1/write` 。

Prometheus

在 Prometheus 配置文件 中添加如下配置：

```
# 上报地址，请替换其中的${region}及${topicId}，本示例采用外网地址，如果网络条件允许建议使用内网地址
# 内网地址 url: https://${region}.cls.tencentyun.com/prometheus/${topicId}/api/v1/write
url: https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v1/write

# 鉴权信息，请替换其中的${SecretId}及${SecretKey}
basic_auth:
  username: ${SecretId}
  password: ${SecretKey}

# 数据写入策略，包括缓存及重试等，数据量较大时可参考如下配置
queue_config:
  capacity: 20480
  min_shards: 100
  max_samples_per_send: 2048
  batch_send_deadline: 20s
  min_backoff: 100ms
  max_backoff: 5s
```

更多配置说明，详见 Prometheus 官方文档 。

Prometheus Operator

使用 Prometheus Operator 在 Kubernetes 中部署的 Prometheus 可通过如下方式上报指标：

1. 使用 kubectl 创建鉴权所需的 Secret，例如如下命令：

```
kubectl create secret generic kubepromsecret \
 --from-literal=username=${SecretId} \
 --from-literal=password=${SecretKey} \
```

```
-n monitoring
```

说明：

- 请替换其中的 `${SecretId}` 及 `${SecretKey}`
- 如果 Prometheus 部署在其它 namespace 中，请修改其中的 `-n monitoring` 为正确的 namespace。
- 也可以使用 manifest file 创建 Secret，详见 kubernetes 官方文档 。

2. 打开 Prometheus manifest 配置文件，配置文件一般位于 Prometheus Operator 使用的 GitHub repo 文件夹中，路径一般为 `kube-prometheus/manifests/prometheus-prometheus.yaml` 。
3. 修改 Prometheus manifest 配置文件，在末尾添加如下配置：

```
remoteWrite:
- url: "https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v1/write"
  basicAuth:
    username:
      name: kubepromsecret
      key: username
    password:
      name: kubepromsecret
      key: password
```

说明：

- 请替换其中的 `${region}` 及 `${topicId}` ，本示例采用外网地址，如果网络条件允许建议使用内网地址。
- 内网地址为 `url: "https://${region}.cls.tencentyun.com/prometheus/${topicId}/api/v1/write"`
- 数据量较大时可添加如下数据写入策略，包括缓存及重试等，详见 prometheus-operator 官方文档 。

```
    queueConfig:
      capacity: 204800
      minShards: 100
      maxShards: 2048
      maxSamplesPerSend: 4096
      batchSendDeadline: 30s
      minBackoff: 100ms
      maxBackoff: 5s
```

4. 应用配置文件，例如如下命令：

```
kubectl apply -f prometheus-prometheus.yaml -n monitoring
```

说明：

- 请替换 `prometheus-prometheus.yaml` 为正确的配置文件路径
- 如果 Prometheus 部署在其它 namespace 中，请修改其中的 `-n monitoring` 为正确的 namespace。

日志转指标

最近更新时间：2025-06-27 11:06:31

存储在日志主题中的原始日志可通过 SQL 进行统计，获取其中的指标。例如通过以下 SQL 获取错误请求数，错误请求数 errorRequestCount 即为一个指标：

```
status:>=400 | select count(*) as errorRequestCount
```

这种获取指标的方式非常灵活，可通过调整检索条件和 SQL 语句随时获取新的指标或更改指标计算规则，但其依赖原始日志完整存储于日志主题 标准存储 中，如果需要查询最近60天的指标变化趋势，则原始日志至少需存储60天，带来较大的数据存储成本。
当指标比较固定时，可将原始日志转换为指标进行存储，降低原始日志存储天数（也可以使用 日志沉降 及 投递到 COS 来低成本保存历史日志），进而降低整体成本。除此之外，由于转换后指标的数据条数一般远小于原始日志条数，指标主题也针对时序类型的数据查询进行了专门的优化，对转换后的指标进行查询时性能往往更好，可以更快的查询指标、绘制仪表盘。

转换逻辑

日志服务通过 定时 SQL 分析 将原始日志转换为指标，例如某一分钟有10条日志，通过 定时 SQL 分析 将这10条日志转换为指标。
原始日志：

```
{"requestTime":"2020/12/30 15:35:46","method":"POST","handler":"/messages","timeCost":"45","status":"200"}
{"requestTime":"2020/12/30 15:35:46","method":"POST","handler":"/users","timeCost":"23","status":"200"}
{"requestTime":"2020/12/30 15:35:47","method":"POST","handler":"/messages","timeCost":"57","status":"200"}
{"requestTime":"2020/12/30 15:35:47","method":"POST","handler":"/users","timeCost":"25","status":"200"}
{"requestTime":"2020/12/30 15:35:47","method":"POST","handler":"/users","timeCost":"689","status":"200"}
{"requestTime":"2020/12/30 15:35:48","method":"POST","handler":"/users","timeCost":"87","status":"200"}
{"requestTime":"2020/12/30 15:35:48","method":"POST","handler":"/users","timeCost":"34","status":"200"}
{"requestTime":"2020/12/30 15:35:48","method":"POST","handler":"/users","timeCost":"698","status":"200"}
{"requestTime":"2020/12/30 15:35:48","method":"POST","handler":"/messages","timeCost":"25","status":"200"}
{"requestTime":"2020/12/30 15:35:48","method":"POST","handler":"/messages","timeCost":"78","status":"200"}
```

定时SQL：

```
* | select count(*) as requests_total,method,handler group by method,handler order by requests_total desc
limit 10000
```

转换为如下指标：

```
requests_total{method="POST", handler="/messages"} 4
requests_total{method="POST", handler="/users"} 6
```

定时 SQL 任务可周期性执行，例如每分钟执行一次，每次对最近一分钟的原始日志进行转换，即可自动的将原始日志转换为分钟粒度的指标。

前提条件

- 已上报日志至日志主题，且采用标准存储，低频存储因为不支持 SQL 所以无法转换为指标。
- 该日志主题已 配置索引。

操作步骤

步骤1：添加定时 SQL 分析

1. 登录 日志服务控制台。
2. 在左侧导航栏中，选择**定时 SQL 分析**，单击**新建**，新建任务。
3. 在**基本配置**中填写如下信息：

配置项	说明

源日志主题	定时 SQL 任务的输入，即需要转换为指标的原始日志所在的日志主题。
写入目标	定时 SQL 任务的输出，选择**指标主题**，将转换后的指标存储在指定的指标主题中。如当前没有指标主题，可 创建指标主题 。
执行语句	使用 SQL 语句 对原始日志进行统计以获取指标，例如： `* | select count(*) as requests_total,method,handler group by method,handler order by requests_total desc limit 10000`
指标名称	从 SQL 语句结果中选择需要作为指标名称的字段，例如上面的例中 `requests_total` 为指标名称，其值为指标值。
指标维度	从 SQL 语句结果中选择需要作为指标维度的字段，例如上面的例中 `method` 及 `handler` 为指标维度。

4. 单击**下一步**，在**调度配置**中填写如下信息：

配置项	说明
调度范围	查询日志的时间范围，例如需要查询2023年1月1日 00:00:00-2023年3月31日00:00:00的日志数据，结束时间为空意为任务将持续运行。 推荐使用将结束时间设为空，持续将新采集的日志转换为指标。
调度周期	周期性查询，取值范围1-1440分钟，意为每 X 分钟会查询一次日志。 推荐每1分钟执行一次，更加及时的将日志转化为指标。
查询时间窗口	指定查询语句的时间窗口。 推荐使用**@m-1m ~ @m**，即最近1分钟，结合调度周期的**1分钟**，相当于每分钟将这一分钟的日志转化为指标。
延迟执行 SQL	查询延迟的时间，在**高级设置**中，默认60秒。日志生成索引一般会有延迟，在索引生成之前不可查询，因此建议设置60秒延迟查询，此时索引已生成（99.9%的索引数据将在5秒内生成）。

5. 单击**确定**，保存配置。

步骤2：查询生成的指标

1. 在左侧导航栏中，选择**检索分析**，在顶部选择指标主题所在的地域，主题类型选择“指标主题”，然后选择需要的指标主题。
2. 在输入框中输入指标名称查询指标。

云产品指标订阅

最近更新时间：2025-07-03 17:35:32

日志服务（Cloud Log Service，CLS）目前已支持主动订阅 腾讯云可观测平台（原云监控）的云产品指标。本文将介绍如何通过云产品指标订阅，将云产品指标接入日志服务。

前提条件

- 已配置云产品指标订阅相关权限，详情请参见 CLS 访问策略模板 。
- 已创建日志集与指标主题，具体操作请参见 日志集 和 指标主题 。

操作步骤

步骤1：选择指标主题

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**指标主题**，选择需要投递的指标主题，单击指标主题名称，进入指标主题详情页面。若您尚未创建指标主题，可单击**创建指标主题**进行创建，详情请参见 指标主题 。
3. 在指标主题详情页中，选中左侧菜单中的**接入管理** > **云产品指标订阅**，并在右侧的云产品指标订阅规则管理页面中单击**新建**，进入云产品指标订阅配置流程。



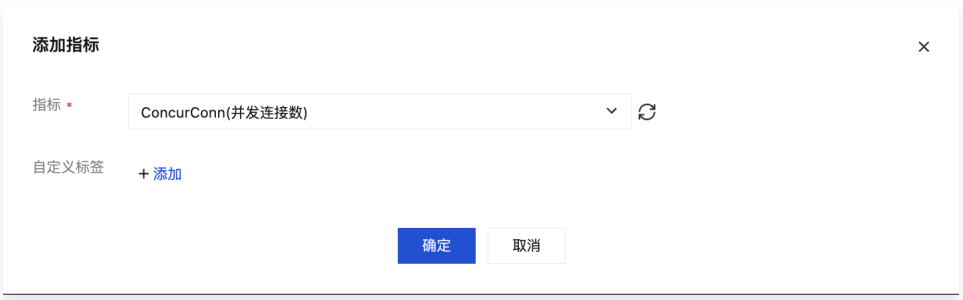
步骤2：配置云产品指标订阅

1. 在配置云产品指标订阅步骤中，进行如下配置：



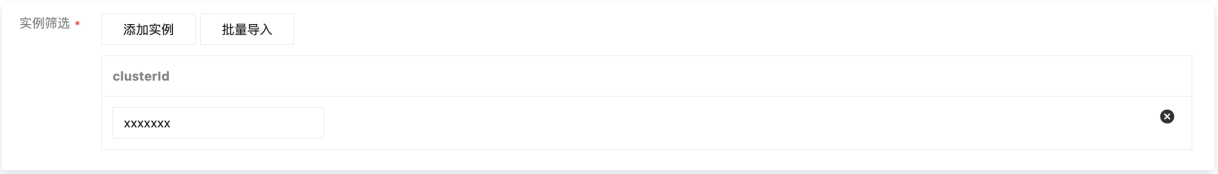
   - 配置名称：指定当前订阅配置的名称。
   - 云产品：选择需要采集的目标云产品对应的 Namespace。您可参考 云产品监控指标 查询您需要采集的云产品对应的 Namespace。

- 指标配置：单击**添加指标**，在弹窗中配置如下参数：



  - 指标：选择需要订阅的指标。
  - 自定义标签：您可选择为订阅的指标添加自定义的标签，支持添加多个。
- 实例筛选：提供手动添加和批量导入的形式筛选目标实例对象。
  - 手动添加：您可单击**添加实例**，并根据填写目标实例的维度信息：

> **说明：**
> 每一个实例对象都是格式为 key-value 键值对形式的集合。不同类型的实例所需要的维度组合可能不同，如 CVM 的每一个实例需要通过 InstanceID 来唯一指定，而 COS 则需要 appid 和 bucket 来唯一指定一个存储桶。详情可参见 云产品监控指标 。



  - 批量导入：您可单击**批量导入**，并在弹窗中导入实例维度组合 CSV 文件：
    - 覆盖导入：导入的实例维度组合将覆盖当前已配置的实例维度组合
    - 追加导入：导入的实例维度组合将追加至当前已配置的实例维度组合

> **说明：**
> 若您不确定 CSV 的格式，可在完成步骤1相关配置后，单击导出现有实例，查看所需 CSV 的文件格式要求。



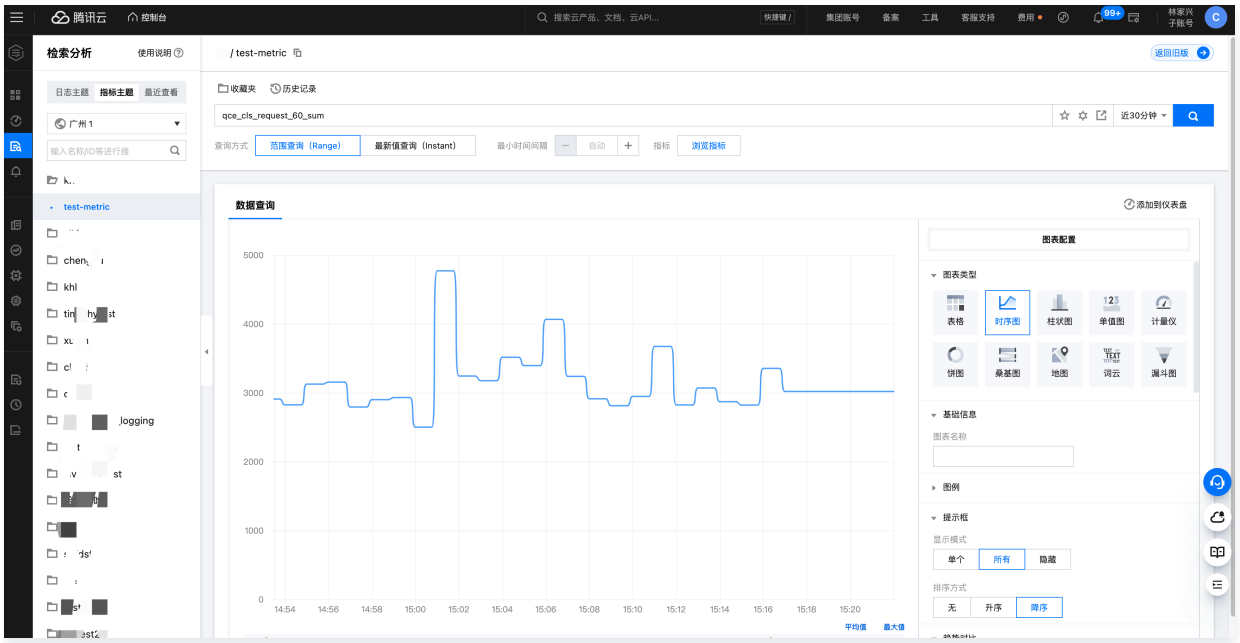
2. 预览指标：您可单击**预览**，对当前配置的指标进行预览，如下图：



3. 保存：指标预览确认无误后，您即可单击**提交**保存指标订阅配置。

步骤3：检索分析

至此，您已完成云产品指标订阅的配置。下一步，您可前往 检索页，在控制台左侧选中目标指标主题，即可进行指标的检索和查询了。有关指标的检索的语法规则可参考 指标检索语法规则 。



其他操作

管理订阅任务

完成云产品指标订阅配置后，您可在指标主题详情页的**云产品指标订阅**中查看并管理所有已创建的云产品指标订阅任务。



创建仪表盘

请参见 仪表盘 。

配置告警策略

请参见 监控告警 。

K8s 指标采集

最近更新时间：2024-07-19 14:09:01

本文将介绍如何在 自建K8s或TKE集群上通过VMagent采集指标至CLS。

操作步骤

安装 VMagent

> **依赖：**
> - 已安装 kubectl，并且 kubectl 可正常访问集群。
> - Linux 环境，不支持 MacOS。

1. 执行以下命令下载 VMagent 安装脚本。

```
wget https://mirrors.tencent.com/install/cls/vmagent/vmagent.sh
```

2. 执行以下命令安装安装 VMagent。

```
./vmagent.sh install --secretid <secretid> --secretkey <secretkey> --endpoint https://<region-
url>/prometheus/<topicID>/api/v1/write --cluster_id <cluster_id>
```

> **说明：**
> 安装命令中，--secretid、--secretkey、--endpoint、cluster_id 为必填，详情请见 参数说明 。

参数说明

参数名	是否必填	参数描述
secretid	是	云 API 密钥 的一部分，SecretId 用于标识 API 调用者身份。
secretkey	是	云 API 密钥 的一部分，SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥。
endpoint	是	访问 CLS 指标主题的 url，路径由 CLS 域名和 topicID 组成，详情请见 指标上报地址 。
cluster_id	是	集群 ID，作为指标 label，区分指标的来源。
namespace	否	指定安装 VMagent 的 namespace。若不指定，默认为 kube-system（一般不需要特别指定）
name	否	指定安装 VMagent 的 deployment 的名称。若不指定，默认为 VMagent（一般不需要特别指定）
dep	否	在安装的时候是否安装依赖组件(node-exporter、kube-state-metric)，默认值为 false。

配置指标采集

参考以下格式创建指标采集配置 ConfigMap：

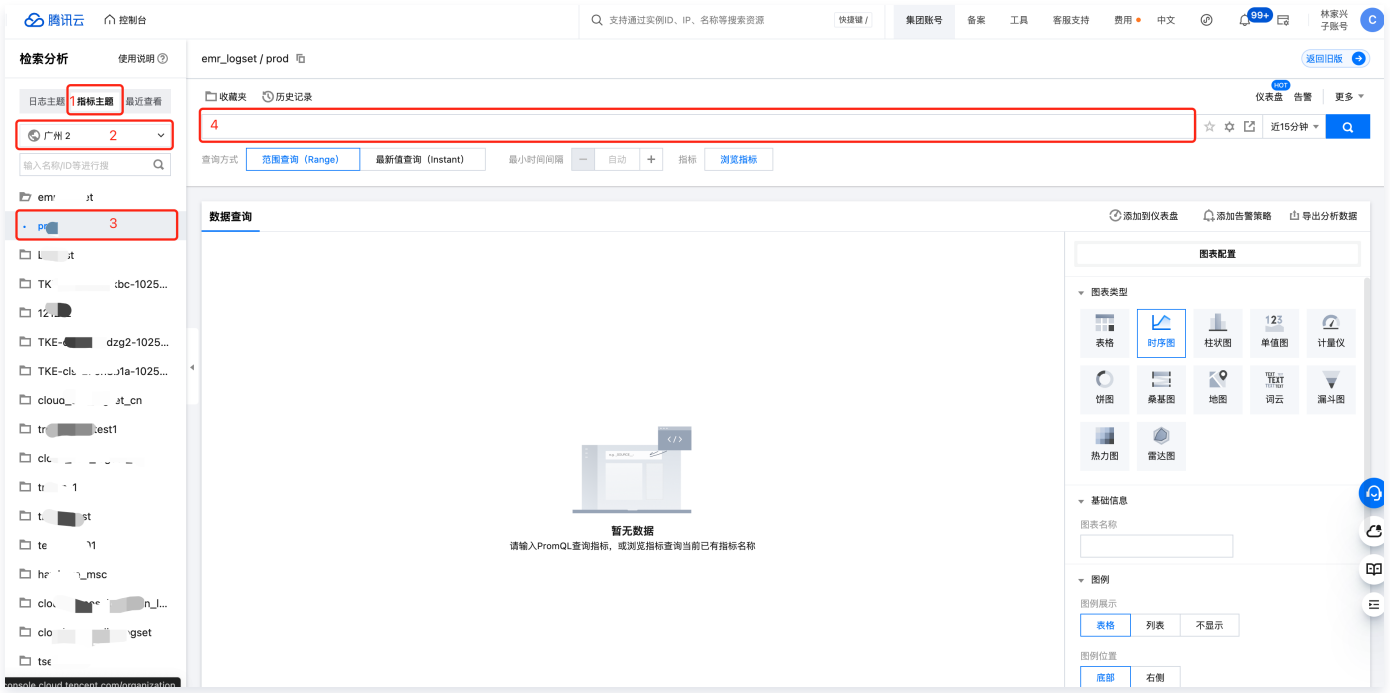
> **说明：**
> 如何创建 ConfigMap 请参见 创建 ConfigMap 。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: ${name}-config        ## 安装VMagent时指定的name参数
  namespace: ${namespace}     ## 安装VMagent时指定的namespace参数
data:
```

```
vmagent.yml: |-
  ## Prometheus指标采集配置，参考https://prometheus.io/docs/prometheus/latest/configuration/configuration/
  global:
    scrape_interval: 15s
    external_labels:
      cluster: ${clusterID}  ## 安装VMagent时指定的clusterID参数
  scrape_configs:
  - job_name: xxx
```

查询采集的指标

1. 登录 日志服务控制台，在左侧导航栏中选择**检索分析**，在顶部选择**指标主题**，以及指标主题的所在地域，然后选择需要的指标主题。



2. 在输入框中输入指标名称查询指标。

卸载 VMagent

执行以下命令即可卸载 VMagent：

```
./vmagent.sh uninstall --name <vmagent-name> --namespace <ns-name> --dep true
```

参数说明

参数名	是否必填	参数描述
namespace	否	VMagent 的 namespace。若不指定，默认为 kube-system。
name	否	VMagent的 deployment的名称。若不指定， 默认为 VMagent。
dep	否	是否卸载依赖组件(node-exporter、kube-state-metric)，默认为 true。

主机监控指标采集

最近更新时间：2025-07-04 18:56:22

本文将介绍如何在通过 LogListener 采集主机相关监控指标数据（CPU、内存、网络、磁盘、系统等指标）到 CLS。

前提条件

- 已配置主机指标采集相关权限，详情请参见 CLS 访问策略模板 。
- 已创建日志集与指标主题，具体操作请参见 日志集 和 指标主题 。

使用限制

- 需要采集的主机，已安装loglistener，且版本≥3.4.2。若您的版本低于 3.4.2，需先升级。具体操作见 LogListener 升级指南 。
- 不支持 Windows 版本。

操作步骤

步骤1：选择指标主题

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**指标主题**，选择需要投递的指标主题，单击指标主题名称，进入指标主题详情页面。若您尚未创建指标主题，可单击**创建指标主题**进行创建，详情请参见 指标主题 。
3. 在指标主题详情页中，选中左侧菜单中的**指标接入 > 主机监控指标**，并在右侧的管理页面中单击**新建**，进入主机监控指标配置流程。



步骤2：选择机器组

1. 选择机器组。选择需要进行主机监控指标采集的机器组

创建新的机器组

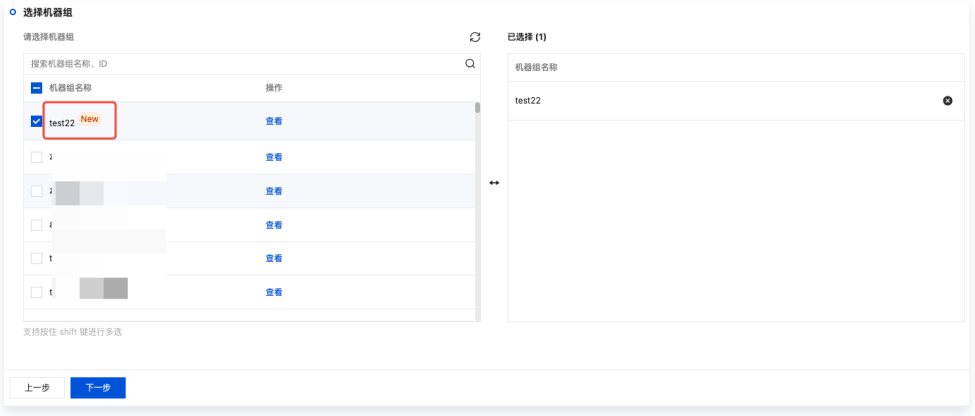
如果您想创建新的机器组，可单击新建机器组：



1. 填写机器组名称，通过机器标识的方式关联**已安装 LogListener** 的目标服务器。详情请参见 **机器组** 。然后单击**下一步**。

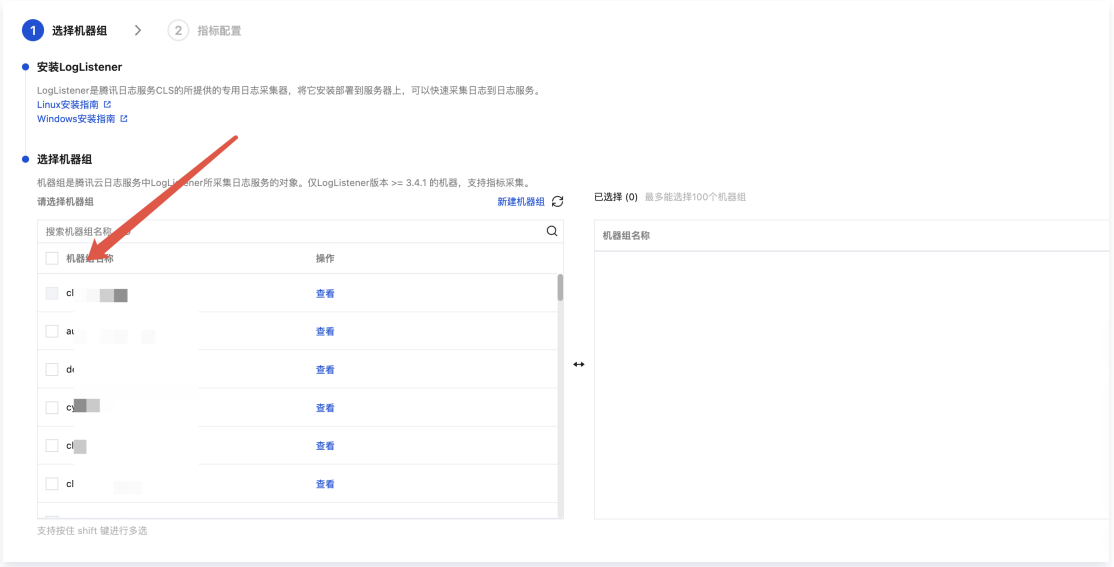


2. 完成创建后，在列表中勾选您创建的机器组。



选择已有的机器组

如果您想选择已有的机器组，可执行如下操作：
在列表中勾选您的目标机器组，然后单击**下一步**。



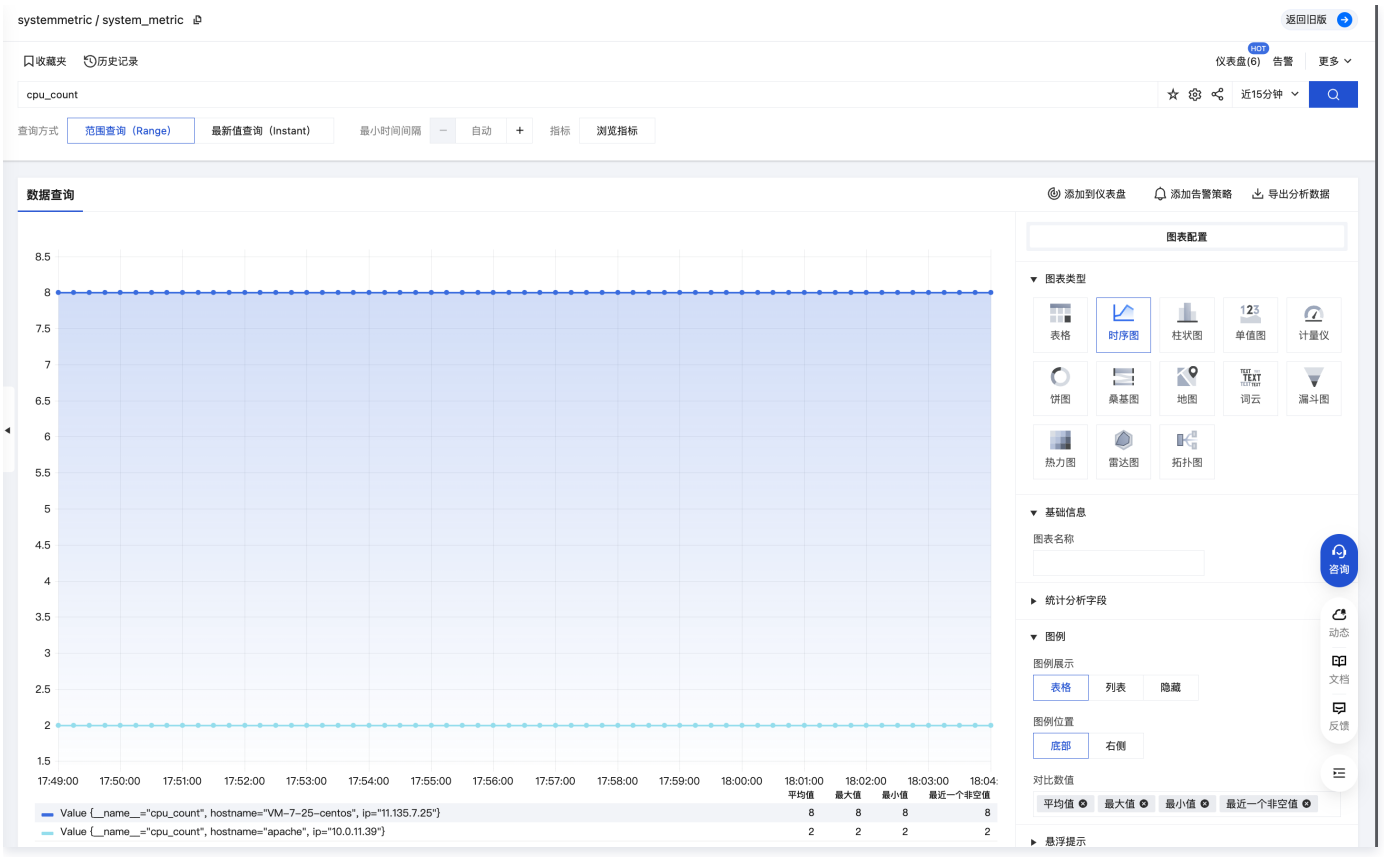
2. 配置采集名称以及采集时间间隔。



指标采集时间间隔：每次请求的时间间隔，单位为ms，默认为30000，不能低于5000。

步骤3：检索分析

至此，您已完成云产品指标订阅的配置。下一步，您可前往 检索分析页面，在控制台左侧选中目标指标主题，即可进行指标的检索和查询了。有关指标的检索的语法规则可参考 指标检索语法规则。



其他操作：查看主机指标数据监控仪表盘

CLS 提供主机监控的预制仪表盘能力。单击主机监控指标列表页的**查看预置仪表盘**，即可进入主机监控指标统计大盘



在主机指标监控预制仪表盘中即可查看所有已采集到的数据指标的可视化统计大盘



指标说明

所采集的主机监控指标说明如下：

- CPU 相关指标

指标名	说明	单位
cpu_count	CPU 核数	个
cpu_util	CPU 使用率，计算方式为排除 idle、wait、steal 后的占比。	%
cpu_guest_util	客户时间（guest time）占比	%
cpu_guestnice_util	Nice 进程客户时间（nice guest time）占比	%
cpu_irq_util	硬中断处理时间（Hard Irq time）占比	%
cpu_nice_util	Nice 时间（Nice time）占比	%
cpu_softirq_util	软中断处理时间（Soft Irq time）占比	%
cpu_steal_util	等待宿主机 CPU 时间（Steal time）占比	%
cpu_sys_util	内核态（System time）占比	%
cpu_user_util	用户态（User time）占比	%
cpu_wait_util	等待 IO（Waiting time）占比	%

- 内存相关指标

指标名	说明	单位
mem_util	内存使用率	%
mem_cache	已申请但未使用的内存	byte
mem_free	未使用的内存	byte
mem_available	可用内存	byte
mem_used	已使用内存	byte
mem_swap_util	Swap 内存使用率	%

mem_total	内存总量	byte

- 系统相关指标

指标名	说明	单位
system_boot_time	系统启动时间	s
system_load1	系统平均负载，1分钟平均值	不涉及
system_load5	系统平均负载，5分钟平均值	不涉及
system_load15	系统平均负载，15分钟平均值	不涉及
protocol_tcp_established	已建立的 TCP 连接数	个
protocol_tcp_insegs	接收的所有 TCP 报文数	个
protocol_tcp_outsegs	发送的 TCP 报文数	个
protocol_tcp_retran_segs	重传的 TCP 报文数	个
protocol_tcp_retran_util	重传报文占总发送报文数量的比值	%
fd_allocated	已分配的文件描述符数量	个
fd_max	最大可分配的文件描述符数量	个

- 磁盘相关指标

指标名	说明	单位
disk_rbps	硬盘每秒读取流量	byte/s
disk_wbps	硬盘每秒写入流量	byte/s
disk_riops	硬盘每秒读取次数	次/s
disk_wiops	硬盘每秒写入次数	次/s
disk_rlatency	平均读延迟	ms
disk_wlatency	平均写延迟	ms
disk_util	IO 使用率	%
disk_space_usage	磁盘使用百分比	%
disk_inode_usage	inode 使用率	%
disk_space_used	磁盘已使用容量	byte
disk_space_total	磁盘总量	byte
disk_inode_total	inode 总量	个
disk_inode_used	inode 已使用容量	个

- 网络相关指标

指标名	说明	单位
net_drop_util	丢弃的数据包占总数据包的比值	%
net_err_util	报错数据包占总数据包的比值	%
net_in	网络接收速率	byte/s

版权所有：腾讯云计算（北京）有限责任公司

net_in_pkt	每秒接收的数据包	个/s
net_out	网络发送速率	byte/s
net_out_pkt	每秒发送的数据包	个/s

日志存储
日志存储概述

最近更新时间：2024-06-26 17:37:51

根据用户对日志不同的检索时延性及日志处理能力需求，日志服务（Cloud Log Service，CLS）提供两种存储类型：**标准存储**和**低频存储**。

基本概念

存储类型：分为标准存储与低频存储两种。两种存储类型的存储介质不同，功能和性能有差异，价格也不相同。
日志沉降：是一种日志分层存储的方式。将需要 SQL 分析、时效要求高的新数据保存在标准存储里；不需要 SQL 分析、时效性要求低的历史数据保存到低频存储，降低日志长周期存储的成本。详情请参见 日志沉降 。

功能特性

功能点	低频存储	标准存储
索引建立	✓（只支持全文索引）	✓
上下文检索	✓	✓
快速分析	×	✓
全文检索	✓（亿条数据2秒响应）	✓（亿条数据0.5秒响应）
键值检索	×	✓
日志下载	✓	✓
SQL 分析	×	✓
仪表盘	×	✓
监控告警	×	✓
投递到 COS	✓	✓
投递到 Ckafka	✓	✓
投递到 ES	✓	✓
投递到 SCF	✓	✓
日志消费	✓	✓
数据加工	✓	✓

产品优势

标准存储

适用于用户对日志有统计分析的需求，提供日志的秒级检索，实时统计分析，实时监控，流式消费等应用能力。

- 运维监控排障：对线上问题进行实时诊断，利用日志秒级检索能力，快速检索散落在多个机器的日志内容，定位故障原因并进行修复；基于日志实时计算各项质量指标，一旦指标超过阈值即告警，便于开发运维人员第一时间发现问题修复故障。
- 流式处理：采集散落在多台机器上 PB 级别埋点日志数据，实时流式拉取至用户自建的大数据处理集群用于后续数据湖计算，例如用于推荐系统的模型数据计算业务等。

低频存储

适用于访问频率较低且无统计分析需求的日志，如审计归档日志。低频存储提供全文检索日志的能力，满足用户对历史日志的回溯检索，归档存储等诉求，整体使用成本相比**标准存储**降低80%。

- 历史日志存储：日志数据的爆炸性增长，对于数月乃至数年的日志进行大规模存储和分析，其成本变得十分高昂。这可能会导致用户删除有价值的数据，而错过了长期数据可能产生的重要洞察和见解。**低频存储**能满足用户以低成本的方式对历史数据进行大规模的检索，历史回溯。

版权所有：腾讯云计算（北京）有限责任公司

- 非关键业务日志：排查异常时，研发人员更多需要关注 ERROR/WARN 日志以及针对其进行监控告警，INFO 等非关键业务日志只作为归档存储，在特定场景下才需要触发日志的检索分析。一般用户对该类日志检索时延无特定要求，使用**低频存储**存储非关键业务日志能显著降低用户使用成本，且满足用户低频检索的诉求。
- 审计日志场景：将操作审计、安全审计的日志采集至**低频存储**，通过 CLS 低频检索能力分析其访问行为，例如某个账号、某个对象的操作记录等，判断是否存在违规操作。同时，满足合规审计需求，日志存储180天以上。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，进入日志主题列表，单击**创建日志主题**。

费用说明

标准存储与低频存储各计费项价格不同，详见 计费概述 。
示例：每天写入2TB原始日志数据，保存30天，压缩比1：8，索引比例100%，以广州地域定价为例。

计费项	标准存储	低频存储	说明
写流量	1350元	1350元	价格均为0.18元/GB/天
索引流量	21500元	6140元	标准存储：0.35元/GB/天 低频存储：0.1元/GB/天
日志存储	2650元	570元	标准存储：0.0115元/GB/天 低频存储：0.0025元/GB/天
索引存储	21200元	4600元	标准存储：0.0115元/GB/天 低频存储：0.0025元/GB/天
其他费用	极少	极少	包含接口调用次数，分区费用等
总价（月）	46700元	12660元（下降73%）	-

规格与限制

日志存储的规格与限制，请参见 规格与限制 。

版权所有：腾讯云计算（北京）有限责任公司

日志沉降

最近更新时间：2025-06-26 18:56:52

CLS 日志沉降功能，支持日志接入标准存储一段时间后，将数据从标准存储沉降到低频存储保存。在保证一定日志统计分析能力的基础上，大幅降低日志的存储成本。详情可参考 日志存储概述 。

操作场景

操作与访问类日志具有巨大的业务价值，可用于业务洞察、客户服务、推广运营、用户行为分析等场景，需要对天、周级别的数据进行统计分析。同时此类日志也要满足180天保存时间的合规要求，保证日志可查询，用于攻击溯源等场景。
该场景需求呈现以下特征：

- 天级别到周级别的数据，业务价值高，需要用于各类统计分析图表，使用频次高，响应要及时。主要关注其**实时性与分析能力**。
- 月级别的数据，业务价值低，基本不再需要分析，使用频次低，同时数据量非常大。主要关注其**存储成本**。

针对以上特征，CLS 日志沉降支持将高价值的新数据接入标准存储做实时分析统计与告警，经过一段周期后（一般是几天到几周）将价值与使用频次降低的旧数据，沉降到存储成本更低的低频存储长时间保存。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，进入日志主题列表，单击**创建日志主题**，或者在日志主题列表选择已创建的日志主题，在基本信息页面进行**编辑**。
   - 配置**日志保存策略**，日志接入选择**标准存储**，日志保存时间支持日志永久保存和固定期限保存。
   - 开启**日志沉降**开关，根据业务需求选择日志在标准存储保存的周期。

     > **注意：**
     > 只有日志保存时间在**7天及以上**时，才可以开启日志沉降功能。

   - 其他配置可根据实际情况填写。



3. 配置完成后单击**确定**。
4. 保存后，日志将先接入标准存储一段周期，然后沉降到低频存储，存储费用大幅降低。如例所示配置，日志将在标准存储保存7天，然后沉降到低频存储保存23天后过期。

各日志存储方案对比

功能点	低频存储	标准存储	标准存储沉降低频存储
适用场景	只检索不分析的日志	需要实时分析与告警，不需要长期留存的日志	短期需要实时分析与告警，长期需要合规留存的日志
索引建立	✓（只支持全文索引）	✓	✓
上下文检索	✓	✓	✓
快速分析	×	✓	✓（仅标准存储的数据支持）
全文检索	✓（亿条数据2秒响应）	✓（亿条数据0.5秒响应）	✓
键值检索	×	✓	✓
日志下载	✓	✓	✓

SQL 分析	×	✓	✓（仅标准存储的数据支持）
仪表盘	×	✓	✓（仅标准存储的数据支持）
监控告警	×	✓	✓（仅标准存储的数据支持）
投递到 COS	✓	✓	✓
投递到 CKafka	✓	✓	✓
投递到 ES	✓	✓	✓
投递到 SCF	✓	✓	✓
日志消费	✓	✓	✓
数据加工	✓	✓	✓
定时 SQL 任务	×	✓	✓（仅标准存储的数据支持）

日志沉降常见问题

如何进行日志沉降计费

开启日志沉降的日志主题，数据将先接入**标准存储**，产生**写流量**、**标准索引流量**费用。接入后在标准存储保存一段时间，期间产生**标准日志存储**、**标准索引存储**的费用。而后数据沉降到**低频存储**长时间保存，直到过期。期间产生**低频日志存储**、**低频索引存储**费用。
例如：每天写入1TB原始日志数据，保存180天，压缩比1:8，索引比例100%，以广州地域定价为例，每个月用量和花费如下：

计费项	标准存储	标准存储沉降低频存储
日志保存策略	标准存储保存180天	标准存储保存30天后，沉降到低频存储保存150天
写流量	691元（用量3.75TB）	691元（用量3.75TB）
标准索引流量	10752元（用量30TB）	10752元（用量30TB）
标准日志存储	7949元（用量675TB）	1325元（用量112.5TB）
标准索引存储	63590元（用量5400TB）	10598元（用量900TB）
低频日志存储	-	1440元（用量562.5TB）
低频索引存储	-	11520元（用量4500TB）
其他费用	极少	极少
总价（每月）	82982元	36326元（总成本下降56%，单看沉降到低频的数据存储成本下降78.3%）

修改日志沉降周期

日志沉降支持修改沉降周期，修改经过一段时间后生效。例如：修改7天标准存储，3天低频存储的日志主题为10天标准存储。修改生效期间，数据保持旧沉降周期不变。经过1小时的生效期，数据迁移完成，开始按照新沉降周期生效（当前标准主题从低频存储恢复数据到标准存储不产生费用）。

历史日志沉降到低频存储与投递到 COS 的场景对比

日志接入标准存储一段周期后（一般是几天到几周），需要将历史数据转存到成本更低的存储类型。此时用户可以依据场景选择沉降到低频存储或投递到 COS 做归档。
其中，沉降到低频会产生低频日志存储（按压缩后计算，为0.0025元/GB/天）、低频索引存储（0.0025/GB/天）费用；投递到 COS 会产生读流量费用（0.18元/GB）、COS 存储费用（例如 COS 标准存储，按压缩后计算，为0.0039元/GB）。
例如：每天写入1TB原始日志数据，标准存储30天后，做合规留存。压缩比1:4，以广州地域定价为例。
场景一：标准存储**30**天后，再留存**30**天，索引占比**100%**。（总计保存60天）

场景需求	功能	用量	单价	费用	总计费用
沉降到低频存储的方案	• 支持日志检索查询 • 不支持 SQL 分析、仪表盘、告警等能力	写流量：256GB/天	0.18元/GB	写流量：46.08元/天	942.08元
		标准索引流量：1024GB/天	0.35元/GB	标准索引流量：358.4元/天	
		标准索引存储：30720GB/天	0.0115元/GB	标准索引存储：353.28元/天	
		标准日志存储：7680GB/天	0.0115元/GB	标准日志存储：88.32元/天	
		低频索引存储：30720GB/天	0.0025元/GB	低频索引存储：76.8元/天	
		低频日志存储：7680GB/天	0.0025元/GB	低频日志存储：19.2元/天	
投递到 COS 标准存储的方案	不支持日志检索、SQL 分析等能力	写流量：256GB/天	0.18元/GB	写流量：46.08元/天	922.112元
		标准索引流量：1024GB/天	0.35元/GB	标准索引流量：358.4元/天	
		标准索引存储：30720GB/天	0.0115元/GB	标准索引存储：353.28元/天	
		标准日志存储：7680GB/天	0.0115元/GB	标准日志存储：88.32元/天	
		内网读流量（投递到 COS 产生的流量）：256GB/天	0.18元/GB	内网读流量：46.08元/天	
		COS 标准存储：7680GB/天	0.0039元/GB	COS 标准存储：29.952元/天	

场景二：标准存储30天后，再留存150天，索引占比100%。（总计保存180天）

场景需求	功能	用量	单价	费用	总计费用
沉降到低频存储的方案	• 支持日志检索查询 • 不支持 SQL 分析、仪表盘、告警等能力	写流量：256GB/天	0.18 元/GB	写流量：46.08元/天	1326.08元
		标准索引流量：1024GB/天	0.35 元/GB	标准索引流量：358.4元/天	
		标准索引存储：30720GB/天	0.0115元/GB	标准索引存储：353.28元/天	
		标准日志存储：7680GB/天	0.0115元/GB	标准日志存储：88.32元/天	
		低频索引存储：153600GB/天	0.0025元/GB	低频索引存储：384元/天	
		低频日志存储：38400GB/天	0.0025元/GB	低频日志存储：96元/天	
投递到 COS 标准存储的方案	不支持日志检索、SQL 分析等能力	写流量：256GB/天	0.18元/GB	写流量：46.08元/天	1041.92元
		标准索引流量：1024GB/天	0.35元/GB	标准索引流量：358.4元/天	
		标准索引存储：30720GB/天	0.0115元/GB	标准索引存储：353.28元/天	
		标准日志存储：7680GB/天	0.0115元/GB	标准日志存储：88.32元/天	
		内网读流量（投递到 COS 产生的流量）：256GB/天	0.18元/GB	内网读流量：46.08元/天	
		COS 标准存储：38400GB/天	0.0039元/GB	COS 标准存储：149.76元/天	

场景三：标准存储30天后，再留存150天，索引占比25%。（总计保存180天）

场景需求	功能	用量	单价	费用	总计费用
沉降到低频存储的方案	• 支持日志检索查询 • 不支持 SQL 分析、仪表盘、告警等能力	写流量：256GB/天	0.18元/GB	写流量：46.08元/天	504.32元
		标准索引流量：256GB/天	0.35元/GB	标准索引流量：89.6元/天	
		标准索引存储：7680GB/天	0.0115元/GB	标准索引存储：88.32元/天	
		标准日志存储：7680GB/天	0.0115元/GB	标准日志存储：88.32元/天	

版权所有：腾讯云计算（北京）有限责任公司

		低频索引存储：38400GB/天	0.0025元/GB	低频索引存储：96元/天	
		低频日志存储：38400GB/天	0.0025元/GB	低频日志存储：96元/天	
投递到 COS 标准存储的方案	不支持日志检索、SQL 分析等能力	写流量：256GB/天	0.18元/GB	写流量：46.08元/天	508.16元
		标准索引流量：256GB/天	0.35元/GB	标准索引流量：89.6元/天	
		标准索引存储：7680GB/天	0.0115元/GB	标准索引存储：88.32元/天	
		标准日志存储：7680GB/天	0.0115元/GB	标准日志存储：88.32元/天	
		内网读流量（投递到 COS 产生的流量）：256GB/天	0.18元/GB	内网读流量：46.08元/天	
		COS 标准存储：38400GB/天	0.0039元/GB	COS 标准存储：149.76元/天	

版权所有：腾讯云计算（北京）有限责任公司

指标存储
指标存储概述

最近更新时间：2024-08-08 15:20:21

基本概念

指标（Metric）

用于衡量系统和应用程序性能及运行情况的度量值，例如 CPU 利用率、内存使用率、访问吞吐量、响应耗时和响应成功率等。指标一般会定时产生，每一个时刻都会有一个值，随着时间变化形成一个序列，这个序列一般被称为时间序列（time series），简称时序。
日志服务（Cloud Log Service，CLS）兼容 Prometheus 指标数据模型，将相同指标名称（metric name）、相同维度（labels）的包含时间戳的指标数据保存为时间序列（time series）。在时间序列中每个数据点称为样本（sample），样本则由时间戳及样本值构成。
例如系统在2020/12/30 15:35:23.123时某一接口的请求总量就是一个样本，其数据如下：

```
requests_total{method="POST", handler="/messages"} 217
```

其由如下几部分构成：

- 指标名称：requests_total
- 维度：{method="POST", handler="/messages"}，即接口名称为 messages，请求方式为 POST。
- 时间戳：2020/12/30 15:35.123
- 样本值：217

被监控系统实际上往往同时有多个指标，其在某一时刻具备大量不同的指标名称和维度，例如 Nginx 监控指标如下：

```
# HELP nginx_http_requests_total The total number of HTTP requests
# TYPE nginx_http_requests_total counter
nginx_http_requests_total 10234

# HELP nginx_http_requests_duration_seconds The HTTP request duration in seconds
# TYPE nginx_http_requests_duration_seconds histogram
nginx_http_requests_duration_seconds_bucket{le="0.005"} 2405
nginx_http_requests_duration_seconds_bucket{le="0.01"} 5643
nginx_http_requests_duration_seconds_bucket{le="0.025"} 7890
nginx_http_requests_duration_seconds_bucket{le="0.05"} 9234
nginx_http_requests_duration_seconds_bucket{le="0.1"} 10021
nginx_http_requests_duration_seconds_bucket{le="0.25"} 10234
nginx_http_requests_duration_seconds_bucket{le="0.5"} 10234
nginx_http_requests_duration_seconds_bucket{le="1"} 10234
nginx_http_requests_duration_seconds_bucket{le="2.5"} 10234
nginx_http_requests_duration_seconds_bucket{le="5"} 10234
nginx_http_requests_duration_seconds_bucket{le="10"} 10234
nginx_http_requests_duration_seconds_bucket{le="+Inf"} 10234
nginx_http_requests_duration_seconds_sum 243.56
nginx_http_requests_duration_seconds_count 10234

# HELP nginx_http_connections Number of HTTP connections
# TYPE nginx_http_connections gauge
nginx_http_connections{state="active"} 23
nginx_http_connections{state="reading"} 5
nginx_http_connections{state="writing"} 7
nginx_http_connections{state="waiting"} 11

# HELP nginx_http_response_count_total The total number of HTTP responses sent
# TYPE nginx_http_response_count_total counter
nginx_http_response_count_total{status="1xx"} 123
nginx_http_response_count_total{status="2xx"} 9123
nginx_http_response_count_total{status="3xx"} 456
```

版权所有：腾讯云计算（北京）有限责任公司

```
nginx_http_response_count_total{status="4xx"} 567
nginx_http_response_count_total{status="5xx"} 65

# HELP nginx_up Is the Nginx server up
# TYPE nginx_up gauge
nginx_up 1
```

其中指标含义如下：

- nginx_http_requests_total：Nginx 处理的 HTTP 请求总数。
- nginx_http_requests_duration_seconds：HTTP 请求的持续时间，使用 Histogram 类型提供不同区间的请求数量。
- nginx_http_connections：当前 Nginx 的 HTTP 连接数，分为 active、reading、writing 和 waiting 状态。
- nginx_http_response_count_total：Nginx 返回的 HTTP 响应总数，按状态码分类。
- nginx_up：Nginx 服务器的运行状态。1表示运行中；0表示未运行。

指标主题

是指标数据在日志服务平台进行采集、存储、检索和分析的基本单元，采集到的指标数据以指标主题为单元进行管理，包括保存时间配置及检索分析等。指标主题兼容 Prometheus 指标数据模型及指标查询接口，相当于一个 Prometheus 实例，不同应用/服务的指标只要指标名称不发生冲突，且数据量未超过产品规格与限制，就可以存储在一个指标主题中。实际应用中，一般将业务系统生产环境、测试环境、开发环境的指标分别存储在不同的指标主题中。

> **说明：**
> 指标主题已于2024年7月1日结束公测并正式收费，详见 计费概述 。

功能特性

- 指标采集：
  - 指标上报：支持 Prometheus Remote Write 协议，可使用兼容该协议的各类采集器采集指标并上报至指标主题，例如 vmagent 及 telegraf。
  - 日志转指标：可通过定时 SQL 将日志主题中的日志转化为指标，适用于长期低成本存储系统关键指标，且基于指标进行可视化分析时性能往往更高。
  - 云产品指标订阅：支持主动订阅 腾讯云可观测平台（原云监控）的云产品指标，在 CLS 中集中存储及查询，可更灵活的对云产品指标进行统计分析。
- 指标查询：使用 PromQL 查询指标。
- 指标可视化：可使用 仪表盘 可视化展示指标数据，支持表格、时序图、单值图和计量仪等形式。也可直接 使用 Grafana 展示指标数据 。
- 监控告警：可为指标主题配置 告警策略，指标发生异常时通过短信、微信、电话、邮件和企业微信等多种渠道通知用户。

产品优势

- 指标主题兼容 Prometheus 指标数据模型及指标查询接口，可无缝与各类兼容 Prometheus 的开源项目对接，例如 Grafana。
- 相比自建 Prometheus，无需部署及维护，能够降低大量人力成本。
- 可与日志结合使用，集中采集、存储、分析指标及日志数据，搭建统一监控平台，提升运维效率。

费用说明

详见 计费概述 。

规格与限制

限制项	说明
指标名称（metric name）	支持英文字母、数字、下划线及冒号，即需符合正则表达式 `[a-zA-Z_:][a-zA-Z0-9_:]*`
维度名称（label name）	支持英文字母、数字、下划线，即需符合正则表达式 `[a-zA-Z_][a-zA-Z0-9_]*`
维度值（label value）	无特殊限制，支持各类 Unicode 字符
样本值（sample value）	float64类型数值
样本时间戳（sample timestamp）	毫秒精度
查询并发	单个指标主题支持15并发

查询数据量	单次查询涉及 time series 不超过20万，查询结果中单个 time series 最大1.1万个时间点
指标上传频控	25000QPS
指标上传流控	250MB/s

兼容 Prometheus 接口

最近更新时间：2024-07-01 16:46:11

指标主题兼容 Prometheus 指标数据模型及如下接口：

指标查询接口

Prometheus HTTP API

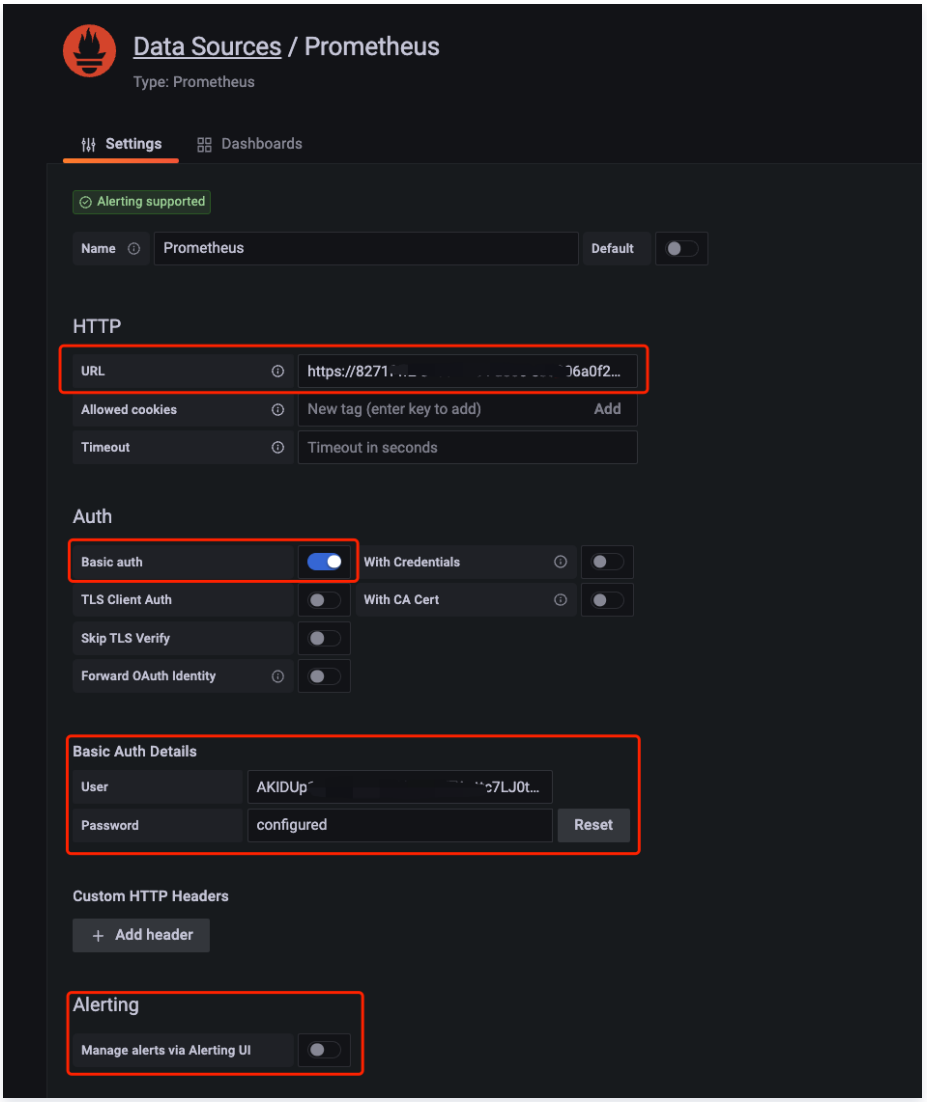
兼容如下 HTTP API，常用于对接 Grafana。

- Instant queries : /api/v1/query
- Range queries : /api/v1/query_range
- Finding series by label matchers : /api/v1/series
- Getting label names : /api/v1/labels
- Querying label values : /api/v1/label/<label_name>/values
- Querying exemplars : /api/v1/query_exemplars

不兼容用于原生 Prometheus 采集配置、告警规则和集群管理等功能的 API，例如： Targets、Rules、Alerts、Querying target metadata、Querying metric metadata、Alertmanagers、Status、TSDB Admin APIs 等。

对接 Grafana 操作步骤：

1. 在 **Grafana Data Sources** 页面，单击 **Add data source**。
2. 选择 **Prometheus**，在表单中配置如下信息：



- URL：请替换其中的 ${region} 及 ${topicId}，${region} 为 地域简称，${topicId} 为指标主题 ID。

- 外网地址：https://${region}.cls.tencentcs.com/prometheus/${topicId}
- 内网地址：https://${region}.cls.tencentyun.com/prometheus/${topicId}

- Basic auth：打开该开关
- Basic Auth Details：接口采用 Basic Auth 进行鉴权，将 API 密钥 中的 SecretId 及 SecretKey 分别作为 username 及 password。
  - username: `${SecretId}`
  - password: `${SecretKey}`

> **说明：**
> 建议您单独创建一个子账号，使用该账号的 SecretId 及 SecretKey。该账号仅授予如下权限即可，以保障账号安全，配置方式详见 子账号授权 。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:MetricsLabelValues",
                "cls:MetricsQueryRange",
                "cls:MetricsLabels",
                "cls:MetricsQuery"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

- Manage alerts via Alerting UI：关闭该开关，CLS 不兼容 Prometheus Alerts 相关接口，不支持该功能。

3. 单击底部 **Save & Test** 测试配置信息是否正确并保存配置。

Prometheus Remote Read API

常用于使用自建 Prometheus 读取指标主题数据，在 Prometheus 配置文件 中添加如下配置（更多配置说明详见 Prometheus 官方文档 ）：

```
# 读取地址，请替换其中的${region}及${topicId}，本示例采用外网地址，如果网络条件允许建议使用内网地址
# 内网地址 url: https://${region}.cls.tencentyun.com/prometheus/${topicId}/api/v1/read
url: https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v1/read

# 鉴权信息，请替换其中的${SecretId}及${SecretKey}
basic_auth:
  username: ${SecretId}
  password: ${SecretKey}
```

url 中 ${region} 为 地域简称，${topicId} 为指标主题 ID。
接口采用 Basic Auth 进行鉴权，将 API 密钥 中的 SecretId 及 SecretKey 分别作为 username 及 password。

- username: `${SecretId}`
- password: `${SecretKey}`

> **说明：**
> 建议您单独创建一个子账号，使用该账号的 SecretId 及 SecretKey。该账号仅授予如下权限即可，以保障账号安全，配置方式详见 子账号授权 。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:MetricsRemoteRead"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

指标上报接口

Prometheus Remote Write API

常用于使用兼容 Prometheus Remote Write 协议的各类采集器采集指标并上报至指标主题，例如 vmagent 及 telegraf。也可将本地 Prometheus 中的指标通过该接口上报至指标主题。使用说明详见 指标上报 。

指标预聚合

最近更新时间：2025-06-27 15:38:52

指标监控数据量较大或者查询较为复杂时，实时查询可能较慢，查询语句编写也较为困难。为了解决这一问题，指标存储支持预聚合功能。预聚合支持预先计算复杂的查询（ PromQL ），并将结果存储为新的指标。这些预计算的数据可以在后续的查询和告警中直接使用，从而减少实时计算的开销，提高查询性能，并使监控配置更为简单和高效。预聚合兼容 Prometheus Recording Rule，可直接导入原生的 YAML 配置文件。
例如原始指标 `prometheus_http_requests_total` 记录了 prometheus 各个接口在不同响应状态码的请求次数：

```
# HELP prometheus_http_requests_total Counter of HTTP requests.
# TYPE prometheus_http_requests_total counter
prometheus_http_requests_total{code="200",handler="/api/v1/label/:name/values"} 7
prometheus_http_requests_total{code="200",handler="/api/v1/query"} 19
prometheus_http_requests_total{code="200",handler="/api/v1/query_range"} 27
prometheus_http_requests_total{code="200",handler="/graph"} 11
prometheus_http_requests_total{code="200",handler="/metrics"} 8929
prometheus_http_requests_total{code="200",handler="/static/*filepath"} 52
prometheus_http_requests_total{code="302",handler="/"} 1
prometheus_http_requests_total{code="400",handler="/api/v1/query_range"} 6
```

如果日常需要查询各个状态码的请求总次数，可使用如下的配置创建预聚合任务：

```
groups:
  - name: example
    rules:
    - record: code:prometheus_http_requests_total:sum
      expr: sum by (code) (prometheus_http_requests_total)
```

其中：

- `sum by (code) (prometheus_http_requests_total)` 为指标查询语句(PromQL)，按照状态码分组求和请求次数。
- `code:prometheus_http_requests_total:sum` 为生成的指标名称，可自定义，后续可直接使用该名称查询计算好的指标。

使用场景

- **提升指标查询性能**：当有一些复杂的查询需要频繁执行时，使用预聚合可以将这些查询预先计算好并存储为新的指标。这可以显著提升查询性能，缩短查询耗时。
- **简化查询语句**：复杂的查询语句可能非常冗长且难以阅读，通过预聚合您可以将复杂的查询结果存储为简化的指标。这使得后续的查询更简单、更容易维护。
- **单独存储核心指标**：指标主题中所有的数据按照相同的周期进行保存，过期后将自动清除。如需将其中少部分核心指标存储更久，可通过预聚合任务将这部分指标数据存储到其它指标主题中。

前提条件

已创建 指标主题 。

操作步骤

新建任务

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**指标主题**。
3. 单击需要新建预聚合任务的指标主题 ID/名称，进入指标主题管理页面。
4. 单击**指标预聚合**页签，进入预聚合任务列表页面，单击**新建任务**。其中主要配置项说明如下：

配置项	说明
配置名称	填写配置名称
服务日志	将任务运行日志保存在日志主题 cls_service_log 中，便于监控任务运行状态。该日志主题免费，默认开启。

执行语句	需要执行的 PromQL 语句，预聚合任务定时执行该语句以获取执行结果。
指标名称	将执行语句结果存储在该指标名称下，后续使用该名称查询数据。支持英文字母、数字、下划线及冒号，需符合正则表达式 `[a-zA-Z_:][a-zA-Z0-9_:]*` 。
自定义维度	为指标添加维度，自定义维度与执行语句查询结果中的维度名称冲突时，以自定义维度为准。
调度周期	预聚合任务的执行间隔，支持1~1440分钟，建议使用1分钟。
高级配置	• 目标指标主题：预聚合获得的指标数据存储到什么地方，默认为当前主题。如需将该部分数据单独存储（例如需单独设定这部分数据的保存时长），可将数据存储到其它指标主题中。 • 延迟执行：指标数据采集可能存在延迟，为了确保预聚合任务执行时数据已完整采集，可设定延迟执行，默认为30秒。

5. 配置完成后单击**提交**。

导入配置文件

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**指标主题**。
3. 单击需要新建预聚合任务的指标主题 ID/名称，进入指标主题管理页面。
4. 单击**指标预聚合**页签，进入预聚合任务列表页面，单击**导入配置文件**。其中主要配置项说明如下：

配置项	说明
配置名称	填写配置名称
启用状态	任务是否需要运行，不运行的任务不会产生预聚合结果数据。
服务日志	将任务运行日志保存在日志主题 cls_service_log 中，便于监控任务运行状态。该日志主题免费，默认开启。
执行间隔	预聚合任务的执行间隔，支持1~1440分钟，建议使用1分钟。

YAML 配置	兼容 Prometheus Recording Rule YAML，interval 仅支持1~1440分钟。例如：
	`groups:`
	`  - name: example`
	`  rules:`
	`  - record: code:prometheus_http_requests_total:sum`
	`  expr: sum by (code) (prometheus_http_requests_total)`

高级配置	• 目标指标主题：预聚合获得的指标数据存储到什么地方，默认为当前主题。如需将该部分数据单独存储（例如需单独设定保存时长），可将数据存储到其它指标主题中。 • 延迟执行：指标数据采集可能存在延迟，为了确保预聚合任务执行时数据已完整采集，可设定延迟执行，默认为30秒。

5. 单击**提交**，保存配置文件，将自动生成预聚合任务，任务名称由 YAML 中的 group name 及 record 组成。

> **说明：**
> 通过配置文件自动生成的预聚合任务需通过配置文件进行修改，不支持直接修改。

管理配置文件

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**指标主题**。
3. 单击需要新建预聚合任务的指标主题 ID/名称，进入指标主题管理页面。
4. 单击**指标预聚合**页签，进入预聚合任务列表页面，单击**管理配置文件**。
5. 在列表中可查看当前已有的配置文件，在**操作**列可管理配置文件：
   - 修改：修改配置文件，配置项与 导入配置文件 一致。修改后系统将根据最新的配置文件自动调整预聚合任务（删除配置文件中已不存在的任务、增加新的任务、更新已有任务）。

- 暂停：暂停配置文件对应的预聚合任务，暂停期间这些任务将不再执行。
- 删除：删除配置文件对应的预聚合任务。

检索分析（日志主题）

检索分析概述

最近更新时间：2025-06-25 19:05:52

基本概念

检索分析提供对日志数据进行过滤、搜索和统计分析的功能，例如查询包含 `error` 的日志、按 URL 分组统计日志条数、计算 PV 变化趋势等，是日志服务中最常用的功能。
检索分析基于 分词与索引，如果您对这两个概念较为陌生，建议您提前查阅该文档。

功能特性

基础功能

- 全文检索：针对日志全文进行检索，日志内任一字段满足检索条件时即可检索到该日志，例如使用 `error` 检索所有出现过 error 的日志。
- 键值检索：针对日志内指定的字段进行检索，指定字段满足检索条件时即可检索到该日志，例如使用 `level:error AND timeCost:>1000` 检索日志级别(level)为 error 且耗时(timeCost)大于 1000ms 的日志。
- 统计分析：针对符合检索条件的日志进行统计分析，返回统计分析结果，例如使用 `status:404 | select count(*) as logCounts` 统计响应状态码为404的日志数量。

上述功能需使用检索分析语句，使用方法详见 语法规则，也可使用 AI 智能编写查询语句 。

> **注意：**
> 低频存储 仅支持全文检索，不支持键值检索及统计分析。

高级功能

- 日志检索
  - 多主题检索：同时检索多个日志主题，暂不支持统计分析。可跨日志集，但不可跨地域。
  - 上下文检索：把目标日志数据在原始文件中的前若干条日志（上文）或后若干条日志（下文）检索查询出来。通过查看指定日志的上下文信息，您可以在业务故障排查中快速查找故障信息，方便定位问题。
  - 自定义跳转：单击日志字段值可进行自定义跳转，方便进行进一步的操作，例如根据 user_id 在内部用户管理平台查询用户信息。
  - 下载日志：将符合检索条件的日志下载至本地，单次可最多下载5000万条日志。
- 日志统计分析
  - 快速分析：点击日志字段名称，可快速分析字段值的分布、随时间变化趋势和数值统计。
  - 采样分析：使用统计分析功能时，若原始日志条数非常多或统计分析语句（SQL）较为复杂，统计分析可能出现缓慢甚至超时的情况。此时可使用采样分析功能对原始日志进行随机采样，再进行统计分析。
  - 关联外部数据：对日志进行统计分析时，可将日志与外部数据关联起来进行统计，例如根据日志中的用户 ID 在用户信息数据库中查询用户性别，然后按性别对日志进行统计分析。

产品优势

- 基于提前构建的索引数据，检索分析性能远高于一般的文本扫描方式的查询（例如 Linux grep 命令、文本编辑器中的查询功能），适用于海量日志数据的实时查询。
- 统计分析支持 SQL，并符合 SQL-92规范，可使用灵活的 SQL 语法及400+ SQL 函数对日志进行统计，满足绝大多数统计分析需求。

费用说明

检索分析本身不产生费用，但由于其依赖日志上报至日志服务平台并开启索引，会产生日志及索引流量费用、存储费用和主题分区费用，详见 计费概述 。另外，使用 API 调用检索分析接口时会产生服务请求次数费用，使用下载日志功能时会产生外网读流量费用。
如需降低产品使用费用，可参见 成本优化 。

规格与限制

检索分析的规格与限制，请参见 规格与限制 > 检索分析 。

版权所有：腾讯云计算（北京）有限责任公司

智能编写查询语句

最近更新时间：2025-05-21 10:24:22

操作场景

日志检索分析需编写符合 语法规则 的查询语句，较为复杂。智能编写功能可根据自然语言自动编写日志查询语句，提升日志查询效率。

- 智能编写：根据自然语言描述，自动编写查询语句。此外，还支持如下使用方式：
  - 解释查询语句：解释指定的查询语句含义，常用于 SQL 语句解释。
  - 探索更多日志分析角度：针对指定日志主题自动推荐日志分析角度及对应语句。
- 智能诊断：查询语句存在语法错误时，智能诊断并纠正错误语句。

> **注意：**
> 查询语句由 AI 大模型生成，仅供参考，建议您阅读其中的解释说明判断其是否满足查询需求。

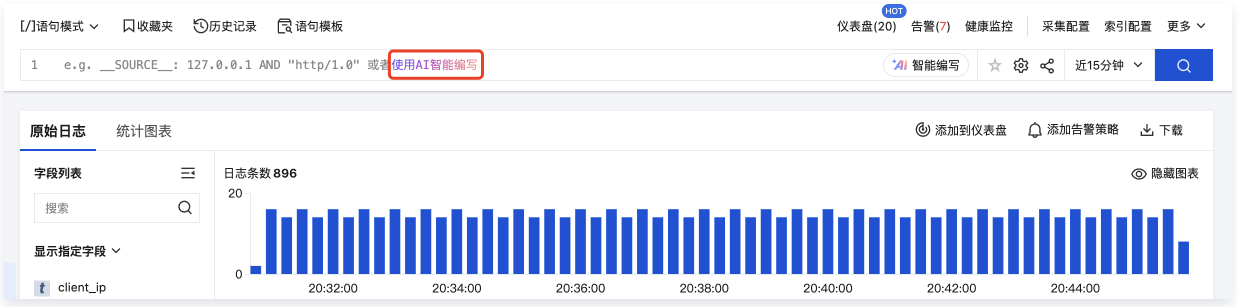
前提条件

- 日志主题需为标准存储，低频存储暂不支持。
- 日志主题已开启 索引 。

操作步骤

智能编写

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**检索分析**，进入检索分析页面。
3. 选择需要检索的日志主题。
4. 在查询语句输入框中单击**使用 AI 智能编写**。



5. 在对话框中输入查询需求。



也可填写其它问题，例如解释 SQL 语句含义或探索更多日志分析角度。

- 解释 SQL 语句含义。



- 探索更多日志分析角度。



智能诊断

1. 在检索分析页面出现查询语句报错时，单击**智能诊断**。



2. 诊断结果中将展示错误原因及修正后的查询语句，单击**执行**使用该语句替换报错语句。


语句模式
收藏夹
历史记录
语句模板
仪表盘(20)
告警(7)
健康监控
采集配置
索引配置
更多
1 * | SELECT count(*) AS pv,http_code limit 1000
智能编写
精确分析
近15分钟
原始日志
统计图表
存为定时SQL分析
添加到仪表盘
添加告警策略
导出分析数据
查询语句运行失败 重试
错误详情
QueryError [line 1:27:'http_code' must be an aggregate expression or appear in GROUP BY clause] RequestId:[f66c1d83-e4c7-40e8-bdb1-815d10bbe2e2]
诊断结果
根据您的需求，以下是修正后的CQL语句及解释说明：
复制 执行
* | SELECT count(*) AS pv, http_code GROUP BY http_code LIMIT 1000
解释说明：
1. 错误原因：原始语句中 http_code 字段未包含在 GROUP BY 子句中，导致报错。在聚合查询中，所有非聚合字段必须出现在 GROUP BY 子句中。
2. 修正内容：将 http_code 添加到 GROUP BY 子句中，确保查询语法正确。
3. 功能说明：该语句会统计每个 http_code 的日志条数（pv），并限制返回结果为前 1000 条记录。
校验结果通过，无语法错误。
图表配置
图表推荐
图表类型
表格
时序图
柱状图
单值图
计量仪
饼图
桑基图
地图
词云
漏斗图
热力图
雷达图
拓扑图
基础信息
图表名称
统计分析字段
图例
图例展示

语法规则

最近更新时间：2025-05-21 10:24:22

操作场景

检索分析语句由检索条件和 SQL 语句组成，两者通过竖线 | 分割，只需要检索日志，不需要统计分析时，可省略其中的管道符 | 及 SQL 语句。

```
[检索条件] | [SQL 语句]
```

- 检索条件：指定日志需要匹配的条件，返回符合该条件的日志。例如使用 `status:404` 检索响应状态码为404的应用请求日志。检索条件为空或 `*` 时代表无检索条件，即所有日志。
- SQL 语句：针对符合检索条件的日志进行统计分析，返回统计分析结果。例如使用 `status:404 | select count(*) as logCounts` 统计响应状态码为404的日志数量。

> **注意：**
> 检索基于日志分词，原始日志需在分词后包含检索条件指定的“词”才能被检索到，例如通过 error 检索不到 errorMessage，因为 errorMessage 本身是一个单独的词，不等于 error，需使用通配符 error*才能检索该日志。关于分词的更多说明及示例参见 分词与索引 。

前提条件

使用检索条件

- 全文检索：配置索引 时开启了全文索引。
- 键值检索：
  - 日志接入标准存储，低频存储不支持使用键值检索，详见 日志存储概述 。
  - 采集日志时按字段对日志进行了提取，即 日志结构化 。
  - 配置索引 时为需要检索的字段开启了键值索引。

使用 SQL 语句

- 日志接入标准存储，低频存储不支持使用 SQL 进行统计分析，详见 日志存储概述 。
- 采集日志时按字段对日志进行了提取，即 日志结构化 。
- 配置索引 时为需要检索的字段开启了键值索引且开启统计。

检索条件语法

检索条件支持两种语法规则：

- CQL：CLS Query Language，日志服务 CLS 专用检索语法，专为日志检索设计，使用容易，推荐使用。
- Lucene：开源 Lucene 语法，由于该语法并非专为日志检索设计，对特殊符号、大小写、通配符等有较多限制，使用较为繁琐，容易出现语法错误，不推荐使用。

CQL（推荐）

语法规则

语法	说明
key:value	键值检索，查询字段（key）的值中包含 value 的日志，例如：`level:ERROR`
value	全文检索，查询日志全文中包含 value 的日志，例如：`ERROR`
AND	“与”逻辑操作符，不区分大小写，例如：`level:ERROR AND pid:1234`
OR	“或”逻辑操作符，不区分大小写，例如：`level:ERROR OR level:WARNING` `level:(ERROR OR WARNING)`
NOT	“非”逻辑操作符，不区分大小写，例如：`level:ERROR NOT pid:1234` `level:ERROR AND NOT pid:1234`
()	逻辑分组操作符，控制逻辑运算优先级，例如：`level:(ERROR OR WARNING) AND pid:1234`

	⚠ 注意： 未使用括号时，`AND` 优先级高于 `OR` 。
" "	短语检索，使用双引号包裹一个字符串，日志需包含字符串内的各个词，且各个词的顺序保持不变，例如：`name:"john Smith"` 短语检索中不存在逻辑操作符，其等同于查询字符本身，例如：`name:"and"`
' '	短语检索，使用单引号包裹一个字符串，功能等价于 `""` ，当被检索短语中包含双引号时，可使用单引号包裹该短语，以避免语法错误，例如：`body:'user_name:"bob"'`
*	模糊检索，匹配零个、单个、多个字符，例如：`host:www.test*.com` ，不支持前缀模糊检索
>	范围操作符，表示大于某个数值，例如：`status>400` 或 `status:>400`
>=	范围操作符，表示大于等于某个数值，例如：`status>=400` 或 `status:>=400`
<	范围操作符，表示小于某个数值，例如：`status<400` 或 `status:<400`
<=	范围操作符，表示小于等于某个数值，例如：`status<=400` 或 `status:<=400`
=	范围操作符，表示等于某个数值，例如：`status=400` ，等价于 `status:400`
\	转义符号，转义后的字符表示符号本身。 • 被检索的值包含空格、`:` 、`(` 、`)` 、`>` 、`=` 、`<` 、`"` 、`'` 、`*` 时，需进行转义，例如：`body:user_name\:bob` • 使用双引号进行短语检索时，仅需转义 `"` 及 `*` • 使用单引号进行短语检索时，仅需转义 `'` 及 `*` • 未转义的 `*` 代表模糊检索
key:*	text 类型字段：查询字段（key）存在的日志，无论值是否为空，例如：`url:*` long/double 类型字段：查询字段（key）存在，且值为数值的日志，例如：`response_time:*`
key:""	text 类型字段：查询字段（key）存在且值为空的日志，值仅包含分词符时也等价为空，例如：`url:""` long/double 类型字段：查询字段值不为数值的日志，包含字段(key)不存在的情况，例如：`response_time:""`

示例	语句
检索来源为某台机器的日志	`__SOURCE__:127.0.0.1` 或 `__SOURCE__:192.168.0.*`
检索来源为某个文件的日志	`__FILENAME__:"/var/log/access.log"`
检索包含 `ERROR` 的日志	`ERROR`
检索失败的日志（状态码大于400）	`status>400`
检索 `GET` 请求中失败（状态码大于400）的日志	`method:GET AND status>400`
检索 `ERROR` 或 `WARNING` 级别的日志	`level:(ERROR OR WARNING)`
检索非 `INFO` 级别 的日志	`NOT level:INFO`

短语检索

使用双引号或单引号包裹一个字符串进行检索，例如 `name:"john Smith"` 、`filepath:"/var/log/access.log"` ，与不使用引号包裹的检索相比，短语检索表示日志需在包含字符串内各个词的同时，词之间的顺序也与检索条件严格一致。

例如有如下两条日志，分词符为 `/` ：

```
#1 filepath:"/var/log/access.log"
#2 filepath:"/log/var/access.log"
```

- 使用 `filepath:/var/log/access.log` 进行检索时会同时检索到上述两条日志，因为非短语检索不要求词之间的顺序。

- 使用 `filepath:"/var/log/access.log"` 进行检索时仅会检索到第一条日志。

短语检索的检索条件更加严格，检索较长的字符串时，建议使用短语检索。

> **注意**
> - 短语检索内支持使用通配符，例如 `filepath:"/var/log/acc*.log"` ，但不支持在词的开头使用通配符，例如 `filepath:"/var/log/*cess.log"` 。
> - 短语检索中的通配符仅能匹配到符合条件的128个词，返回包含这128个词的所有日志，指定的词越精确，查询结果越精确，非短语检索无该限制。

模糊检索

模糊检索需要在词的中间或末尾加上通配符，可使用 `*` 匹配零个、单个、多个字符，例如：

- `IP:192.168.1.*` 可检索到 `192.168.1.1` 、 `192.168.1.34` 等。
- `host:www.te*t.com` 可检索到 `www.test.com` 、 `www.telt.com` 等。

> **注意**
> - `*` 不能用在词的开头，即不支持前缀模糊检索。
> - long 和 double 类型字段不支持使用 `*` 进行模糊检索，可以使用数值范围进行检索，例如： `status>400 and status<500` 。

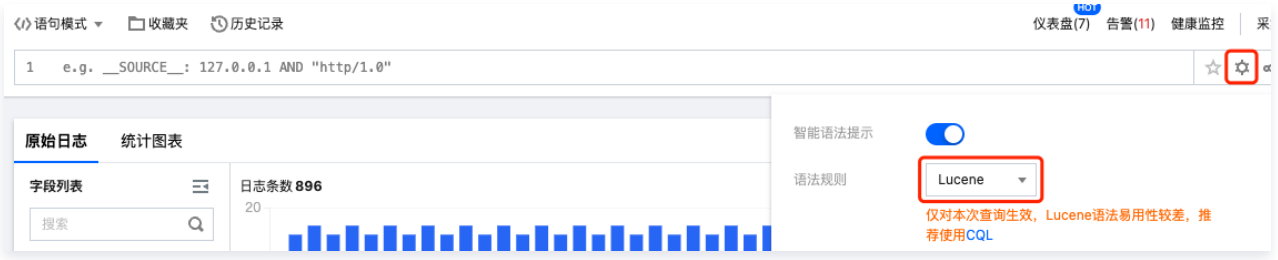
如果您需要使用前缀模糊查询，可使用如下方法替代：

- 添加分词符：例如日志为 `host:www.test.com` 、 `host:m.test.com` ，需要查询字段中间包含 test 的日志，可为该字段添加分词符 `.` ，便可以直接使用 `host:test` 对日志进行检索。
- 使用 SQL中的 strpos 函数：例如 `* | select * where strpos(host,'test')>0` ，但这种方式相比检索条件性能较差，不适合日志数据量过大的场景。

通配符也可使用在短语检索内，例如 `filepath:"/var/log/acc*.log"` ，但同样不支持在词的开头使用通配符，例如 `filepath:"/var/log/*cess.log"` 。且短语检索中的通配符仅能匹配到符合条件的128个词，返回包含这128个词的所有日志，指定的词越精确，查询结果越精确，非短语检索无该限制。

Lucene

如需使用 Lucene 语法，请在此处切换语法规则：



语法规则

语法	说明
AND	“与”逻辑操作符，例如 `level:ERROR AND pid:1234`
OR	“或”逻辑操作符，例如 `level:ERROR OR level:WARNING`
NOT	“非”逻辑操作符，例如 `level:ERROR NOT pid:1234`
()	分组操作符，控制逻辑运算优先级，例如 `(ERROR OR WARNING) AND pid:1234`
:	冒号，表示作用于的 key 字段，即键值检索，例如 `level:ERROR`
""	双引号，引用一个短语，日志需包含短语内的各个词，且各个词的顺序保持不变，例如 `name:"john Smith"`
*	通配符查询，匹配零个、单个、多个字符，例如 `host:www.test*.com` ，不支持前缀模糊查询

	还可以通过 `key:*` 的方式查询字段（key）存在的日志，等价于 `_exists_:key`
?	通配符查询，匹配单个字符，例如 `host:www.te?t.com` 与 `*` 类似，不支持前缀模糊查询
>	范围操作符，表示大于某个数值，例如 `status:>400`
>=	范围操作符，表示大于等于某个数值，例如 `status:>=400`
<	范围操作符，表示小于某个数值，例如 `status:<400`
<=	范围操作符，表示小于等于某个数值，例如 `status:<=400`
TO	“范围”逻辑操作符，例如 `request_time:[0.1 TO 1.0]`
[]	范围操作符，包含边界值的范围，例如 `age:[20 TO 30]`
{}	范围操作符，不包含边界值的范围，例如 `age:{20 TO 30}`
\	转义符号，转义后的字符表示符号本身，例如 `url:\/images\/favicon.ico` 如不想使用转义符，可使用 `""` 包裹，例如 `url:"/images/favicon.ico"`，但需注意，双引号内的词会被当作一个短语，日志需包含短语内的各个词，且各个词的顺序保持不变
exists	_exists_:key，返回 key 存在的日志，例如 `_exists_:userAgent` 表示搜索存在 `userAgent` 字段的日志

注意

- 语法区分大小写，例如 AND、OR 表示检索逻辑操作符，而 and、or 视为普通文本。
- 多个检索条件使用空格连接时，视为“或”逻辑，例如 `warning error` 等价于 `warning OR error`。
- 检索关键字中存在特殊字符时，需使用转义符进行转义，特殊字符包括 `+ - && || ! ( ) { } [ ] ^ " ~ * ? : \`。
- 同时使用 AND 和 OR 逻辑运算符时，请使用 `()` 对检索条件进行分组，以明确逻辑优先级，例如 `(ERROR OR WARNING) AND pid:1234`。

示例	语句
检索来源为某台机器的日志	`__SOURCE__:127.0.0.1` 或 `__SOURCE__:192.168.0.*`
检索来源为某个文件的日志	`__FILENAME__:"/var/log/access.log"` 或 `__FILENAME__:\/var\/log\/*.log`
检索包含 `ERROR` 的日志	`ERROR`
检索失败的日志（状态码大于400）	`status:>400`
检索 `GET` 请求中失败（状态码大于400）的日志	`method:GET AND status:>400`
检索 `ERROR` 或 `WARNING` 级别的日志	`level:ERROR OR level:WARNING`
检索非 `INFO` 级别 的日志	`NOT level:INFO`
检索 `192.168.10.10` 主机上非 `INFO` 级别的日志	`__SOURCE__:192.168.10.10 NOT level:INFO`
检索 `192.168.10.10` 主机上 `/var/log/access.log` 文件中不包含 `INFO` 级别的日志	`(__SOURCE__:192.168.10.10 AND __FILENAME__:"/var/log/access.log") NOT level:INFO`
检索 `192.168.10.10` 主机上 `ERROR` 或 `WARNING` 级别的日志	`__SOURCE__:192.168.10.10 AND (level:ERROR OR level:WARNING)`
检索 `4XX` 状态码的日志	`status:[400 TO 500}`
检索元数据中容器名为 `nginx` 的日志	`__TAG__.container_name:nginx`

检索元数据中容器名为 `nginx`，且请求延时大于1s 的日志	`__TAG__.container_name:nginx AND request_time:>1`
检索包含 `message` 字段的日志	`message:*` 或 `_exists_:message`
检索不包含 `message` 字段的日志	`NOT _exists_:message`

模糊检索

在日志服务中进行模糊检索，需要在词的中间或末尾加上通配符，可使用 `*` 匹配零个、单个、多个字符，或 `?` 匹配单个字符，例如：

- `IP:192.168.1.*` 可检索到 `192.168.1.1` 、`192.168.1.34` 等。
- `host:www.te*t.com` 可检索到 `www.test.com` 、`www.telt.com` 等。

> **注意**
> - `*` 或 `?` 不能用在词的开头，即不支持前缀模糊检索。
> - long 和 double 类型字段不支持使用 `*` 或 `?` 进行模糊检索，可以使用数值范围进行检索，例如 `status:[400 TO 500}` 。

如果您需要使用前缀模糊查询，可使用如下方法替代：

- 添加分词符：例如日志为 `host:www.test.com` 、`host:m.test.com` ，需要查询字段中间包含 test 的日志，可为该字段添加分词符 `.` ，便可以直接使用 `host:test` 对日志进行检索。
- 使用 SQL 中的 LIKE 语法：例如 `* | select * where host like '%test%'` ，但这种方式相比检索条件性能较差，不适合日志数据量过大的场景。

CQL 及 Lucene 对比

CQL 语法相比 Lucene 语法在易用性上进行了大量优化，并精简了部分不常用的功能，主要区别如下：

功能	Lucene	CQL
逻辑操作符	仅支持大写，例如 AND NOT OR	同时支持大小写，例如 AND and NOT not OR or
特殊符号转义	存在较多特殊符号需进行转义，例如检索 `/book/user/login/` 需转义为 `\/book\/user\/login\/`	需转义的特殊符号较少，可直接检索 `/book/user/login/`
关键词检索	关键词包含分词时，分词间的关系为或，例如分词符为 `/` 时，检索 `/book/user/login/` 等价于 `book OR user OR login` ，会检索到很多不相关的日志	关键词包含分词时，分词间的关系为与，例如分词符为 `/` 时，检索 `/book/user/login/` 等价于 `book AND user AND login` ，符合检索习惯
短语检索	检索短语时，不支持使用通配符，例如 `"/book/user/log*/"` 检索不到 `/book/user/login/` 及 `/book/user/logout/`	检索短语时，支持使用通配符，例如 `"/book/user/log*/"` 可以检索到 `/book/user/login/` 及 `/book/user/logout/`
数值范围检索	使用 `timeCost:[20 TO 30]` 形式的语法进行检索	使用 `timeCost>=20 AND timeCost<=30` 形式的语法进行检索
字段存在的日志	使用 `_exists_:key` 进行检索，其中key为字段名称	使用 `key:*` 进行检索，其中key为字段名称

SQL 语句语法

语法	说明
SELECT	从表中选取数据，默认从当前日志主题中获取符合检索条件的数据
AS	为列名称（KEY）指定别名
GROUP BY	结合聚合函数，根据一个或多个列（KEY）对原始数据进行分组聚合

ORDER BY	根据指定的 KEY 对结果集进行排序
LIMIT	限制结果集数据行数，默认限制为100，最大100万
WHERE	对查询到的原始数据进行过滤
HAVING	对分组聚合后的数据进行过滤，与 WHERE 的区别在于其作用于分组（GROUP BY）之后，排序（ORDER BY）之前，而 WHERE 作用于聚合前的原始数据
嵌套子查询	针对一些复杂的统计分析场景，需要先对原始数据进行一次统计分析，再针对该分析结果进行二次统计分析，这时候需要在一个 SELECT 语句中嵌套另一个 SELECT 语句，这种查询方式称为嵌套子查询
SQL 函数	使用 SQL 函数对日志进行更丰富的分析处理，例如从 IP 解析地理信息、时间格式转换、字符串分隔及连接、JSON 字段提取、算数运算、统计唯一值个数等。

注意：

- SQL 语句对大小写不敏感，`SELECT` 等效于 `select` 。
- 使用 `SELECT` 从日志主题中获取数据时，默认最大获取100行数据，如需获取更多数据请使用 LIMIT 语法 指定需要获取的行数，最多可获取100万行。
- 字符串必须使用单引号 `''` 包裹，无符号包裹或被双引号 `""` 包裹的字符表示字段或列名。例如 `'status'` 表示字符串 status，`status` 或 `"status"` 表示日志字段 status。字符串内本身包含单引号 `'` 时，需使用 `''` （两个单引号）代表单引号本身。例如 `'{''version'': ''1.0''}'` 表示原始字符串 `{'version': '1.0'}` 。字符串内本身包含双引号 `"` 时无需特殊处理。
- `SELECT` 中字段名称需符合 列名规范，不符合该规范时，需使用双引号 `""` 包裹。
- 不需要在末尾加分号表示 SQL 结束。

示例	语句
统计 `GET` 请求中失败（状态码大于400）的日志条数	`method:GET AND status:>400 | select count(*) as errorCount`
按分钟统计 `GET` 请求中失败（状态码大于400）的日志条数	`method:GET AND status:>400 | select histogram(__TIMESTAMP__, interval 1 minute) as analytic_time_minute, count(*) as errorCount group by analytic_time_minute limit 1000`
统计请求数量最大的5个 URL	`* | select URL, count(*) as log_count group by URL order by log_count desc limit 5`
统计 ERROR 日志占比	`* | select round((count_if(upper(Level) = 'ERROR'))*100.0/count(*),2) as "ERROR日志占比(%)"`
统计每个省份的请求数量	`* | select ip_to_province(client_ip) as province , count(*) as PV group by province order by PV desc limit 1000`

说明：
更多 SQL 示例请参见 SQL 案例 。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**检索分析**。
3. 在顶部选择需要查询的日志主题。
4. 选择检索分析语句输入模式，日志服务提供两种模式来输入检索分析语句。



- 交互模式：通过鼠标点击指定检索条件及统计分析规则，自动生成检索分析语句，易用性高。
- 语句模式：直接输入检索分析语句，需符合语法规则，灵活度高。

5. 输入检索分析语句后，在右侧选择**时间范围**，然后单击**搜索**按钮，执行检索分析。
   - 当检索分析语句仅包含检索条件时：可在**原始日志**中查看匹配检索条件的日志，默认按日志时间倒排。
   - 当检索分析语句包含 SQL 语句时：可在**统计图表**中查看分析结果，同时还可在**原始日志**中查看符合检索条件的日志，以便于对比分析统计结果及原始日志。

统计分析（SQL）
快速分析

最近更新时间：2024-07-11 16:43:41

操作场景

日志服务提供了针对字段的快速统计分析功能，无需编写查询语句即可快速分析字段值的分布、随时间变化趋势和数值统计，例如：统计 Top5 请求接口、接口响应时间变化趋势等。

前提条件

已在索引配置中开启了统计的字段。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**检索分析**，进入检索分析页面。
3. 选择需要检索的日志主题。
4. 在左侧字段列表中，单击**字段名称**。



> **说明：**
> 灰色字段不支持点击，是由于该字段未开启统计，需在索引配置中为该字段开启统计。

5. 在快速分析弹框中，单击某一数字，即可自动生成对应的检索分析语句及图表。

SQL 语法
SELECT 语法

最近更新时间：2024-08-15 09:57:01

用于从表中选取数据，默认从当前日志主题中获取符合检索条件的数据。

语法格式

```
* | SELECT [列名(KEY)] FROM log
```

SELECT默认从日志中获取数据，可省略 `FROM log` ，简写为 `* | SELECT [列名(KEY)]`

> **注意：**
> 从日志主题中获取数据时，默认最大获取100行数据，如需获取更多数据请使用 LIMIT 语法 指定需要获取的行数，最多可获取100万行。

语法示例

从日志数据中选取列（KEY）为 `remote_addr` 以及 `method` 的值：

```
* | SELECT remote_addr, method
```

从日志数据中选取所有列（KEY）：

```
* | SELECT *
```

SELECT 后面也可以跟算术表达式，如从日志数据中查询下载速度：
下载速度（ `speed` ）= 总发送字节数（ `body_bytes_sent` ）/ 请求时长（ `request_time` ）

```
* | SELECT body_bytes_sent / request_time AS speed
```

列名规范

在 SQL 规范中，列名须由字母、数字和下划线 `_` 组成，且以字母开头，例如 `remote_addr` 。日志中字段名称不符合该规范时，需使用双引号 `""` 包裹，也可以在 SQL 中使用 AS 语法 为该字段指定别名。字段名称与 SQL 语法关键词冲突时，也需使用双引号 `""` 包裹。
日志字段名为 `remote_addr` ，符合 SQL 列名规范，可直接使用 select 查询：

```
* | SELECT remote_addr
```

日志字段名为 `__TAG__.pod_label_qcloud-app` ，不符合 SQL 列名规范，需使用双引号 `""` 包裹：

```
* | SELECT "__TAG__.pod_label_qcloud-app"
```

日志字段名为 `__TIMESTAMP__` ，不符合 SQL 列名规范，需使用双引号 `""` 包裹，同时使用 AS 为该字段指定别名：

```
* | SELECT "__TIMESTAMP__" AS log_time
```

日志字段名为 `order` ，与 ORDER BY 语法 冲突，需使用双引号 `""` 包裹：

```
* | SELECT "order"
```

AS 语法

最近更新时间：2024-10-16 16:03:11

AS 子句用于为列名称（KEY）指定别名。

语法格式

```
* | SELECT 列名（KEY） AS  别名
```

语法样例

创建中文别名：

```
* | SELECT remote_addr AS "客户端IP", request_time AS "请求时间(单位：秒)"
```

> **注意**
> 如果别名中包含中文或其他特殊字符，需要使用双引号。

统计访问次数：

```
* | SELECT COUNT(*) AS PV
```

GROUP BY 语法

最近更新时间：2024-10-18 10:20:11

GROUP BY 语法用于结合 聚合函数，根据一个或多个列对分析结果进行分组统计。

语法格式

```
* | SELECT 列名, 聚合函数 GROUP BY [ 列名 | 别名 | 序号 ]
```

> **注意**
> 在 SQL 语句中，如果您使用了 GROUP BY 语法，则在执行 SELECT 语句时，只能选择 GROUP BY 的列或聚合计算函数，不允许选择非 GROUP BY 的列。例如 `* | SELECT status, request_time, COUNT(*) AS PV GROUP BY status` 为非法分析语句，因为 request_time 不是 GROUP BY 的列。

GROUP BY 语法支持按照列名、别名或序号进行分组，详细说明如下表所示：

参数	说明
列名	日志字段名称或聚合函数计算结果列，支持单列或多列
别名	按照日志字段或聚合函数计算结果的别名进行分组
序号	某列在 SELECT 语句中的序号（从1开始）。 例如 status 列的序号为1，所以下面两个语句为等同关系。 • `* | SELECT status, count(*) AS PV GROUP BY status` • `* | SELECT status, count(*) AS PV GROUP BY 1`
聚合函数	GROUP BY 语法常与 MIN、MAX、AVG、SUM、COUNT 等聚合函数搭配使用，更多信息请参见 聚合函数。

语法示例

- 统计不同状态码的访问次数：

```
* | SELECT status, count(*) AS pv GROUP BY status
```

- 按照每分钟的时间粒度，计算 PV 随时间变化的趋势：

```
* |
SELECT
  date_trunc(
    'minute',
    cast(__TIMESTAMP__ as timestamp)
  ) AS dt,
  count(*) AS pv
GROUP BY
  dt
ORDER BY
  dt
LIMIT
  10
```

`__TIMESTAMP__` 字段为日志服务中的保留字段，表示时间列。`dt` 为 `date_trunc('minute', cast(__TIMESTAMP__ as timestamp))` 的别名，表示将时间截断到分钟粒度。`date_trunc()` 函数的更多信息，请参见 时间截断函数。

> **说明**
> - limit 10表示最多获取10行结果。如果不使用 LIMIT 语法，则默认获取100行结果。

- 在索引配置中，当您开启任意字段的统计功能后，日志服务会自动开启 `__TIMESTAMP__` 字段的统计功能。

- 按照5分钟的时间粒度，计算 PV 及 UV 随时间变化的趋势：
  因为 `date_trunc()` 函数只能按照固定时间间隔统计，可以使用 histogram 函数进行自定义时间间隔统计。

```
* |
SELECT
  histogram(
    cast(__TIMESTAMP__ as timestamp),
    interval 5 minute
  ) as dt,
  count(*) as pv,
  count(
    distinct(remote_addr)
  ) as uv
GROUP BY
  dt
ORDER BY
  dt
```

ORDER BY 语法

最近更新时间：2024-10-16 16:03:11

ORDER BY 语法用于根据指定的列名对分析结果进行排序。

语法格式

```
ORDER BY 列名 [DESC | ASC]
```

说明

- 您可以指定多个列名，按照不同的排序方式排序。例如 ORDER BY 列名1[DESC | ASC]，列名2[DESC | ASC]。
- 如果您未配置关键字 DESC 或 ASC，则默认进行升序排列。
- 当排序的目标列中存在相同的值时，每次排序结果可能不同。如果您希望每次序列结果相同，可指定多列进行排序。

参数说明如下表所示：

参数	说明
列名	列名即为日志字段名称或聚合函数计算结果列
DESC	降序排列
ASC	升序排列

语法示例

- 统计不同请求状态的数量，并按照请求数量降序排列：

```
* |
SELECT
  status,
  count(*) AS pv
GROUP BY
  status
ORDER BY
  pv DESC
```

- 计算各服务器的平均请求时间，并按照请求时间进行升序排列：

```
* |
SELECT
  remote_addr,
  avg(request_time) AS request_time
GROUP BY
  remote_addr
ORDER BY
  request_time ASC
LIMIT
  10
```

LIMIT 语法

最近更新时间：2024-10-16 16:03:11

LIMIT 语法用于限制输出结果的行数。

语法格式

- 读取前 N 行：

```
limit N
```

- 从第 S 行开始读，读取 N 行：

```
offset S limit N
```

语法示例

- 获取10行结果：

```
* | select status, count(*) as pv group by status limit 10
```

- 获取第3行到第42行的结果，共计40行：

```
* | select status, count(*) as pv group by status offset 2 limit 40
```

限制说明

指标	限制说明	备注
SQL 结果条数	SQL 返回结果条数最大100万条	limit 默认为100

版权所有：腾讯云计算（北京）有限责任公司

WHERE 语法

最近更新时间：2024-08-15 09:57:01

WHERE 用于提取那些满足指定条件的日志。

语法格式

```
* | SELECT 列名（KEY） WHERE 列名（KEY） 运算符 值
```

运算符可以是 `=`、`<>`、`>`、`<`、`>=`、`<=`、`BETWEEN`、`IN`、`LIKE`。

注意

- SQL 中的过滤功能相比检索条件性能较差，建议尽可能地使用检索条件满足数据过滤需求，例如使用 `status:>400 | select count(*) as logCounts` 代替 `* | select count(*) as logCounts where status>400` 以更快地获得统计结果。
- WHERE 中不能使用 AS 别名，例如 `level:* | select level as log_level where log_level='ERROR'`，该语句执行会报错，因为其不符合 SQL-92规范。

语法示例

从日志数据中查询状态码大于400的日志：

```
* | SELECT *  WHERE status > 400
```

从日志数据中查询请求方式为 GET 且客户端 IP 为 192.168.10.101 的日志条数：

```
* | SELECT count(*) as count WHERE method='GET' and remote_addr='192.168.10.101'
```

统计 URL 后缀为 .mp4 的平均请求大小：

```
* | SELECT round(sum(body_bytes_sent) / count(body_bytes_sent), 2) AS avg_size  WHERE url like '%.mp4'
```

版权所有：腾讯云计算（北京）有限责任公司

HAVING 语法

最近更新时间：2024-10-16 16:03:11

HAVING 用于对分组聚合后的数据进行过滤，与 WHERE 的区别在于其作用于分组（GROUP BY）之后，排序（ORDER BY）之前，而 WHERE 作用于聚合前的原始数据。

语法格式

```
* | SELECT 列名, 聚合函数 GROUP BY [ 列名 | 别名 | 序号 ] HAVING 聚合函数 运算符 值
```

运算符可以是 `=`、`<>`、`>`、`<`、`>=`、`<=`、`BETWEEN`、`IN`、`LIKE`。

语法示例

统计平均响应耗时大于1000ms的 URL，并按耗时倒排：

```
* |
select
  avg(responseTime) as time_avg,
  URL
group by
  URL
having
  avg(responseTime)> 1000
order by
  avg(responseTime) desc
limit
  10000
```

由于过滤条件为各个 URL 的平均响应耗时，属于聚合后的结果，因此不能使用 WHERE 进行数据过滤。

版权所有：腾讯云计算（北京）有限责任公司

嵌套子查询

最近更新时间：2024-10-16 16:03:11

本文介绍嵌套子查询的语法使用与操作示例。

语法格式

针对一些复杂的统计分析场景，需要先对原始数据进行一次统计分析，再针对该分析结果进行二次统计分析，这时候需要在一个 SELECT 语句中嵌套另一个 SELECT 语句，这种查询方式称为嵌套子查询。

```
* | SELECT key FROM (subquery)
```

- subquery：子查询，需被包裹在括号中。
- key：需要从子查询中获取哪些字段进行二次统计分析。
- 支持多层嵌套，不局限于两层。

语法示例

计算当前1小时和昨天同时段的网站访问量比值：
选择查询和分析的时间范围为近1小时，并执行如下查询和分析语句，其中86400表示当前时间减去86400秒（1天）。

```
* | SELECT compare(PV, 86400) FROM (SELECT count(*) AS PV)
```

- `SELECT count(*) AS PV` 为第一层统计分析，针对原始日志统计获得网站访问量 PV。
- `SELECT compare(PV, 86400) FROM` 为第二层统计分析，针对第一层统计分析的结果，对 PV 进行二次统计，使用 同环比函数compare 获得1天前的网站访问量 PV。

结果为数组：

```
[1860,1656,1.1231884057971016]
```

- 1860表示当前1小时的网站访问量。
- 1656表示昨天同时段的网站访问量。
- 1.1231884057971016表示当前1小时与昨天同时段的网站访问量比值。

如需分别获取数组中的元素，则可进一步嵌套查询：

```
* |
SELECT compare[1] AS today, compare[2] AS yesterday, compare[3] AS ratio
FROM (
    SELECT compare(PV, 86400) AS compare
    FROM (
        SELECT COUNT(*) AS PV
    )
)
```

`SELECT compare[1] AS today, compare[2] AS yesterday, compare[3] AS ratio FROM` 针对 compare 函数的结果，通过数组下标获取其中特定位置的数值。

版权所有：腾讯云计算（北京）有限责任公司

JOIN 语法

最近更新时间：2025-07-03 15:06:02

JOIN 语法用于连接多个表，支持如下使用方式：

- 同日志主题内多个查询结果进行连接。
- 连接对象存储中的 CSV 文件、MySQL 数据库，详情请参见 关联外部数据 。
- 连接多个日志主题，详情请参见 跨主题关联分析 。

语法格式

```
* | SELECT [列名(KEY)] FROM table1 JOIN_TYPE table2 ON table1.key=table2.key
```

其中：

- 列名（KEY）在多个表名中相同时，必须指定表名或表的别名。
- JOIN_TYPE 支持如下类型（[ ]内容可省略）：
  - [INNER] JOIN：返回两个表同时存在的结果。
  - LEFT [OUTER] JOIN：即使右表（table2）中没有匹配的数据，也从左表（table1）返回所有 SELECT 结果。
  - RIGHT [OUTER] JOIN：即使左表（table1）中没有匹配的数据，也从右表（table2）返回所有 SELECT 结果。
  - FULL [OUTER] JOIN：只要一个表中存在匹配的数据，则返回满足条件的所有 SELECT 结果。
  - CROSS JOIN：无需指定 JOIN_CONDITION（即 `ON table1.key=table2.key` ），返回两个表的笛卡尔积（所有可能的组合）。
- JOIN_CONDITION，即 `ON table1.key=table2.key` 。也可同时指定过滤条件，例如 `ON (table1.key=table2.key and table2.key='xxx')` ，将会先按照指定条件过滤原始表中的数据，再进行连接查询。

语法示例

- LEFT JOIN：从日志数据（log）中查询 ip、url、user_id，并根据 user_id 在 userinfo 表中查询 Name、Gender。

```
* |
select
  log.ip,
  log.url,
  log.user_id,
  userinfo.Name,
  userinfo.Gender
from
  log
  left join userinfo on log.user_id = userinfo.id
limit
  10000
```

返回结果：

ip	url	user_id	Name	Gender
168[redacted]6	"GET /class/111.html HTTP/1.1"	3	Michael Johnson	Male
163[redacted]9	"GET /NoBUG/666 HTTP/1.1"	17	Logan Thompson	Male
125[redacted]7	"GET /class/114.html HTTP/1.1"	4	Sarah Davis	Female
86.[redacted]8	"GET /NoBUG/666 HTTP/1.1"	4	Sarah Davis	Female
19.[redacted]2	"GET /NoBUG/666 HTTP/1.1"	6	Emily Anderson	Female
125[redacted]6	"GET /NoBUG/666 HTTP/1.1"	23	Benjamin Rodriguez	Male
132[redacted]3	"GET /class/113.html HTTP/1.1"	17	Logan Thompson	Male
168[redacted]2	"GET /class/112.html HTTP/1.1"	6	Emily Anderson	Female

- INNER JOIN：从当前一个日志主题内，分别统计各个 URL 的错误日志条数 errorCount 和总日志条数 allCount，然后关联在一起。该实例同时演示了嵌套子查询与 JOIN 语法一起使用的方式。

```
* |
select
```

版权所有：腾讯云计算（北京）有限责任公司

```
  errorCount,
  allCount,
  table1.url as url
from
  (
    select
      count(*) as errorCount,
      url
    where
      status_code != 200
    group by
      url
  ) as table1
  inner join (
    select
      count(*) as allCount,
      url
    group by
      url
  ) as table2 on table1.url = table2.url
```

返回结果：

# errorCount	# allCount	t url
7	17	"GET /class/113.html HTTP/1.1"
14	16	"GET /class/111.html HTTP/1.1"
14	22	"GET /class/112.html HTTP/1.1"
11	19	"GET /class/114.html HTTP/1.1"
11	16	"GET /anyway/666 HTTP/1.1"
15	24	"GET /NoBUG/666 HTTP/1.1"
9	17	"GET /class/115.html HTTP/1.1"
12	19	"GET /class/116.html HTTP/1.1"

版权所有：腾讯云计算（北京）有限责任公司

UNION 语法

最近更新时间：2024-10-18 10:20:11

UNION 语法用于合并多个 SELECT 语句的分析结果。

语法格式

```
* | SELECT [列名(KEY)] FROM table1 UNION SELECT [列名(KEY)] FROM table2
```

- 其中每个 SELECT 语句的分析结果必须拥有相同数量的列和相同的字段类型，才能进行合并。
- SELECT 结果中包含重复行时，UNION 默认仅保留其中一行，如需全部保留，需使用 `UNION ALL` 。

语法示例

- 将两个只有一列的表合并在一起。其中：第一个表只有一行，值为13；第二个表有两行，值分别为42、13。**合并**其中的重复行。

```
* | SELECT * FROM (VALUES 13)
UNION
SELECT * FROM (VALUES 42,13)
```

返回结果：

_col0
42
13

- 将两个只有一列的表合并在一起。其中：第一个表只有一行，值为13；第二个表有两行，值分别为42、13。**不合并**其中的重复行。

```
* | SELECT * FROM (VALUES 13)
UNION ALL
SELECT * FROM (VALUES 42,13)
```

返回结果：

_col0
13
42
13

INTERSECT 语法

最近更新时间：2024-10-18 10:20:11

INTERSECT 语法用于组合两个 SELECT 语句的分析结果，仅返回两个结果集中共同存在的行。

语法格式

```
* | SELECT [列名(KEY)] FROM table1 INTERSECT SELECT [列名(KEY)] FROM table2
```

- 其中每个 SELECT 语句的分析结果必须拥有相同数量的列和相同的字段类型，才能进行组合。
- SELECT 结果中包含重复行时，INTERSECT 仅保留其中一行。

语法示例

将两个只有一列的表组合在一起，获取其中共同存在的行。其中：第一个表只有一行，值为13；第二个表有两行，值分别为42、13。

```
* | SELECT * FROM (VALUES 13)
INTERSECT
SELECT * FROM (VALUES 42,13)
```

返回结果：

_col0
13

EXCEPT 语法

最近更新时间：2024-10-18 10:20:11

EXCEPT 语法用于组合两个 SELECT 语句的分析结果，仅返回第一个结果集中存在但第二个结果集中不存在的行。

语法格式

```
* | SELECT [列名(KEY)] FROM table1 EXCEPT SELECT [列名(KEY)] FROM table2
```

- 其中每个 SELECT 语句的分析结果必须拥有相同数量的列和相同的字段类型，才能进行组合。
- SELECT 结果中包含重复行时，EXCEPT 仅保留其中一行。

语法示例

将两个只有一列的表组合在一起，获取表一中存在但表二中不存在的行。其中：第一个表有两行，值分别为42、13；第二个表只有一行，值为13。

```
* | SELECT * FROM (VALUES 42,13)
EXCEPT
SELECT * FROM (VALUES 13)
```

返回结果：

_col0
42

VALUES 语法

最近更新时间：2024-10-18 10:20:11

VALUES 语法用于构造数据，以便在 SQL 中临时插入少量数据。

语法格式

```
* | SELECT * FROM (VALUES (key1_value1, key2_value1...),(key1_value2, key2_value2...)...) AS
table_name(key1_name,key2_name...)
```

语法示例

构造一个包含 id 和 name 两列的表。

```
* | SELECT * FROM (VALUES (1, 'John'), (2, 'Jane'), (3, 'Alice')) AS user(id, name)
```

返回结果：

# id	t name
1	John
2	Jane
3	Alice

UNNEST 语法

最近更新时间：2024-07-11 17:43:21

UNNEST 可将 ARRAY 或 MAP 展开为多行数据。

语法格式

- 将 ARRAY 展开为单列的多行数据：

```
* | SELECT * FROM UNNEST(ARRAY['value1','value2',...]) AS table_name(key_name)
```

- 将 MAP 展开为多行数据：

```
* | SELECT * FROM UNNEST(map) AS table_name(key1_name, key2_name,...)
```

语法示例

- 将 ARRAY 展开为多行数据：：

```
* | SELECT * FROM UNNEST(ARRAY[1,2]) AS t(number)
```

返回结果：

number
1
2

- 将 MAP 展开为多行数据：

```
* | SELECT * FROM UNNEST(
        map_from_entries(
            ARRAY[
                ('SQL',1974),
                ('Java', 1995)
            ]
        )
) AS t(language, first_appeared_year)
```

返回结果：

language	first_appeared_year
SQL	1974
Java	1995

- 将日志内由逗号连接的响应时间字段拆分为多行数据：
  - 原始日志：

```
## 第一条原始日志
rsp_time: 0.004, 0.002
url: /demo/test1

## 第二条原始日志
rsp_time: 0.006, 0.008
```

```
url: /demo/test2
```

- 检索分析语句：

```
* | SELECT time,url
FROM (
  SELECT split(rsp_time,',') AS time_array,url
) t
CROSS JOIN UNNEST(time_array) AS t(time)
```

- 返回结果：

time	url
0.004	/demo/test1
0.002	/demo/test1
0.006	/demo/test2
0.008	/demo/test2

- 将日志内由逗号连接的响应时间及响应状态字段拆分为多行数据：
  - 原始日志：

```
## 第一条原始日志
rsp_time: 0.004, 0.002
rsp_status: 200, 502
url: /demo/test1

## 第二条原始日志
rsp_time: 0.006, 0.008
rsp_status: 500, 404
url: /demo/test2
```

- 检索分析语句：

```
* | SELECT time,status,url
FROM (
  SELECT zip(split(rsp_time,','),split(rsp_status,',')) as data_array,url
) t
CROSS JOIN UNNEST(data_array) AS t(time,status)
```

- 返回结果：

time	status	url
0.004	200	/demo/test1
0.002	502	/demo/test1
0.006	500	/demo/test2
0.008	404	/demo/test2

版权所有：腾讯云计算（北京）有限责任公司

SQL 函数
函数概览

最近更新时间：2025-05-16 10:47:52

本文列举 CLS 支持的各类 SQL 函数、运算符及表达式。

通用聚合函数

函数名称	说明
arbitrary(key)	随机返回目标列中一个非 NULL 的值。
avg(KEY)	计算 KEY 的算数平均值。
bitwise_and_agg(key)	返回目标列汇所有值按位与运算（AND）的结果。
bitwise_or_agg(key)	返回目标列汇所有值按位或运算（OR）的结果。
bool_and(boolean)	是否所有的日志都满足指定的条件，如果是则返回 TRUE，否则返回 FALSE。
bool_or(boolean)	是否有任意一条日志满足指定的条件，如果是则返回 TRUE，否则返回 FALSE。
checksum(key)	计算目标列的校验和值，返回结果为二进制字符串。
count(*)	统计所有行的行数。
count(1)	COUNT(1) 等同于 COUNT(*)，表示所有的行数。
count(KEY)	统计 KEY 列非 NULL 的行数。
count_if(boolean)	统计满足指定条件的行数。
every(boolean)	是否所有的日志都满足指定的条件，如果是则返回 TRUE，否则返回 FALSE。
geometric_mean(key)	计算 KEY 的几何平均数，KEY 不允许包含负数，否则结果为 NaN。
max(KEY)	查询 KEY 中最大值。
max_by(x,y)	返回 y 为最大值时对应的 x 值。
max_by(x,y,n)	返回最大的 n 个 y 值对应的 x 值，返回结果为 JSON 数组。
min(KEY)	查询 KEY 中最小值。
min_by(x,y)	返回 y 为最小值时对应的 x 值。
min_by(x,y,n)	返回最小的 n 个 y 值对应的 x 值。返回结果为 JSON 数组。
sum(KEY)	计算 KEY 的总和。

逻辑运算符

函数名称	说明
AND	只有左右运算数都是 TRUE 时，结果才为 TRUE。
OR	左右运算数任一个为 TRUE 时，结果为 TRUE。
NOT	右侧运算数为 FALSE 时，结果才为 TRUE。

算数及比较运算符

函数名称	说明

+	加法：运算符两边的值相加。
-	减法：运算符左边减去运算符右边。
*	乘法 - 把运算符两边的值相乘。
/	除法 - 运算符左边除以运算符右边。
%	取模 - 运算符左边除以运算符右边后得到的余数。
=	判断运算符两边的值是否相等，如果相等则条件为真。
!=	判断运算符两边的值是否相等，如果不相等则条件为真。
<>	判断运算符两边的值是否相等，如果不相等则条件为真。
>	判断运算符左边的值是否大于运算符右边的值，如果是则条件为真。
<	判断运算符左边的值是否小于运算符右边的值，如果是则条件为真。
>=	判断运算符左边的值是否大于等于运算符右边的值，如果是则条件为真。
<=	判断运算符左边的值是否小于等于运算符右边的值，如果是则条件为真。
IN	IN 运算符用于把某个值与一系列指定列表的值进行比较。
NOT IN	IN 运算符的对立面，用于把某个值与不在一系列指定列表的值进行比较。
BETWEEN AND	BETWEEN 运算符用于在给定最小值和最大值范围内的一系列值中搜索值。
LIKE	LIKE 运算符用于把某个值与使用通配符运算符的相似值进行比较。%代表零个、一个或者多个字；_代表单个数字或者字符。
IS NULL	NULL 运算符用于把某个值与 NULL 值进行比较，为空为真。
IS NOT NULL	NULL 运算符用于把某个值与 NULL 值进行比较，不为空为真。
DISTINCT	对比 x 和 y 是否相等。
LEAST	返回 x,y...中的最小值。
GREATEST	返回 x,y...中的最大值。
ALL	x 满足所有条件时，返回 true。
ANY / SOME	x 满足任意一个条件时，返回 true。

估算函数

函数名称	说明
approx_distinct(KEY)	估算 KEY 列的唯一值的个数，估算结果标准误差为2.3%。
approx_percentile(KEY,percentage)	对于 KEY 列进行正序排序，返回大约处于 percentage 位置的值。KEY 需为数值类型字段，percentage 取值范围为[0,1]。

同环比函数

函数名称	说明
compare(x,n)	对比当前时间周期内 x 的值与 n 秒之前时间周期内的值。
compare(x,n1,n2,n3...)	对比当前时间周期内 x 的值与 n1秒、n2秒、n3秒之前时间周期内的值。
compare(x,n,time)	对比当前时间周期内 x 随时间的变化趋势与 n 秒之前时间周期内的变化趋势，time 为 x 变化趋势对应的时间列。

版权所有：腾讯云计算（北京）有限责任公司

compare(x,n1,n2,n3...,time)	对比当前时间周期内 x 随时间的变化趋势与 n1秒、n2秒、n3秒之前时间周期内的变化趋势，time 为 x 变化趋势对应的时间列。

字符串函数

函数名称	说明
chr(number)	返回与输入参数指定的 ASCII 码位值匹配的字符。
codepoint(string)	把 ASCII 码形式的字段值转换为 BigInt 类型。
concat(KEY1,KEY2,KEY3,KEYN)	将多个字符串 KEY1,KEY2,KEY3,...,KEYN 拼接成一个字符串，与||连接符连接效果一致。 注意：任意字符串为 null 时，返回结果为 null，如需跳过 null，可使用 concat_ws。
concat_ws(split_string,KEY1,KEY2,KEY3,KEYN)	以 split_string 作为分隔符连接字符串 KEY1,KEY2,KEY3,...,KEYN，会跳过 KEY1,...,KEYN 中的 null 值。
concat_ws(split_string, array(varchar))	以 split_string 作为分隔符将数组内的元素连接为一个字符串。如果 split_string 为 null，则结果为 null，且会跳过数组中的 null 值。返回结果为 varchar 类型。 注意：此函数中 array(varchar) 参数为一个数组，不是字符串。
format(format,args1,args2,args3,argsN)	使用格式 format 对参数 args 进行格式化输出。
from_utf8(binary)	将二进制字符串转为 UTF-8格式的字符串。非法的 UTF-8字符会被替换为 U+FFFD。
from_utf8(binary, replace)	将二进制字符串转为 UTF-8格式的字符串。非法的 UTF-8字符会被替换为 replace。
hamming_distance(KEY1,KEY2)	返回 KEY1字符串与 KEY2字符串的汉明距离。注意两个字符串长度必须一致。
length(KEY)	计算字符串 KEY 的长度。
levenshtein_distance(KEY1,KEY2)	返回 KEY1字符串与 KEY2字符串的编辑距离。
lower(KEY)	将字符串 KEY 转换为小写形式。
lpad(KEY,size,padstring)	使用 padstring 对字符串 KEY 进行向左补全到 size 大小。如果 size 小于 KEY 的长度，则对字符串进行截断到 size 大小。size 必须为非负数，padstring 必须非空。
ltrim(KEY)	移除字符串 KEY 左侧的空白字符。
normalize(string)	转化 string 为 NFC 标准格式。
normalize(string, form)	转化 string 为 form 格式。该函数 form 参数需要使用关键字（NFD、NFC、NFKD、NFKC），而非字符串。
position(substring IN KEY)	返回 substring 在字符串 KEY 中的位置，从1开始。如果没找到，则返回0。
replace(KEY,substring)	从字符串 KEY 中移除所有 substring 字符串
replace(KEY,substring,replace)	将字符串 KEY 中所有 substring 替换为 replace。
reverse(KEY)	翻转字符串 KEY。
rpad(KEY,size,padstring)	使用 padstring 对字符串 KEY 进行向右补全到 size 大小。如果 size 小于 KEY 的长度，则对字符串进行截断到 size 大小。size 必须为非负数，padstring 必须非空。
rtrim(KEY)	移除字符串 KEY 右侧的空白字符。
split(KEY,delimiter)	使用指定的分隔符 delimiter 对字符串 KEY 进行分割，返回数组。
split(KEY,delimiter,limit)	使用指定的分隔符 delimiter 对字符串 KEY 进行分割，返回数组，数组长度最多为 limit。数组中最后一个元素始终为 key 中剩余的所有部分，limit 必须为正数。

版权所有：腾讯云计算（北京）有限责任公司

split_part(KEY,delimiter,index)	使用指定的分隔符 delimiter 对字符串 KEY 进行分割，返回数组中 index 位置的字符串。index 从1开始，如果 index 大于数组长度则返回 null。
strpos(KEY,substring)	返回 substring 在字符串 KEY 中的位置，从1开始。如果没找到，则返回0。
strpos(KEY,substring,instance)	返回字符串 KEY 经 substring 分割后的第 instance 个实例在 string 中的位置。如果 instance 是负数，则从尾到头数。位置结果从1开始，如果没找到，则返回0。
substr(KEY,start)	返回字符串中从 start 位置开始的剩余所有字符，位置从1开始算。如果 start 为负数，则是相对于尾部开始算。
substr(KEY,start,length)	返回字符串从 start 开始的最多 length 长度的子串，位置从1开始算。如果 start 为负数，则是相对于尾部开始算。
to_utf8(string)	将 string 转化为 UTF-8格式的二进制字符串 varbinary。
translate(KEY,from,to)	将 KEY 中出现在 from 中的字符全部替换为 to 中对应位置的字符。如果 from 包含重复字符，则只算第一个字符。如果 KEY 中没出现 from 中的字符，则 KEY 直接复制。如果 from 长度超过 to 的长度，则对应字符会被删除。
trim(KEY)	删除字符串中前后的空白字符。
upper(KEY)	将字符串 KEY 转换为大写形式。
word_stem(word)	返回 word 的英语词根。
word_stem(word,lang)	返回 word 的 lang 语言词根。

日期和时间函数

函数名称	说明
current_date	返回当前日期，例如2022-12-28。
current_time	返回当前时间，例如17:08:18。
current_timestamp	返回当前时间的时间戳（非 Unix 格式），例如2022-12-28T17:09:05.655+08:00[Asia/Shanghai]。
current_timezone()	返回当前时区，例如 Asia/Shanghai。
date_add(unit,value,timestamp)	在 timestamp 上加上 value 个时间单位（unit）。value为负数则为减法。unit 支持 millisecond、second、minute、hour、day、week、month、quarter、year。
date_diff(unit, timestamp1, timestamp2)	返回两个 timestamp 之间的时间差值。unit 支持 millisecond、second、minute、hour、day、week、month、quarter、year。
date_format(timestamp,format)	把 TIMESTAMP 类型的日期时间转化为 format 格式的字符串，format 说明参见函数详细说明。
date_parse(string,format)	把 format 格式的日期时间字符串 string 转化为 TIMESTAMP 类型，format 说明参见函数详细说明。
date_trunc(unit,timestamp)	将 timestamp 截断至 unit 单位，该函数常用于需要按照时间进行统计分析的场景。unit 支持unit、second、minute、hour、day、week、month、quarter、year。
extract(field FROM timestamp)	从 timestamp 中提取指定的时间部分（field），field 支持如下取值：year、quarter、month、week、day、day_of_month、day_of_week、dow、day_of_year、doy、year_of_week、yow、hour、minute、second。
from_iso8601_date(string)	把 ISO8601格式的日期字符串转化为 DATE 类型的日期表达式。
from_iso8601_timestamp(string)	把 ISO8601格式的日期时间字符串转化为具有时区的 Timestamp 类型的日期时间表达式。
from_unixtime(unixtime)	把 Unix 时间戳转化为 TIMESTAMP 类型的日期时间表达式。
from_unixtime(unixtime, zone)	把 Unix 时间戳转化为具有时区的 TIMESTAMP 类型的日期时间表达式。

版权所有：腾讯云计算（北京）有限责任公司

histogram(time_column,interval)	对时间列 time_column 按时间间隔 interval 分为多个组，并按该组对日志进行统计。time_column必须为毫秒的 LONG 类型 UNIX 时间戳或 TIMESTAMP 类型的日期时间表达式，interval 支持单位为 SECOND（秒）、MINUTE（分）、HOUR（小时）、DAY（天）。
human_readable_seconds	将秒转换为易读的时间间隔字符串，例如1 minute、36 seconds。
last_day_of_month(timestamp)	返回 timestamp 所属月份最后一天。
localtime	返回本地时间。
localtimestamp	返回本地的日期和时间。
now()	返回当前时间时间戳（非Unix格式）与 current_timestamp 函数同等用法，例如2022-12-28T17:09:05.655+08:00[Asia/Shanghai]。
parse_duration(string)	将字符串 string 转换为时间段表达式。
time_series(time_column,interval,format,padding)	对时间列 time_column 按时间间隔 interval 分为多个组，并按该组对日志进行统计，支持补全查询时间窗口内缺失的数据。time_column 必须为毫秒的 LONG 类型 UNIX 时间戳或 TIMESTAMP 类型的日期时间表达式，interval 支持单位为s（秒）、m（分）、h（小时）、d（天），format 为时间格式，推荐使用'%Y-%m-%dT%H:%i:%s+08:00'，padding 为数据补全方式，详见函数详细说明。
timezone_hour(timestamp)	返回 TIMESTAMP 所属时区的小时偏移量。
timezone_minute(timestamp)	返回 TIMESTAMP 所属时区的分钟偏移量。
to_iso8601(x)	将 DATE 类型或 TIMESTAMP 类型的日期和时间表达式转换为 ISO8601格式的日期和时间字符串。
to_milliseconds(interval)	以毫秒为单位返回间隔的时间值。
to_unixtime(timestamp)	把 TIMESTAMP 类型的日期时间表达式转化为 Unix 时间戳。

IP 函数

函数名称	说明
ip_version(ip)	获取 IP 地址版本，返回结果为 ipv4或 ipv6。
ip_prefix(ip,prefix_bits)	获取目标 IP 地址的前缀。返回结果为子网掩码格式 IP 地址。
ip_subnet_max(ip)	获取 IP 网段中的最大 IP 地址。返回结果为 IP 地址。
ip_subnet_min(ip)	获取 IP 网段中的最小 IP 地址。返回结果为 IP 地址。
ip_subnet_range(ip)	获取 IP 网段范围。返回结果为 Array 类型的 IP 地址。
ip_to_city(ip)	分析目标 IP 地址所属城市。
ip_to_city_code(ip)	分析目标 IP 地址所属城市的代码。返回结果为城市的行政区规划代码，暂不支持中国台湾及国外城市。
ip_to_city_geo(ip)	分析目标 IP 地址所属城市的经纬度。返回结果为城市的经纬度，暂不支持中国台湾及国外城市。
ip_to_country(ip)	分析目标 IP 地址所属国家或地区。
ip_to_country_code(ip)	分析目标 IP 地址所属国家或地区的代码。
ip_to_country_geo(ip)	分析目标 IP 地址所属国家或地区的经纬度。
ip_to_domain(ip)	判断目标 IP 地址是内网地址还是外网地址。内网则返回 intranet，外网则返回 internet，非法 IP 地址则返回 invalid。
ip_to_provider(ip)	分析目标 IP 地址对应的网络运营商。

版权所有：腾讯云计算（北京）有限责任公司

ip_to_province(ip)	分析目标 IP 地址所属省份。
ip_to_province_code(ip)	分析目标 IP 地址所属省份的代码。
ip_to_province_geo(ip)	分析目标 IP 地址所属省份的经纬度。
is_prefix_subnet_of(string,string)	判断目标网段是否为某网段的子网。
is_subnet_of(string,ip)	判断目标 IP 地址是否在某网段内。
ip_threat(ip)	根据 腾讯安全威胁情报中心 中的安全情报，判断指定 IP 是否为恶意 IP。
ip_threat_tags(ip)	可疑及恶意 IP 的 威胁标签，代表该 IP 存在的具体恶意行为，返回结果为数组。
ip_threat_detail(ip)	IP 威胁检测详情，返回结果为 JSON。

URL 函数

函数名称	说明
url_decode(KEY)	对 URL 进行解码。
url_encode(KEY)	对 KEY 进行 URL 转义编码。
url_extract_fragment(url)	提取出 URL 中的 fragment。
url_extract_host(url)	提取出 URL 中的 host。
url_extract_parameter(url, name)	提取出 URL 中的 query 对应的参数值。
url_extract_path(url)	提取出 URL 中的 path。
url_extract_port(url)	提取出 URL 中的端口。
url_extract_protocol(url)	提取出 URL 中的协议。
url_extract_query(url)	提取出 URL 中的 query。

JSON 函数

函数名称	说明
json_array_contains(x，value)	判断 JSON 数组中是否包含某个值。
json_array_get(x, index)	获取 JSON 数组中某个下标对应的元素。
json_array_length(x)	计算 JSON 数组中元素的数量。x 不是 JSON 数组时，返回 null。
json_extract(x,json_path)	使用 json_path 从 JSON 对象或 JSON 数组中提取一组 JSON 值（数组或对象），不支持需要遍历数组元素的 json_path 语法，例如 $.store.book[*].author。
json_extract_scalar(KEY,json_path)	使用 json_path 从JSON 对象或 JSON 数组中提取一组标量值（字符串、整数或布尔值），不支持需要遍历数组元素的 json_path 语法，例如 $.store.book[*].author。
json_format(x)	将 JSON 类型转化成字符串类型。
json_parse(x)	将字符串类型转化成 JSON 类型，判断是否符合 JSON 格式。
json_size(x, json_path)	计算 JSON 对象或 JSON 数组中元素的数量。

正则式函数

函数名称	说明
regexp_extract(KEY,regular_expression)	提取并返回目标字符串 KEY 中符合正则表达式的第一个子串。
regexp_extract(KEY,regular_expression,n)	提取目标字符串 KEY 中符合正则表达式的子串，然后返回与目标捕获组匹配的第一个子串。n 代表第 n 个捕获组，n 为从1开始的整数。
regexp_extract_all(KEY,regular_expression)	提取目标字符串 KEY 中符合正则表达式的子串，并返回所有子串的合集。
regexp_extract_all(KEY,regular_expression,n)	提取目标字符串 KEY 中符合正则表达式的子串，并返回与目标捕获组匹配的子串合集。n 代表第 n 个捕获组，n 为从1开始的整数。
regexp_like(KEY,regular_expression)	判断目标字符串 KEY 是否符合正则表达式。
regexp_replace(KEY, regular_expression, replace_string)	替换目标字符串 KEY 中符合正则表达式的子串，返回被替换后的字符串。
regexp_replace(KEY,regular_expression)	删除目标字符串 KEY 中符合正则表达式的子串。
regexp_split(KEY,regular_expression)	使用正则表达式分割目标字符串，返回被分割后的子串合集。

数组函数

函数名称	说明
array_agg(KEY)	以数组形式返回 KEY 中的所有值。
array_distinct(KEY)	删除数组中重复的元素，仅保留唯一值。
array_except(x, y)	计算两个数组的差集。
array_intersect(x, y)	计算两个数组的交集。
array_join(x, delimiter)	使用指定的连接符 delimiter 将数组中的元素拼接为一个字符串。如果数组中包含 null 元素，则 null 元素将被忽略。
array_join(x, delimiter, null_replacement)	使用指定的连接符将数组中的元素拼接为一个字符串。如果数组中包含 null 元素，则 null 元素将被替换为 null_replacement。
array_max(x)	获取数组中的最大值。
array_min(x)	获取数组中的最小值。
array_position(x, element)	获取指定元素的下标，下标从1开始。如果指定元素不存在，则返回0。
array_remove(x, element)	删除数组中指定的元素。
array_sort(x)	对数组元素进行升序排序。
array_union(x, y)	计算两个数组的并集。
cardinality(x)	计算数组中元素的个数。
contains(x, element)	判断数组中是否包含指定元素。如果包含，则返回 true。
element_at(x, y)	返回数组中的第 y 个元素，y 从1开始。
filter(x, lambda_expression)	和 Lambda 表达式结合，用于过滤数组中的元素。只返回满足 Lambda 表达式的元素。
flatten(x)	将二维数组转换为一维数组。
reduce(x,	根据 Lambda 表达式中的定义，对数组中的各个元素进行相加计算，然后返回计算结果。

lambda_expression)	
sequence(x, y)	通过指定的起始值 x 返回一个数组，其元素为 x~y 内一组连续且递增的值。
sequence(x, y, step)	通过指定的起始值 x 返回一个数组，其元素为 x~y 内一组连续且递增的值（间隔为 step）。
shuffle(x)	对数组元素进行随机排列。
slice(x, start, length)	返回数组 x 的子集，start 指定索引开始的位置，length 指定子集中元素的个数。
transform(x, lambda_expression)	将 Lambda 表达式应用到数组的每个元素中。
zip(x, y)	将多个数组合并为一个二维数组，且各个数组中下标相同的元素组成一个新的数组。
zip_with(x, y, lambda_expression)	根据 Lambda 表达式中的定义将两个数组合并为一个数组。

类型转换函数

函数名称	说明
cast(KEY as type)	转换字段 KEY 的数据类型，如果某个值转换失败，将终止整个查询，如需跳过失败请使用 try_cast。type 可选值为 bigint、varchar、double、boolean、timestamp、decimal、array 或 map。
try_cast(KEY as type)	转换字段 KEY 的数据类型，如果某个值转换失败，该值返回 null，并跳过该值继续处理 type 可选值为 bigint、varchar、double、boolean、timestamp、decimal、array 或 map。
typeof(x)	返回 x 的数据类型。

条件表达式

函数名称	说明
CASE WHEN condition1 THEN result1 END	通过条件判断，对数据进行归类。
COALESCE(expression1, expression2, expression3, expressionN)	获取多个表达式中的第一个非 NULL 值。
IF(condition, result1)	如果 condition 为 true，则返回 result1，否则返回 null。
IF(condition, result1, result2)	如果 condition 为 true，则返回 result1，否则返回 result2。
NULLIF(expression1, expression2)	比较两个表达式的值是否相等。如果相等，则返回 null，否则返回第一个表达式的值。
TRY(expression)	捕获异常信息，使得系统继续执行查询和分析操作。

数学计算函数

函数名称	说明
abs(number)	计算数字的绝对值。
acos(number)	计算数字的反余弦。
asin(number)	计算数字的反正弦。
atan(number)	计算数字的反正切。
atan2(number1,number2)	计算两个数字相除的结果的反正切。
cbrt(number)	计算数字的立方根。
ceiling(double)	向上取整数。
cos(number)	计算数字的余弦。

版权所有：腾讯云计算（北京）有限责任公司

cosh(number)	计算数字的双曲余弦。
cosine_similarity(x,y)	计算向量 x 和 y 之间的余弦相似度。
degrees(x)	将弧度转换为度。
e(number)	计算数字的自然对数。
exp(number)	返回自然对数的指数。
floor(double)	向下取整数。
from_base(string,radix)	根据 BASE 编码将字符串 string 转为 radix 进制的数字。
infinity()	返回正无穷的数值。
is_nan(KEY)	判断目标值 KEY是否为 NaN（Not a Number）。
ln(number)	计算数字的自然对数。
log(number,b)	计算以 b 为底的对数。
log10(number)	计算以10为底的对数。
log2(number)	计算以2为底的对数。
mod(number1,number2)	用于计算 x 与 y 相除的余数。
nan()	返回一个 NaN 值（Not a Number）。
pi()	返回包含14个小数位的 π 值。
pow(number,b)	计算数字的 b 次幂。
radians(x)	将度转换为弧度。
rand()	返回随机数。
random(0,n)	返回[0,n)之间的随机数。
round(double)	返回 double 四舍五入后的整数。
round(double,n)	保留 double 的 n 位小数。
sign(number)	返回number的符号，通过1、0、-1表示。
sin(number)	计算数字的正弦。
sqrt(number)	计算数字的平方根。
tan(number)	计算数字的正切。
tanh(number)	计算数字的双曲正切。
to_base(x,radix)	根据 BASE 编码将 x 转为 radix 进制的字符串。
truncate(double)	截断数字的小数部分。
width_bucket(number,bins)	使用数组（bins）指定 Bucket 的范围，返回 number 所属的 Bucket。
width_bucket(number,bound1,bound2,n)	将一段数值（bound1 - bound2）划分成大小相同的 n 个 Bucket，返回 number 所属的 Bucket。

数学统计函数

函数名称	说明
corr(key1,key2)	计算两列的相关度。计算结果范围为[0,1]。

covar_pop(key1,key2)	计算两列的总体协方差。
covar_samp(key1,key2)	计算两列的样本协方差。
regr_intercept(key1,key2)	返回输入值的线性回归截距。key1是依赖值，key2是独立值。
regr_slope(key1,key2)	返回输入值的线性回归斜率。key1是依赖值，key2是独立值。
stddev(key)	计算 key 列的样本标准差。与 stddev_samp 函数同义。
stddev_pop(key)	返回 key 列的总体标准差。
stddev_samp(key)	计算 key 列的样本标准差。
var_pop(key)	计算 key 列的总体方差。
var_samp(key)	计算 key 列的样本方差。
variance(key)	计算 key 列的样本方差，与 var_samp 函数同义。

二进制字符串函数

函数名称	说明
from_base64(string)	把 base64转换成二进制字符串。
from_base64url(string)	从 URL 安全的 base64转化成二进制字符串。
from_big_endian_64(binary)	把大端表示的二进制字符串转化成数字。
from_hex(string)	从十六进制转化成二进制。
md5(binary)	计算二进制字符串的 md5。
sha1(binary)	计算二进制字符串的 sha1。
sha256(binary)	计算二进制字符串的 sha256 hash。
sha512(binary)	计算二进制字符串的 sha512 hash。
to_base64(binary)	把二进制字符串转换成 base64。
to_base64url(binary)	把二进制字符串转化成 URL 安全的 base64。
to_big_endian_64(bigint)	把数字转化成大端表示的二进制。
to_hex(binary)	把二进制字符串转化成十六进制表示。
xxhash64(binary)	计算二进制字符串的 xxhash64。

位运算函数

函数名称	说明
bit_count(x, bits)	统计 x 的二进制表示中1的个数。
bitwise_and(x, y)	以二进制的形式对 x，y 进行与运算。
bitwise_not(x)	以二进制的形式对 x 的所有位进行取反运算。
bitwise_or(x, y)	以二进制形式对 x，y 进行或运算。
bitwise_xor(x, y)	以二进制形式对 x，y 进行异或运算。

Lambda 函数

Lambda 表达式需与函数一起使用，例如 filter 函数、reduce 函数、transform 函数、zip_with 函数，说明详见 Lambda 函数。

窗口函数

窗口函数用于对指定的多行数据进行计算并返回计算的结果，与 GROUP BY 的区别在于其仅会将计算结果附加到每一行数据上，而不会对行本身进行合并。函数说明详见 窗口函数 。

空间几何函数

空间几何函数用于在几何空间中构造空间实体并对空间关系进行判断、操作及计算，函数说明详见 空间几何函数 。

版权所有：腾讯云计算（北京）有限责任公司

通用聚合函数

最近更新时间：2024-07-11 16:43:41

本文介绍通用聚合函数的基本语法和示例。

函数说明

聚合函数是对一组值执行计算并返回计算结果的函数，日志服务支持如下聚合函数：

> **说明**
> 日志服务分析语句中，表示字符串的字符必须使用单引号（''）包裹，无符号包裹或被双引号（""）包裹的字符表示字段名或列名。例如：`'status'` 表示字符串 status，`status` 或 `"status"` 表示日志字段 status。

函数语句	说明	示例
arbitrary(KEY)	随机返回目标列中一个非 NULL 的值。	`* | SELECT arbitrary(request_method) AS request_method`
avg(KEY)	计算目标列的算数平均值。	`* | SELECT AVG(request_time)`
bitwise_and_agg(KEY)	返回目标列汇所有值按位与运算（AND）的结果。	`* | SELECT bitwise_and_agg(status)`
bitwise_or_agg(KEY)	返回目标列汇所有值按位或运算（OR）的结果。	`* | SELECT bitwise_or_agg(request_length)`
checksum(KEY)	计算目标列的校验和值，返回结果为二进制字符串。	`* | SELECT checksum(request_method) AS request_method`
count(*)	表示所有的行数。	`* | SELECT COUNT(*) WHERE http_status >200`
count(1)	COUNT(1) 等同于 COUNT(*)，表示所有的行数。	`* | SELECT COUNT(1)`
count(KEY)	计算某一 KEY 列非 NULL 的行数。	`* | SELECT COUNT(request_time) WHERE request_time >5.0`
count_if(boolean)	统计满足指定条件的日志条数。	`* | select count_if(returnCode>=400) as errorCounts`
geometric_mean(KEY)	计算 KEY 的几何平均数，KEY 不允许包含负数，否则结果为 NaN。	`* | SELECT geometric_mean(request_time) AS request_time`
max(KEY)	查询 KEY 中的最大值。	`* | SELECT MAX(request_time) AS max_request_time`
max_by(x,y)	返回 y 为最大值时对应的 x 值。	`* | SELECT MAX_BY(request_method, request_time) AS method`
max_by(x,y,n)	返回最大的 n 个 y 值对应的 x 值，返回结果为 JSON 数组。	`* | SELECT max_by(request_method, request_time, 3) AS method`
min(KEY)	查询 KEY 中最小值。	`* | SELECT MIN(request_time) AS min_request_time`
min_by(x,y)	返回 y 为最小值时对应的 x 值。	`* | SELECT min_by(request_method, request_time) AS method`
min_by(x,y,n)	返回最小的 n 个 y 值对应的 x 值。返回结果为 JSON 数组。	`* | SELECT min_by(request_method, request_time, 3) AS method`

sum(KEY)	计算 KEY 的总值。	`* | SELECT SUM(body_bytes_sent) AS sum_bytes`
bool_and(boolean)	是否所有的日志都满足指定的条件，如果是则返回 TRUE ，否则返回 FALSE。	`* | select bool_and(returnCode>=400)`
bool_or(boolean)	是否有任意一条日志满足指定的条件，如果是则返回 TRUE ，否则返回 FALSE。	`* | select bool_or(returnCode>=400)`
every(boolean)	等效于 bool_and(boolean)。	`* | select every(returnCode>=400)`

参数说明

参数	说明
KEY	索引配置-键值索引 中开启统计的日志字段名称，也可以是多个数值字段的四则运算结果或嵌套子查询的查询结果。
x	参数值为任意数据类型。
y	参数值为任意数据类型。
n	大于0的整数。

逻辑运算符

最近更新时间：2024-07-11 16:43:41

本文介绍逻辑运算的语法和示例。

逻辑运算符

运算符	描述	示例
AND	只有左右运算数都是 TRUE 时，结果才为 TRUE。	a AND b
OR	左右运算数任一个为 TRUE 时，结果为 TRUE。	a OR b
NOT	右侧运算数为 FALSE 时，结果才为 TRUE。	NOT a

与 NULL 相关的逻辑运算

a 和 b 分别取值 TRUE，FALSE 和 NULL 时的真值表如下：
AND，OR 的真值表

a	b	a AND b	a OR b
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	NULL	NULL	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL
NULL	TRUE	NULL	TRUE
NULL	FALSE	FALSE	NULL
NULL	NULL	NULL	NULL

NOT 的真值表

a	NOT a
TRUE	FALSE
FALSE	TRUE
NULL	NULL

版权所有：腾讯云计算（北京）有限责任公司

算术及比较运算符

最近更新时间：2024-07-11 16:43:41

运算符是一个保留字或字符，主要用于指定 SQL 语句中的条件，并在语句中连接多个条件。

- 算术运算符
- 比较运算符

算术运算符

算术运算符是用来处理四则运算的符号，是最简单最常用的符号，尤其是数字的处理，几乎都会使用到算术运算符号。
假设变量 a=1，变量 b=2，则：

运算符	描述	示例
+	加法：运算符两边的值相加	a + b
-	减法：运算符左边减去运算符右边	a - b
*	乘法：把运算符两边的值相乘	a * b
/	除法：运算符左边除以运算符右边	b / a
%	取模：运算符左边除以运算符右边后得到的余数	b % a

比较运算符

比较运算符用于判断值的大小关系，支持任何可比较的类型，例如：int、long、double 和 text 等。
假设变量 a=1，变量 b=2，则：

运算符	描述	示例
=	判断运算符两边的值是否相等，如果相等则条件为真。	a = b
!=	判断运算符两边的值是否相等，如果不相等则条件为真。	a != b
<>	判断运算符两边的值是否相等，如果不相等则条件为真。	a <> b
>	判断运算符左边的值是否大于运算符右边的值，如果是则条件为真。	a > b
<	判断运算符左边的值是否小于运算符右边的值，如果是则条件为真。	a < b
>=	判断运算符左边的值是否大于等于运算符右边的值，如果是则条件为真。	a >= b
<=	判断运算符左边的值是否小于等于运算符右边的值，如果是则条件为真。	a <= b
IN	IN 运算符用于把某个值与一系列指定列表的值进行比较。	status IN (200,206,404)
NOT IN	IN 运算符的对立面，用于把某个值与不在一系列指定列表的值进行比较。	status NOT IN (200,206,404)
BETWEEN AND	BETWEEN 运算符用于在给定最小值和最大值范围内的一系列值中搜索值。	status between 200 AND 400
LIKE	LIKE 运算符用于把某个值与使用通配符运算符的相似值进行比较。%代表零个、一个或者多个字；_代表单个数字或者字符。	url LIKE '%.mp4'
IS NULL	NULL 运算符用于把某个值与 NULL 值进行比较，为空为真。	status IS NULL
IS NOT NULL	NULL 运算符用于把某个值与 NULL 值进行比较，不为空为真。	status IS NOT NULL
DISTINCT	语法：`x IS DISTINCT FROM y` 或 `x IS NOT DISTINCT FROM y`。 用于对比 x 和 y 是否相等，与<>的差异在于 DISTINCT 可用于对 null 的比较，详见 <> 与 DISTINCT 运算符差异 。	NULL IS NOT DISTINCT FROM NULL
LEAST	语法：`LEAST(x, y...)` 返回 x,y...中的最小值。	LEAST(1,2,3)

GREATEST	语法：`GREATEST(x, y...)` 返回 x,y...中的最大值。	GREATEST(1,2,3)
ALL	语法：`x expression operator ALL ( subquery )` x 满足所有条件时，返回 true，operator 支持 `<`、`>`、`<=`、`>=`、`=`、`<>`、`!=` 。	示例1：21 < ALL (VALUES 19, 20, 21) 示例2：* | SELECT 200 = ALL(SELECT status)
ANY / SOME	语法：`x expression operator ANY ( subquery )` 或 `x expression operator SOME ( subquery )` x 满足任意一个条件时，返回 true，operator 支持 `<`、`>`、`<=`、`>=`、`=`、`<>`、`!=` 。	示例1：'hello' = ANY (VALUES 'hello', 'world') 示例2：* | SELECT 200 = ANY(SELECT status)

<> 与 DISTINCT 运算符差异：

x	y	x = y	x <> y	x IS DISTINCT FROM y	x IS NOT DISTINCT FROM y
1	1	true	false	false	true
1	2	false	true	true	false
1	null	null	null	true	false
null	null	null	null	false	true

版权所有：腾讯云计算（北京）有限责任公司

估算函数

最近更新时间：2024-10-24 20:59:22

本文介绍估算函数的基本语法及示例。

函数名称	函数语法	说明
approx_distinct 函数	approx_distinct(x)	估算 x 列的唯一值的个数。
approx_percentile函数	approx_percentile(x,percentage)	对于 x 列进行正序排序，返回大约处于 percentage 位置的值。
	approx_percentile(x,array[percentage01, percentage02...])	对于 x 列进行正序排序，返回大约处于 percentage01、percentage02 位置的值。

approx_distinct 函数

approx_distinct 函数用于估算字段值中不重复值的个数，估算结果标准误差为2.3%。

语法

```
approx_distinct(x)
```

参数说明

参数	说明
x	参数值为任意数据类型。

返回值类型

bigint 类型。

示例

使用 count 函数计算 PV，使用 approx_distinct 函数估算不重复的 client_ip 字段值作为 UV。

```
* | SELECT count(*) AS PV, approx_distinct(ip) AS UV
```

approx_percentile 函数

approx_percentile 函数用于对目标字段的值进行正序排列，返回大约处于 percentage 位置的数值。采用 T-Digest 算法进行估算，误差较低，能够满足绝大多数统计分析需求，如有需要，可使用 `* | select count_if(x<(select approx_percentile(x,percentage))),count(*)` 分别精确统计低于 percentage 的字段值个数和总数来验证该统计误差。

语法

- 返回处于 percentage 位置的数值，返回结果为 double 类型。

  ```
  approx_percentile(x, percentage)
  ```

- 返回处于 percentage01、percentage02 位置的数值，返回结果为 array 类型。

  ```
  approx_percentile(x, array[percentage01,percentage02...])
  ```

参数说明

参数	说明
x	参数值为 double 类型。

percentage	百分比值，取值范围为[0,1]。

返回值类型

double 类型或 array 类型。

示例

1. 对 **resTotalTime** 列进行排列后，返回大约处于50%位置的 **resTotalTime** 字段的值。

```
* | select approx_percentile(resTotalTime,0.5)
```

2. 对 **resTotalTime** 列进行排列后，返回处于20%、40%及60%位置的 **resTotalTime** 字段的值。

```
* | select approx_percentile(resTotalTime, array[0.2,0.4,0.6])
```

同环比函数

最近更新时间：2024-10-24 20:59:22

本文介绍同环比函数的基础语法和示例。

函数名称	语句	含义
compare 函数	compare(x,n)	对比当前时间周期内 x 的值与 n 秒之前时间周期内的值。
	compare(x,n1,n2,n3...)	对比当前时间周期内 x 的值与 n1 秒、n2 秒、n3 秒之前时间周期内的值。
	compare(x,n,time)	对比当前时间周期内 x 随时间的变化趋势与 n 秒之前时间周期内的变化趋势，time 为 x 变化趋势对应的时间列
	compare(x,n1,n2,n3..., time)	对比当前时间周期内 x 随时间的变化趋势与 n1 秒、n2 秒、n3 秒之前时间周期内的变化趋势，time 为 x 变化趋势对应的时间列

compare 函数

compare 函数用于对比当前时间周期内的计算结果与 n 秒之前时间周期内的计算结果。

语法

- 对比当前时间周期内 x 的值与 n 秒之前时间周期内的值。

```
compare(x, n)
```

- 对比当前时间周期内 x 的值与 n1 秒、n2 秒、n3 秒之前时间周期内的值。

```
compare(x, n1, n2, n3...)
```

- 对比当前时间周期内 x 随时间的变化趋势与 n 秒之前时间周期内的变化趋势，time 为 x 变化趋势对应的时间列。

```
compare(x,n,time)
```

- 对比当前时间周期内 x 随时间的变化趋势与 n1 秒、n2 秒、n3 秒之前时间周期内的变化趋势，time 为 x 变化趋势对应的时间列。

```
compare(x,n1,n2,n3...,time)
```

参数说明

参数	说明
x	参数值为 double 类型或 long 类型。
n	时间窗口，单位为秒。例如3600（1小时）、86400（1天）、604800（1周）、31622400（1年）。
time	对比 x 随时间变化趋势时需填写该参数，time 为 x 变化趋势对应的时间列，需为 timestamp 类型 。

返回值类型

JSON 数组。格式为 `[当前计算结果，n 秒前的计算结果，当前计算结果与 n 秒前计算结果的比值]` 。

示例

示例1：计算当前1小时和昨天同时段的网站访问量比值

选择查询和分析的时间范围为近1小时，并执行如下查询和分析语句，其中86400表示当前时间减去86400秒（1天）。

- 检索分析语句

```
* | SELECT compare(PV, 86400) FROM (SELECT count(*) AS PV)
```

- 查询结果

```
[1860,1656,1.1231884057971016]
```

  - 1860 表示当前1小时的网站访问量。
  - 1656 表示昨天同时段的网站访问量。
  - 1.1231884057971016 表示当前1小时与昨天同时段的网站访问量比值。
- 如需查询结果为分列显示，可修改语句为：

```
* |
SELECT compare[1] AS today, compare[2] AS yesterday, compare[3] AS ratio
FROM (
    SELECT compare(PV, 86400) AS compare
    FROM (
        SELECT COUNT(*) AS PV
    )
)
```

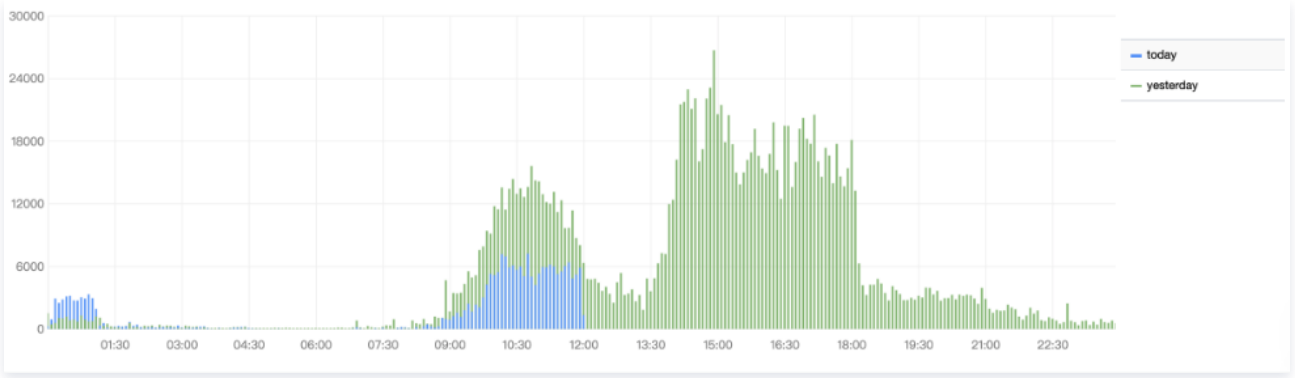
示例2：计算今天每5分钟网站访问量变化趋势与昨天同时间段的对比

选择查询和分析的时间范围为今天，并执行如下查询和分析语句，其中86400表示当前时间减去86400秒（1天）。

- 检索分析语句

```
* |
SELECT
  compare[1] AS today, compare[2] AS yesterday, time
FROM
  (
    SELECT
      compare(PV, 86400, time) AS compare, time
    FROM
      (
        SELECT
          count(*) AS PV, histogram(__TIMESTAMP__, INTERVAL 5 MINUTE) AS time
        GROUP BY
          time
      )
  )
```

- 查询结果



字符串函数

最近更新时间：2024-08-15 15:04:21

本文介绍字符串函数的基本语法和示例。

> **注意**
> - 字符串必须使用单引号 `''` 包裹，无符号包裹或被双引号 `""` 包裹的字符表示字段或列名。例如 `'status'` 表示字符串 status，`status` 或 `"status"` 表示日志字段 status。
> - 字符串内本身包含单引号 `'` 时，需使用 `''`（两个单引号）代表单引号本身。例如 `'{''version'': ''1.0''}'` 表示原始字符串{'version': '1.0'}。字符串内本身包含双引号 `"` 时无需特殊处理。
> - 如下函数中的 key 参数均表示日志字段名称。

函数说明

函数名称	说明	示例
chr(number)	返回与输入参数指定的 ASCII 码位值匹配的字符。返回结果为 varchar 类型。	返回第77位 ASCII 码中匹配的字符值。 `* | SELECT chr(77)`
codepoint(string)	把 ASCII 码形式的字段值转换为 BigInt 类型。返回结果为 integer 类型。	将 ASCII 码中的字符值转换为对应的位置。 `* | SELECT codepoint('M')`
concat(key1, ..., keyN)	连接字符串 key1，key2，...keyN，与||连接符连接效果一致。返回结果为 varchar 类型。注意：任意字符串为 null 时，返回结果为 null，如需跳过 null，可使用 concat_ws。	将多个字符串拼接成一个字符串。 `* | SELECT concat(remote_addr, host, time_local)`
concat_ws (split_string,key0, ..., keyN)	以 split_string 作为分隔符连接字符串 key1，key2，...keyN。split_string可以为字符串或变量，如果 split_string 为 null，则结果为 null，且会跳过 key1，key2，...keyN 中的 null 值。返回结果为 varchar 类型。	以/作为分隔符进行字符串之间的连接。 `* | SELECT concat_ws('/', remote_addr,host,time_local)`
concat_ws (split_string, array(varchar))	以 split_string 作为分隔符将数组内的元素连接为一个字符串。如果 split_string 为 null，则结果为 null，且会跳过数组中的 null 值。返回结果为 varchar 类型。 注意：此函数中 array(varchar)参数为一个数组，不是字符串。	以#作为分割符进行字符串之间的连接，此例中split函数的输出为array。 `* | select concat_ws('#',split('cloud.tencent.com/product/cls', '/'))`
format(format，args...)	使用格式 format 对参数 args 进行格式化输出。返回结果为 varchar 类型。	使用 format 格式 ‘IP 地址: %s, 域名: %s’ 对参数 remote_addr 和参数 host 进行格式化输出。 `* | SELECT format('IP地址: %s, 域名: %s', remote_addr, host)`
hamming_distance (key1, key2)	返回 key1 字符串与 key2 字符串的汉明距离。注意两个字符串长度必须一致。返回结果为 bigint 类型。	返回 remote_addr 字符串与 remote_addr 字符串的汉明距离。 `* | SELECT hamming_distance(remote_addr, remote_addr)`
length(key)	计算字符串的长度。返回结果为 bigint 类型。	返回 http_user_agent 的字符串长度。 `* | SELECT length(http_user_agent)`
levenshtein_distance (key1, key2)	返回字符串 key1 和 key2 的编辑距离。返回结果为 bigint 类型。	返回 remote_addr 字符串和 http_protocol 字符串的编辑距离。 `* | SELECT levenshtein_distance(remote_addr, http_protocol)`

lower(key)	将字符串转换为小写形式。返回结果为 varchar 类型，小写形式。	将 http_protocol 字符串转换为小写形式。 `* | SELECT lower(http_protocol)`
lpad(key, size, padstring)	使用 padstring 对字符串进行向左补全到 size 大小。如果 size 小于 key 的长度，则对字符串进行截断到 size 大小。size 必须为非负数，padstring 必须非空。返回结果为 varchar 类型。	使用 ‘0’ 对 remote_addr 字符串进行向左补全至32字符。 `* | SELECT lpad(remote_addr, 32, '0')`
ltrim(key)	移除字符串左侧的空白字符。返回结果为 varchar 类型。	移除 http_user_agent 字符串左侧的空白字符。 `* | SELECT ltrim(http_user_agent)`
position(substring IN key)	返回 substring 在字符串中的位置，从1开始。如果没找到，则返回0。该函数有特殊语法 IN 作为参数。另外参考 strpos()。返回结果为 bigint 类型。	返回 'G' 字符在 http_method 中的位置。 `* | select position('G' IN http_method)`
replace(key, substring)	从字符串 key 中移除所有 substring 字符串。返回结果为 varchar 类型。	从 time_local 字符串中移除所有 ‘Oct’ 。 `* | select replace(time_local, 'Oct')`
replace(key, substring, replace)	替换字符串中所有 substring 为 replace 的字符串。返回结果为 varchar 类型。	替换 time_local 字符串中所有 ‘Oct’ 为 ‘10’ 。 `* | select replace(time_local,'Oct','10')`
reverse(key)	翻转字符串 key。返回结果为 varchar 类型。	翻转 host 字符串。 `* | select reverse(host)`
rpad(key, size, padstring)	使用 padstring 对字符串进行向右补全到 size 大小。如果 size 小于 key 的长度，则对字符串进行截断到 size 大小。size 必须为非负数，padstring 必须非空。返回结果为 varchar 类型。	使用 ‘0’ 对 remote_addr 字符串进行向右补全到32位大小字符。 `* | select rpad(remote_addr, 32, '0')`
rtrim(key)	移除字符串右侧所有空白字符。返回结果为 varchar 类型。	移除 http_user_agent 字符串右侧所有空白字符。 `* | select rtrim(http_user_agent)`
split(key, delimiter)	使用指定的分隔符对字符串进行分割，返回字符串数组。	使用指定的 ‘/’ 分隔符对 http_user_agent 字符串进行分割，返回字符串数组。 `* | SELECT split(http_user_agent, '/')`
split(key, delimiter, limit)	使用指定的分隔符对字符串进行分割，返回字符串数组，数组长度最多为 limit。数组中最后一个元素始终为 key 中剩余的所有部分。limit 必须为正数。	使用指定的 ‘/’ 分隔符对 http_user_agent 字符串进行分割，返回数组长度为10的字符串数组。 `* | SELECT split(http_user_agent, '/', 10)`
split_part(key, delimiter, index)	使用指定的分隔符对字符串进行分割，返回数组中 index 位置的字符串。index 从1开始。如果 index 大于数组长度，则返回 null。返回结果为 varchar 类型。	使用指定的 ‘/’ 分隔符对 http_user_agent 字符串进行分割，返回第1位的字符串。 `* | SELECT split_part(http_user_agent, '/', 1)`
strpos(key, substring)	返回 substring 在字符串中的位置，从1开始。如果没找到，则返回0。返回结果为 bigint 类型。	返回 ‘org’ 字符串在 host 字符串中的位置。 `* | SELECT strpos(host, 'org')`
strpos(key, substring, instance)	返回字符串经 substring 分割后的第 instance 个实例在 string 中的位置。如果 instance 是负数，则从尾到头数。位置结果从1开始，如果没找到，则返回0。返回结果为 bigint 类型。	返回 host 字符中经过 ‘g’ 分割后的第1个实例在 host 字符串中的位置。 `* | SELECT strpos(host, 'g', 1)`
substr(key, start)	返回字符串中从 start 位置开始的剩余所有字符，位置从1开始算。如果 start 为负数，则是相对于尾部开始算，例如：[...]返回结果为 varchar 类型。	返回 remote_user 字符串中从第2位开始剩余字符。 `* | SELECT substr(remote_user, 2)`
substr(key, start, length)	返回字符串从 start 开始的最多 length 长度的子串，位置从1开始算。start 为负数，则相对于尾部开始算。返回结果为 varchar 类型。	返回 remote_user 字符串中从第2位开始，总长不超过5位的剩余字符。 `* | SELECT substr(remote_user, 2, 5)`

版权所有：腾讯云计算（北京）有限责任公司

translate(key, from, to)	将 key 中出现在 from 中的字符全部替换为 to 中对应位置的字符。如果 from 包含重复字符，则只算第一个字符。如果 source 中没出现 from 中的字符，则 source 直接复制。如果 from 长度超过 to 的长度，则对应字符会被删除。返回结果为 varchar 类型。	将 remote 字符串中 ‘123’ 的字符替换为 ‘ABC’ 对应的字符。 `* | SELECT translate(remote_user, '123', 'ABC')`
trim(key)	删除字符串中前后的空白字符。返回结果为 varchar 类型。	删除 http_cookies 字符串中前后的空白字符。 `* | SELECT trim(http_cookies)`
upper(key)	将字符串转化为大写字符。 返回结果为 varchar 类型，大写形式。	将 host 字符串中的小写字符转换为大写。 `* | SELECT upper(host)`
word_stem(word)	返回 word 的英语单词。返回结果为 varchar 类型。	返回 Mozila 的单词。 `* | SELECT word_stem('Mozilla')`
word_stem(word, lang)	返回 word 的 lang 语言词根。返回结果为 varchar 类型。	返回 selects 词根的单词。 `* | SELECT word_stem('selects', 'en')`

Unicode 函数

函数名称	说明	示例
normalize (string)	转化 string 为 NFC 标准格式。返回结果为 varchar 类型。	`* | SELECT normalize('ĉ')`
normalize (string, form)	转化 string 为 form 格式。该函数 form 参数需要使用关键字（NFD，NFC，NFKD，NFKC），而非字符串。返回结果为 varchar 类型。	`* | SELECT normalize('ĉ',NFC)`
to_utf8(string)	将 string 转化为 UTF-8 格式的二进制字符串 varbinary。	`* | SELECT to_utf8('test')`
from_utf8 (binary)	将二进制字符串转为 UTF-8 格式的字符串。非法的 UTF-8 字符会被替换为 U+FFFD。返回结果为 varchar 类型。	`* | SELECT from_utf8(from_base64('SGVsbG8='))`
from_utf8 (binary, replace)	将二进制字符串转为 UTF-8 格式的字符串。非法的 UTF-8 字符会被替换为 replace。返回结果为 varchar 类型。	`* | SELECT from_utf8(from_base64('SGVsbG8='),0)`

示例

基于如下日志样例介绍查询和分析语句示例。
原始日志数据样例：

```
10.135.46.111 - - [05/Oct/2015:21:14:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703
"http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum" "Mozilla/5.0
(Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0"  0.354 0.354
```

配置 单行-完全正则采集 模式后，自定义的 key 名称如下所示：

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
```

版权所有：腾讯云计算（北京）有限责任公司

```
status: 200
time_local: [05/Oct/2015:21:14:30 +0800]
upstream_response_time: 0.354
```

分析语句示例：

- 使用问号（？）拆分 **request_url** 字段的值并返回第一个字符串（即文件路径部分），然后统计不同路径对应的请求数量。

```
* | SELECT count(*) AS pv, split_part(request_url, '?', 1) AS Path GROUP BY Path ORDER BY pv DESC LIMIT
3
```

- 提取字段值 **http_protocol** 中的前4个字符（即HTTP部分），然后统计 HTTP 协议对应的请求数量。

```
* | SELECT substr(http_protocol,1,4) AS http_protocol, count(*) AS count group by http_protocol
```

- 将 **remote_user** 字符串中 123 的字符替换为 ABC 对应的字符，返回结果为 Varchar 类型。

```
* | SELECT translate(remote_user, '123', 'ABC')
```

- 从字段值 **remote_user** 中提取第2位至第5位的值。

```
* | SELECT substr(remote_user, 2, 5)
```

- 返回字母 H 在 **http_protocol** 字段值中的位置。

```
* | SELECT strpos(http_protocol, 'H')
```

- 使用正斜线（/）将 **http_protocol** 字段的值拆分成2个子串，并返回子串的集合。

```
* | SELECT split(http_protocol, '/', 2)
```

- 将 **time_local** 字段值中的 **oct** 替换成**10**。

```
* | SELECT replace(time_local, 'Oct', '10')
```

日期和时间函数

最近更新时间：2025-05-28 09:36:52

日志服务（Cloud Log Service，CLS）提供时间分组函数、时间截断函数、时间间隔函数和时序补全函数，支持对日志中的日期和时间进行格式转换，分组聚合等处理。

> **注意：**
> 转换 UNIX 时间戳时，除 histogram 函数和 time_series 函数采用 UTC+8 时区外，其它函数均采用 UTC+0 时区。如需转换时区，可使用具备指定时区功能的函数（例如 `from_unixtime(__TIMESTAMP__/1000, 'Asia/Shanghai')`）或手动添加时区偏移（例如 `cast(__TIMESTAMP__+8*60*60*1000 as timestamp)` 或 `cast(__TIMESTAMP__ as timestamp)+ INTERVAL 8 HOUR`）。

基本函数

函数名称	说明	示例
current_date	返回当前日期。 • 返回值格式：YYYY-MM-DD，例如：2021-05-21。 • 返回值类型：DATE 。	`* | select current_date`
current_time	返回当前时间。 • 返回值格式：HH:MM:SS.Ms Time zone，例如：17:07:52.143+08:00。 • 返回值类型：TIME。	`* | select current_time`
current_timestamp	返回当前时间戳。 • 返回值格式：YYYY-MM-DDTHH:MM:SS.Ms Time zone，例如：2021-07-15T17:10:56.735+08:00[Asia/Shanghai]。 • 返回值类型：TIMESTAMP。	`* | select current_timestamp`
current_timezone()	• 返回 IANA 定义的时区（America/Los_Angeles）或相对于 UTC 的偏移时差（+08:35）。 • 返回值类型：VARCHAR，例如： Asia/Shanghai。	`* | select current_timezone()`
localtime	返回本地时间。 • 返回值格式：HH:MM:SS.Ms，例如：19:56:36。 • 返回值类型：TIME。	`* | select localtime`
localtimestamp	返回本地的日期和时间。 • 返回值格式：YYYY-MM-DD HH:MM:SS.Ms，例如：2021-07-15 19:56:26.908。 • 返回值类型：TIMESTAMP。	`* | select localtimestamp`
now()	返回当前日期和时间，与 current_timestamp 函数同等用法。 • 返回值格式：YYYY-MM-DDTHH:MM:SS.Ms Time zone，例如：2021-07-15T17:10:56.735+08:00[Asia/Shanghai]。 • 返回值类型：TIMESTAMP。	`* | select now()`
last_day_of_month(x)	返回月份最后一天。 • 返回值格式：YYYY-MM-DD，例如：2021-05-31。 • 返回值类型：DATE。	`* | select last_day_of_month(cast(__TIMESTAMP__ as timestamp))`
from_iso8601_date(string)	把 ISO8601 格式的日期表达式转化为 DATE 类型的日期表达式。 • 返回值格式：YYYY-MM-DD，例如：2021-05-31。 • 返回值类型：DATE。	`* | select from_iso8601_date('2021-03-21')`

from_iso8601_timestamp (string)	把 ISO8601 格式的日期时间表达式转化为具有时区的 Timestamp 类型的日期时间表达式。 • 返回值格式：HH:MM:SS.Ms Time zone，例如：2021-07-15T17:10:56.735+08:00[Asia/Shanghai]。 • 返回值类型：TIMESTAMP。	`* | select from_iso8601_timestamp('2020-05-13')`

from_unixtime (unixtime)	把 Unix 时间戳（秒级）转化为 TIMESTAMP 类型的日期时间表达式。
	• 返回值格式：YYYY-MM-DD HH:MM:SS.Ms，例如：2017-05-17 01:41:15.000。
	• 返回值类型：TIMESTAMP 。
	注意：该函数采用UTC+0时区，如需指定时区请使用from_unixtime(unixtime, zone)
		示例1：
		`* | select from_unixtime(1494985275)`
		示例2：
		`* | select from_unixtime(__TIMESTAMP__/1000)`

from_unixtime (unixtime, zone)	把 Unix 时间戳（秒级）转化为具有时区的 TIMESTAMP 类型的日期时间表达式。
	• 返回值格式：YYYY-MM-DD HH:MM:SS.Ms Time zone，例如：2017-05-17T09:41:15+08:00[Asia/Shanghai]。
	• 返回值类型：TIMESTAMP。
		示例1：
		`* | select from_unixtime(1494985275, 'Asia/Shanghai')`
		示例2：
		`* | select from_unixtime(__TIMESTAMP__/1000, 'Asia/Shanghai')`

to_unixtime (timestamp)	把 TIMESTAMP 类型的日期时间表达式转化为 Unixtime 时间戳。 返回值类型：LONG。例如：1626347592。	`* | select to_unixtime(cast(__TIMESTAMP__ as timestamp))`
to_milliseconds (interval)	以毫秒为单位返回间隔的时间值。 返回值类型：BIGINT。例如：300000。	`* | select to_milliseconds(INTERVAL 5 MINUTE)`
to_iso8601(x)	将 DATE 类型或 TIMESTAMP 类型的日期和时间表达式转换为 ISO8601 格式的日期和时间表达式。	`* | select to_iso8601(current_timestamp)`
timezone_hour (timestamp)	返回 TIMESTAMP 所属时区的小时偏移量。	`* | SELECT current_timestamp, timezone_hour(current_timestamp)`
timezone_minute (timestamp)	返回 TIMESTAMP 所属时区的分钟偏移量	`* | SELECT current_timestamp, timezone_minute(current_timestamp)`

时间分组函数

支持按固定时间间隔对日志数据进行分组聚合统计，例如统计每5分钟的访问次数等场景。

函数格式

```
histogram(time_column, interval)
```

参数说明

参数	说明

time_column	时间列（KEY），例如 __TIMESTAMP__。该列的值必须为毫秒的 LONG 类型 UNIX 时间戳或 TIMESTAMP 类型的日期时间表达式。 如果时间列不符合上述要求，可以使用 cast 函数将 ISO8601 格式的时间字符串转换为 TIMESTAMP 类型，例如 `cast('2020-08-19T03:18:29.000Z' as timestamp)`，或使用 date_parse 函数转换其他自定义类型时间字符串。 注意：时间列使用 TIMESTAMP 时，其对应的日期时间表达式需要为 UTC+0 时区。如果日期时间表达式本身为其他时区，需通过计算调整为 UTC+0 时区。例如原始时间为北京时间（UTC+8）时，使用 `cast('2020-08-19T03:18:29.000Z' as timestamp) - interval 8 hour` 进行调整。
interval	固定时间间隔，支持单位为 SECOND（秒）、MINUTE（分）、HOUR（小时）、DAY（天）。例如时间间隔5分钟，即 INTERVAL 5 MINUTE。 动态时间间隔，格式为：${__interval}。${__interval}取值会依据查询的时间范围自动变化。例如，当查询最近1小时的数据，时间间隔是1分钟。查询最近1天的数据，时间间隔是15分钟。（注意：${__interval} 暂不支持在API中使用）

示例

1. 统计每5分钟访问次数 PV 值：

```
* | select histogram(__TIMESTAMP__, INTERVAL 5 MINUTE) AS dt, count(*) as PV group by dt order by dt limit 10000
```

2. 统计动态时间间隔的访问次数 PV 值：

```
* | select histogram(__TIMESTAMP__, INTERVAL ${__interval}) AS dt, count(*) as PV group by dt order by dt limit 10000
```

时间补全函数

time_series() 支持按固定时间间隔对日志数据进行分组聚合统计，与 histogram() 函数的主要差异在于支持补全查询时间窗口内缺失的数据。

> ⚠ **注意：**
> time_series() 函数必须搭配 GROUP BY 语法和 ORDER BY 语法使用，且 ORDER BY 语法不支持 desc 排序方式。

函数格式

```
time_series(time_column, interval, format, padding)
```

参数说明

参数	说明
time_column	时间列（KEY），例如 __TIMESTAMP__。该列的值必须为毫秒的 LONG 类型 UNIX 时间戳或 TIMESTAMP 类型的日期时间表达式。 如果时间列不符合上述要求，可以使用 cast 函数将 ISO8601 格式的时间字符串转换为 TIMESTAMP 类型，例如 `cast('2020-08-19T03:18:29.000Z' as timestamp)`，或使用 date_parse 函数转换其他自定义类型时间字符串。 注意：时间列使用 TIMESTAMP 时，其对应的日期时间表达式需要为 UTC+0 时区。如果日期时间表达式本身为其它时区，需通过计算调整为 UTC+0 时区。例如原始时间为北京时间（UTC+8）时，使用 `cast('2020-08-19T03:18:29.000Z' as timestamp) - interval 8 hour` 进行调整。
interval	固定时间间隔，支持单位为s（秒）、m（分）、h（小时）、d（天）。例如5分钟，即5m。 动态时间间隔，格式为：${__interval}。${__interval}取值会依据查询的时间范围自动变化。例如，当查询最近1小时的数据，时间间隔是1分钟。查询最近1天的数据，时间间隔是15分钟。（注意：${__interval} 暂不支持在API中使用）
format	返回结果的时间格式，使用 format 格式。
padding	补全的内容，包括： • 0：将缺失的值补全为0。 • null：将缺失的值补全为 null。 • ast：将缺失的值补全为上一个时间点的值。

	• next：将缺失的值补全为下一个时间点的值。 • avg：将缺失的值补全为前后两个时间点的平均值。

示例

1. 按照2分钟的时间粒度进行数据补全，查询和分析语句如下：

```
* | select time_series(__TIMESTAMP__, '2m', '%Y-%m-%dT%H:%i:%s+08:00', '0')  as time, count(*) as count
group by time order by time limit 10000
```

2. 按照动态时间粒度进行数据补全，查询和分析语句如下：

```
* | select time_series(__TIMESTAMP__, '${__interval}', '%Y-%m-%dT%H:%i:%s+08:00', '0')  as time, count(*)
as count group by time order by time limit 10000
```

时间截断函数

date_trunc() 函数根据您指定的日期时间部分截断日期时间表达式，支持按照秒、分钟，小时、日、月、年对齐。该函数常用于需要按照时间进行统计分析的场景。

函数名	说明	示例
date_trunc(unit,x)	将 x 截断至 unit 单位。x 为 timestamp 类型。	`* | SELECT date_trunc('second', cast(__TIMESTAMP__ as timestamp))`

截断支持如下粒度：

unit	转化结果	说明
second	2021-05-21 05:20:01.000	-
minute	2021-05-21 05:20:00.000	-
hour	2021-05-21 05:00:00.000	-
day	2021-05-21 00:00:00.000	返回指定日期的零点。
week	2021-05-19 00:00:00.000	返回指定周的周一零点
month	2021-05-01 00:00:00.000	返回指定月份的第一天零点。
quarter	2021-04-01 00:00:00.000	返回指定季度的第一天零点
year	2021-01-01 00:00:00.000	返回本年度第一天零点。

时间提取函数

时间提取函数用于日期和时间表达式中提取指定的时间部分，例如提取年份、月份等。

函数名	说明	示例
extract(field FROM x)	从日期和时间表达式（x）中提取指定的时间部分（field）。	`* |select extract(hour from cast('2021-05-21 05:20:01.100' as timestamp))`

field 支持如下取值：year、quarter、month、week、day、day_of_month、day_of_week、dow、day_of_year、doy、year_of_week、yow、hour、minute、second。

`extract(field FROM x)` 也可以简化为 `field()` 的形式，例如 `extract(hour from cast('2021-05-21 05:20:01.100' as timestamp))` 可简化为 `hour(cast('2021-05-21 05:20:01.100' as timestamp))` 。

版权所有：腾讯云计算（北京）有限责任公司

field	提取结果	说明	简化形式
year	2021	提取目标日期中的年份。	year(x)
quarter	2	提取目标日期所属的季度。	quarter(x)
month	5	提取目标日期中的月份。	month(x)
week	20	计算目标日期是在一年中的第几周。	week(x)
day	21	提取目标日期中的天数，按月分别计算，等同于 day_of_month。	day(x)
day_of_month	21	等同于 day。	day(x)
day_of_week[]	5	计算目标日期是一周中的第几天，等同于 dow。	day_of_week(x)
dow[]	5	等同于 day_of_week。	day_of_week(x)
day_of_year	141	计算目标日期是一年中的第几天，等同于 doy。	day_of_year(x)
doy	141	等同于 day_of_year。	day_of_year(x)
year_of_week	2021	获取目标日期在 ISO week date 中的年份，等同于 yow。	year_of_week(x)
yow	2021	等同于 year_of_week。	year_of_week(x)
hour	5	提取目标日期中的小时。	hour(x)
minute	20	提取目标日期中的分钟。	minute(x)
second	1	提取目标日期中的秒。	second(x)

时间间隔函数

时间间隔函数用来执行时间段相关的运算，例如在日期中添加或减去指定的时间间隔、计算两个日期之间的时间。

函数名	说明	示例
date_add (unit,value,timestamp)	在 timestamp 上加上 N 个时间单位（unit）。value 为负数则为减法。	`* | SELECT date_add('day', -1, TIMESTAMP '2020-03-03 03:01:00')` 返回2020年3月3日1天前的日期和时间，即2020-03-02 03:01:00。
date_diff(unit, timestamp1, timestamp2)	返回两个时间表达式之间的时间差值，例如计算 timestamp1 和 timestamp2 之间相差几个时间单位（unit）。	`* |SELECT date_diff('hour', TIMESTAMP '2020-03-01 00:00:00', TIMESTAMP '2020-03-02 00:00:00')` 返回2020年3月1日和3月2日之间相差的时间单位值，即相差1天。

unit 取值如下：

unit	说明
millisecond	毫秒
second	秒
minute	分钟
hour	小时
day	天

week	周
month	月
quarter	季度
year	年

示例

返回 '2020-03-01 00:00:00' 和 '2020-03-02 00:00:00' 在second单位下的间隔数值，查询分析语句如下：

```
* | SELECT date_diff('second', TIMESTAMP '2020-03-01 00:00:00', TIMESTAMP '2020-03-02 00:00:00')
```

时间持续函数

函数名	说明	示例
parse_duration(string)	将单元值字符串转换为时间段表达式。 返回值类型：INTERVAL。例如：0 00:00:00.043（D HH:MM:SS.Ms）	`* | SELECT parse_duration('3.81 d')`
human_readable_seconds(double)	将单元值字符串转换为时间段表达式。 返回值类型：VARCHAR。例如：1 minutes, 36 seconds。	`* | SELECT human_readable_seconds(96)`

支持粒度如下：

Unit	说明
ns	纳秒
us	微秒
ms	毫秒
s	秒
m	分钟
h	小时
d	天

示例

将单元值'3.81 d' 单元字符串的值转换为间隔字符串的值，查询分析语句如下：

```
* | SELECT parse_duration('3.81 d')
```

时间格式化函数

函数名	说明	示例
date_format(timestamp, format)	把 TIMESTAMP 类型的日期时间转化为 Format 格式的字符串。	`* | select date_format(cast(__TIMESTAMP__ as timestamp), '%Y-%m-%d')`
date_parse(string, format)	把 Format 格式的日期时间字符串转化为 TIMESTAMP 类型。	`* | select date_parse('2017-05-17 09:45:00','%Y-%m-%d %H:%i:%s')`

Format 说明：

Format	说明
%a	星期的缩写。例如 Sun、Sat。
%b	月份的缩写。例如 Jan、Dec。
%c	月份。数值类型，取值范围为1 – 12。
%d	每月的第几天。十进制格式，取值范围为01 – 31。
%e	每月的第几天。十进制格式，取值范围为1 – 31。
%f	秒的小数部分，date_format展示数据时采用6位数字(000000~999000)，date_parse转换数据时支持1～9位数字(0~999999999)。
%H	小时，24小时制。
%h	小时，12小时制。
%I	小时，12小时制。
%i	分钟。数值类型，取值范围为00 – 59。
%j	每年的第几天。取值范围为001 – 366。
%k	小时。取值范围为0 – 23。
%l	小时。取值范围为1 – 12。
%M	月份的英文表达，例如 January、December。
%m	月份的数字表达，例如 01、02。
%p	AM、PM。
%r	时间。12小时制，格式为 `hh:mm:ss AM/PM` 。
%S	秒。取值范围为00 – 59。 注意：此处不支持毫秒，如需使用毫秒可使用%f并结合其它函数进行替代，例如 `substr(date_format(current_timestamp,'%Y-%m-%d %T.%f'),1,23)` 。
%s	秒。取值范围为00 – 59。
%T	时间。24小时制，格式为 `hh:mm:ss` 。
%v	每年的第几周，星期一是一周的第一天。取值范围为01 – 53。
%W	星期几的名称。例如 Sunday、Saturday。
%Y	4位数的年份。例如2020。
%y	2位数的年份。例如20。
%%	%的转义字符。

示例

将 format 格式的时间字符串 '2017-05-17 09:45:00' 转换为 TIMESTAMP 类型的日期时间表达式，即 '2017-05-17 09:45:00.0' 。查询分析语句如下：

```
* | SELECT date_parse('2017-05-17 09:45:00','%Y-%m-%d %H:%i:%s')
```

IP 函数

最近更新时间：2025-05-16 10:47:52

IP 函数可根据 IP 地址获取 IP 地理信息及安全威胁信息，本文介绍 IP 函数的基本语法及示例。

IP 地理函数

仅支持 IPv4

说明：

- 如下函数中的 KEY 参数表示日志字段（例如 ip），其值为 IP 地址；若其值为内部 IP 地址或非法字段将无法被解析，以 “NULL” 或 “未知” 展示。
- 基于 IP 地址分配机制的限制，IP 地址库无法100%准确涵盖所有 IP 地址的地理信息，极少部分 IP 地址可能会查询不到详细地理信息或地理信息存在错误。

函数名称	说明	示例
ip_to_country(KEY)	分析目标 IP 地址所属国家或地区。返回结果为国家或地区的中文名称。	`* | SELECT ip_to_country(ip)`
ip_to_country_code(KEY)	分析目标 IP 地址所属国家或地区的代码。返回结果为国家或地区的代码。	`* | SELECT ip_to_country_code(ip)`
ip_to_country_geo(KEY)	分析目标 IP 地址所属国家或地区的经纬度。返回结果为国家或地区的经纬度。	`* | SELECT ip_to_country_geo(ip)`
ip_to_province(KEY)	分析目标 IP 地址所属省份。返回结果为省份的中文名称。	`* | SELECT ip_to_province(ip)`
ip_to_province_code(KEY)	分析目标 IP 地址所属省份的代码。返回结果为省份的行政区划代码。	`* | SELECT ip_to_province_code(ip)`
ip_to_province_geo(KEY)	分析目标 IP 地址所属省份的经纬度。返回结果为省份的经纬度。	`* | SELECT ip_to_province_geo(ip)`
ip_to_city(KEY)	分析目标 IP 地址所属城市。返回结果为城市的中文名称，国外城市为英文名。	`* | SELECT ip_to_city(ip)`
ip_to_city_code(KEY)	分析目标 IP 地址所属城市的代码。返回结果为城市的行政区规划代码，仅支持中国城市。	`* | SELECT ip_to_city_code(ip)`
ip_to_city_geo(KEY)	分析目标 IP 地址所属城市的经纬度。返回结果为城市的经纬度，仅支持中国城市。	`* | SELECT ip_to_city_geo(ip)`
ip_to_provider(KEY)	分析目标 IP 地址对应的网络运营商，返回结果为网络运营商名称。	`* | SELECT ip_to_provider(ip)`

支持 IPv4 及 IPv6

注意：

- **受版权限制，该函数仅面向少部分大客户开放，如需开通请联系您的腾讯云技术服务团队进行评估。**
- 使用该函数时，SQL 返回结果超过1万行后，超出部分的 IP 解析结果将自动隐藏，显示为 `Hiding Detailed Geolocation Data in IPv6` 或 `NULL`，该限制仅针对 SQL 最终计算结果，不影响 SQL 中间环节的计算准确性。
- 不支持获取 IP 地址对应的网络运营商，即 ip_to_provider 函数。
- 基于 IP 地址分配机制的限制，IP 地址库无法100%准确涵盖所有 IP 地址的地理信息，极少部分 IP 地址可能会查询不到详细地理信息或地理信息存在错误。

函数名称	说明	示例

ip_to_country(KEY,'enable_ipv6')	分析目标 IP 地址所属国家或地区。返回结果为国家或地区的中文名称。	`* | SELECT ip_to_country(ip,'enable_ipv6')`
ip_to_country_code(KEY,'enable_ipv6')	分析目标 IP 地址所属国家或地区的代码。返回结果为国家或地区的代码。	`* | SELECT ip_to_country_code(ip,'enable_ipv6')`
ip_to_country_geo(KEY,'enable_ipv6')	分析目标 IP 地址所属国家或地区的经纬度。返回结果为国家或地区的经纬度。	`* | SELECT ip_to_country_geo(ip,'enable_ipv6')`
ip_to_province(KEY,'enable_ipv6')	分析目标 IP 地址所属省份。返回结果为省份的中文名称。	`* | SELECT ip_to_province(ip,'enable_ipv6')`
ip_to_province_code(KEY,'enable_ipv6')	分析目标 IP 地址所属省份的代码。返回结果为省份的行政区划代码。	`* | SELECT ip_to_province_code(ip,'enable_ipv6')`
ip_to_province_geo(KEY,'enable_ipv6')	分析目标 IP 地址所属省份的经纬度。返回结果为省份的经纬度。	`* | SELECT ip_to_province_geo(ip,'enable_ipv6')`
ip_to_city(KEY,'enable_ipv6')	分析目标 IP 地址所属城市。返回结果为城市的中文名称，国外城市为英文名。	`* | SELECT ip_to_city(ip,'enable_ipv6')`
ip_to_city_code(KEY,'enable_ipv6')	分析目标 IP 地址所属城市的代码。返回结果为城市的行政区规划代码，仅支持中国城市。	`* | SELECT ip_to_city_code(ip,'enable_ipv6')`
ip_to_city_geo(KEY,'enable_ipv6')	分析目标 IP 地址所属城市的经纬度。返回结果为城市的经纬度，仅支持中国城市。	`* | SELECT ip_to_city_geo(ip,'enable_ipv6')`

IP 网段函数

说明：

- 如下函数中的 KEY 参数表示日志字段（例如 ip），其值为 IP 地址。
- 其中 ip_subnet_min、ip_subnet_max、ip_subnet_range 函数中的字段值为子网掩码格式的 IP 地址（例如：192.168.1.0/24），如果字段值为通用的 IP 地址，则需要使用 concat 函数将其转换为子网掩码格式。

仅支持 IPv4

函数名	说明	示例
ip_prefix(KEY,prefix_bits)	获取目标 IP 地址的前缀。返回结果为子网掩码格式 IP 地址。例如192.168.1.0/24。	`* | SELECT ip_prefix(ip,24)`
ip_subnet_min(KEY)	获取 IP 网段中的最小 IP 地址。返回结果为 IP 地址，例如192.168.1.0。	`* | SELECT ip_subnet_min(concat(ip,'/24'))`
ip_subnet_max(KEY)	获取 IP 网段中的最大 IP 地址。返回结果为 IP 地址，例如192.168.1.255。	`* | SELECT ip_subnet_max(concat(ip,'/24'))`
ip_subnet_range(KEY)	获取 IP 网段范围。返回结果为 Array 类型的 IP 地址，例如["172.16.0.0","172.16.0.255"]。	`* | SELECT ip_subnet_range(concat(ip,'/24'))`
is_subnet_of(KEY)	判断目标 IP 地址是否在某网段内。返回结果为布尔值。	`* | SELECT is_subnet_of('192.168.0.1/24', ip)`
is_prefix_subnet_of(KEY)	判断目标网段是否为某网段的子网。返回结果为布尔值。	`* | SELECT is_prefix_subnet_of('192.168.0.1/24',concat(ip, '/24'))`

支持 IPv4 及 IPv6

函数名	说明	示例
ip_to_domain(KEY)	判断目标 IP 地址是内网地址还是外网地址。内网返回 intranet，外网返回 internet，非法地址返回 invalid。	`* | SELECT ip_to_domain(ip)`
ip_version(KEY)	获取 IP 地址版本，返回结果为 ipv4或 ipv6。	`* | SELECT ip_version(ip,24)`

IP 威胁检测函数

根据 腾讯安全威胁情报中心 中的安全情报，判断指定 IP 是否为恶意 IP。

说明：
- 如下函数中的 KEY 参数表示日志字段（例如 ip），其值为 IP 地址。
- 检测基于安全情报大数据，其结果表示该 IP 是否曾存在恶意行为（例如曾注入攻击某第三方系统），并不代表其在当前日志中或当前系统中一定存在恶意行为。

函数名	说明	示例
ip_threat(KEY)	IP 是否为恶意 IP，返回结果为： • UNKNOWN：未知（未检测到威胁信息）。 • SECURE：安全。 • SUSPICIOUS：可疑。 • MALICIOUS：恶意。	`* | SELECT ip_threat(ip)`
ip_threat_tags(KEY)	可疑及恶意 IP 的 威胁标签，代表该 IP 存在的具体恶意行为，返回结果为数组。	`* | SELECT ip_threat_tags(ip)`

ip_threat_detail(KEY)	IP 威胁检测详情，返回结果为 JSON，例如：
	{
	"reputation": "SECURE",
	"tags": ["Tencent Business Scan"],
	"extraInfo": {
	"announcement": "This IP is the scanning IP for WeChat Official Accounts and Mini-Programs, which will regularly scan your data transmission to ensure smooth callbacks and stable services. If you detect scanning activities from this IP, please allow or whitelist it.",
	"demo": "-",
	"descibe": null,
	"referer": "https://developers.weixin.qq.com/community/develop/doc/0000682a9aca4090cdfb008475b401"
	},
	"extraInfo_zh": {
	"announcement": "此 IP为 微信公众号和小程序 的扫描 IP，将定期对您的 数据传输 进行检测，以 保障回调顺畅、服务稳定，若您发现来自此 IP 的扫描行为，请进行放行或加白...",
	"demo": "-",
	"descibe": null,
	"referer": "https://developers.weixin.qq.com/community/develop/doc/0000682a9aca4090cdfb008475b401"
	}
	}
	其中：
	• reputation：IP 是否为恶意 IP，内容同 `ip_threat(KEY)` 。
		`* | SELECT ip_threat_detail(ip)`

	• tags：可疑及恶意 IP 的威胁标签，内容同 `ip_threat_tags(KEY)`。 • extraInfo：IP 详细说明，仅部分安全（SECURE）IP 具备，其他 IP 请前往 腾讯安全威胁情报中心 查看。 • extraInfo_zh：IP 详细说明中文版本。	

威胁标签说明

部分常见威胁标签及含义如下：

威胁标签	含义	详细说明
xx Exploit vulnerability	xx 漏洞利用	该 IP 使用特定漏洞对用户发起攻击，例如 Weblogic 漏洞利用。
xx attack	xx 攻击	该 IP 对用户发起过特定类型的攻击，例如 CC 攻击、撞库攻击等。
xx Scanning	xx 扫描	该 IP 使用特定的漏洞对用户资产进行扫描，例如致远 OA 扫描（Zhiyuan OA Scanning）。
Brute force	暴力破解	该 IP 对用户系统存在密码尝试行为。
Proxy seconds dial	代理秒拨	• 代理：该 IP 在某时间点被网络黑产作为代理 IP 使用。 • 秒拨：该 IP 在某时间点是网络黑产提供的秒拨 IP 资源（黑产利用家庭宽带拨号在一定时间内持有作恶的 IP）。
Environmental forgery	环境伪造	该 IP 通过伪造环境以绕过或欺骗服务的安全检测。
Network traffic risk control	流量风控	该 IP 的流量触发了流量风控规则。
group control	群控	该 IP 的流量来源可能来自于群控平台或系统。

示例

此处列举了 IP 函数在不同场景下的查询和分析示例。您在执行查询和分析操作后，还可以选择合适的统计图表展示查询和分析结果。

> **说明：**
> 如下示例中的 ip 为日志字段。

- 统计不是来自内网的请求总数。

```
* | SELECT count(*) AS PV where ip_to_domain(ip)!='intranet'
```

- 统计请求总数 Top10的省份。

```
* | SELECT ip_to_province(ip) AS province, count(*) as PV GROUP BY province ORDER BY PV desc LIMIT 10
```

- 如果上述结果中包含了内网请求，且您希望过滤这部分请求，可参考如下语句。

```
* | SELECT ip_to_province(ip) AS province, count(*) as PV where ip_to_domain(ip)!='intranet' GROUP BY
province ORDER BY PV desc LIMIT 10
```

- 统计 IP 地址的经纬度，确认客户端分布情况。

```
* | SELECT ip_to_geo(ip) AS geo, count(*) AS pv GROUP BY geo ORDER BY pv DESC
```

- 统计恶意 IP，并获取每个 IP 的威胁等级、威胁标签、访问次数。

```
  * |
select
  ip,
```

```
  CASE WHEN ip_threat(ip)= 'SUSPICIOUS' THEN '可疑' WHEN ip_threat(ip)= 'MALICIOUS' THEN '恶意' END as
"威胁等级",
  ip_threat_tags(ip) as "威胁标签",
  count(*) as "访问次数"
where
  ip_threat(ip) = 'SUSPICIOUS'
  or ip_threat(ip) = 'MALICIOUS'
group by
  ip
order by
  count(*) desc
limit
  10000
```

URL 函数

最近更新时间：2024-07-11 16:43:41

本文介绍 URL 函数的语法及示例。

语法格式

URL 函数支持从标准 HTTP URL 路径中提取字段，一个标准的 URL 如下：

```
[protocol:][//host[:port]][path][?query][#fragment]
```

> **注意：**
> 提取的字段中不包含 URL 分割符 `:` 或 `?` 。

常见 URL 函数

函数名	说明	示例	输出结果
url_extract_fragment(url)	提取出 URL 中的 fragment，结果为 varchar 类型。	`* | select url_extract_fragment('https://console.cloud.tencent.com/#/project/dashboard-demo/categoryList')`	`/project/dashboard-demo/categoryList`
url_extract_host(url)	提取出 URL 中的 host，结果为 varchar 类型。	`* | select url_extract_host('https://console.cloud.tencent.com/cls')`	`console.cloud.tencent.com`
url_extract_parameter(url, name)	提取出 URL 中的 query 对应的参数值，结果为 varchar 类型。	`* | select url_extract_parameter('https://console.cloud.tencent.com/cls?region=ap-chongqing','region')`	`ap-chongqing`
url_extract_path(url)	提取出 URL 中的 path，结果为 varchar 类型。	`* | select url_extract_path('https://console.cloud.tencent.com/cls?region=ap-chongqing')`	`/cls`
url_extract_port(url)	提取出 URL 中的端口，结果为 bigint 类型。	`* | select url_extract_port('https://console.cloud.tencent.com:80/cls?region=ap-chongqing')`	`80`
url_extract_protocol(url)	提取出 URL 中的协议，结果为 varchar 类型。	`* | select url_extract_protocol('https://console.cloud.tencent.com:80/cls?region=ap-chongqing')`	`https`

url_extract_query(url)	提取出 URL 中的 query，结果为 varchar 类型。	`* | select url_extract_query('https://console.cloud.tencent.com:80/cls?region=ap-chongqing')`	`region=ap-chongqing`
url_encode(value)	对 value 进行转义编码，使之能应用在 URL_query 中。 • 字母不会被解码。 • .-*_不会被编码。 • 空格被解码为+。 • 其他字符被解码为 UTF8 格式。	`* | select url_encode('https://console.cloud.tencent.com:80/cls?region=ap-chongqing')`	`https%3A%2F%2Fconsole.cloud.tencent.com%3A80%2Fcls%3Fregion%3Dap-chongqing`
url_decode(value)	对 URL 进行解码。	`* | select url_decode('https%3A%2F%2Fconsole.cloud.tencent.com%3A80%2Fcls%3Fregion%3Dap-chongqing')`	`https://console.cloud.tencent.com:80/cls?region=ap-chongqing`

JSON 函数

最近更新时间：2025-06-25 19:05:52

本文介绍 JSON 函数的基本语法及示例。

函数名称	语句	含义
json_array_contains 函数	json_array_contains(x,value)	判断 JSON 数组中是否包含某个值。
json_array_get 函数	json_array_get(x,index)	获取 JSON 数组中某个下标对应的元素。
json_array_length 函数	json_array_length(x)	计算 JSON 数组中元素的数量。x 不是 JSON 数组时，返回 null。
json_extract 函数	json_extract(x,json_path)	从 JSON 对象或 JSON 数组中提取一组 JSON 值（数组或对象）。
json_extract_scalar 函数	json_extract_scalar(x,json_path)	从 JSON 对象或 JSON 数组中提取一组标量值（字符串、整数或布尔值）。类似于 json_extract 函数。
json_format 函数	json_format(x)	把 JSON 类型转化成字符串类型。
json_parse 函数	json_parse(x)	把字符串类型转化成 JSON 类型。
json_size 函数	json_size(x,json_path)	计算 JSON 对象或数组中元素的数量。

json_array_contains 函数

json_array_contains 函数用于判断 JSON 数组中是否包含某个值。

语法

```
json_array_contains(x, value)
```

参数说明

参数	说明
x	参数值为 JSON 数组。
value	数值。

返回值类型

boolean 类型。

示例

判断 JSON 字符[1，2，3]中是否包含2。

- 查询和分析语句。

```
* | SELECT json_array_contains('[1, 2, 3]', 2)
```

- 查询和分析结果。

```
true
```

json_array_get 函数

json_array_get 函数用于获取 JSON 数组中某个下标对应的元素。

语法

```
json_array_get(x, index)
```

参数说明

参数	说明
x	参数值为 JSON 数组。
index	JSON 下标，从0开始。

返回值类型

varchar 类型。

示例

返回 JSON 数组["a", [3, 9], "c"]下标为1的元素。

- 查询和分析语句。

```
* | SELECT json_array_get('["a", [3, 9], "c"]', 1)
```

- 查询和分析结果。

```
[3,9]
```

json_array_length 函数

json_array_length 函数用于计算 JSON 数组中元素的数量。x 不是 JSON 数组时，返回 null。

语法

```
json_array_length(x)
```

参数说明

参数	说明
x	参数值为 JSON 数组。

返回值类型

bigint 类型。

示例

- 示例1：计算 apple.message 字段值中 JSON 元素的数量。

```
apple.message:[{"traceName":"StoreMonitor"},{"topicName":"persistent://apache/pulsar/test-partition-
17"},{"producerName":"pulsar-mini-338-36"},{"localAddr":"pulsar://pulsar-mini-broker-5.pulsar-mini-
broker.pulsar.svc.cluster.local:6650"},{"sequenceId":826},{"storeTime":1635905306062},
{"messageId":"19422-24519"},{"status":"SUCCESS"}]
```

- 查询和分析语句。

```
* | SELECT json_array_length(apple.message)
```

- 查询和分析结果。

```
8
```

json_extract 函数

json_extract 函数用于从 JSON 对象或 JSON 数组中提取一组 JSON 值（数组或对象）。

语法

```
json_extract(x, json_path)
```

参数说明

参数	说明
x	参数值为 JSON 对象或 JSON 数组。
json_path	JSONPath 形式的 JSON 路径，例如 `$.store.book[0].title`。 ⚠ 注意： • 不支持需要遍历数组元素的 JSON 语法，例如 `$.store.book[*].author`、`$..book[(@.length-1)]`、`$..book[?(@.price<10)]` 等。 • 元素名称包含特殊字符时，使用 `[]` 语法代替 `.` 语法并用双引号包裹元素名称，例如 `$.store["book name"]`。

返回值类型

JSON 格式的 string 类型。

示例

获取 apple.instant 字段中 epochSecond 的值。

- 字段样例。

```
apple.instant:{"epochSecond":1635905306,"nanoOfSecond":63001000}
```

- 查询和分析语句。

```
* | SELECT json_extract(apple.instant, '$.epochSecond')
```

- 查询和分析结果。

```
1635905306
```

json_extract_scalar 函数

json_extract_scalar 函数用于从 JSON 对象或 JSON 数组中提取一组标量值（字符串、整数或布尔值）。

语法

```
json_extract_scalar(x, json_path)
```

参数说明

参数	说明
x	参数值为 JSON 数组。

json_path	JSONPath 形式的 JSON 路径，例如 `$.store.book[0].title` 。 **注意：** • 不支持需要遍历数组元素的 JSON 语法，例如 `$.store.book[*].author` 、 `$..book[(@.length-1)]` 、 `$..book[?(@.price<10)]` 等。 • 元素名称包含特殊字符时，使用 `[]` 语法代替 `.` 语法并用双引号包裹元素名称，例如 `$.store["book name"]` 。

返回值类型

varchar 类型。

示例

从 apple.instant 字段中获取 epochSecond 字段的值，并将该值转换为 bigint 类型进行求和。

- 字段样例。

```
apple.instant:{"epochSecond":1635905306,"nanoOfSecond":63001000}
```

- 查询和分析语句。

```
* | SELECT sum(cast(json_extract_scalar(apple.instant,'$.epochSecond') AS bigint) )
```

- 查询和分析结果。

```
1635905306
```

json_format 函数

json_format 函数用于将 JSON 类型转化成字符串类型。

语法

```
json_format(x)
```

参数说明

参数	说明
x	参数值为 JSON 类型。

返回值类型

varchar 类型。

示例

将 JSON 数组[1,2,3]转换为字符串。

- 查询和分析语句。

```
* | SELECT json_format(json_parse('[1, 2, 3]'))
```

- 查询和分析结果。

```
[1, 2, 3]
```

json_parse 函数

json_parse 函数只用于将字符串类型转化成 JSON 类型，判断是否符合 JSON 格式。

语法

```
json_parse(x)
```

参数说明

参数	说明
x	参数值为字符串。

返回值类型

JSON 类型。

示例

将字符串[1, 2, 3]转换为 JSON 数组。

- 查询和分析语句。

```
* | SELECT json_parse('[1, 2, 3]')
```

- 查询和分析结果。

```
[1, 2, 3]
```

json_size 函数

json_size 函数用于计算 JSON 对象或 JSON 数组中元素的数量。

语法

```
json_size(x, json_path)
```

参数说明

参数	说明
x	参数值为 JSON 对象或 JSON 数组。
json_path	JSONPath 形式的 JSON 路径，例如 `$.store.book[0].title` 。 **注意：** • 不支持需要遍历数组元素的 JSON 语法，例如 `$.store.book[*].author` 、 `$..book[(@.length-1)]` 、 `$..book[?(@.price<10)]` 等。 • 元素名称包含特殊字符时，使用 `[]` 语法代替 `.` 语法并用双引号包裹元素名称，例如 `$.store["book name"]` 。

返回值类型

bigint 类型。

示例

获取 JSON 数组[1,2,3]的长度。

- 查询和分析语句。

```
* | SELECT json_size(json_parse('[1, 2, 3]'),'$')
```

- 查询和分析结果。

```
3
```

正则式函数

最近更新时间：2024-05-29 14:28:51

本文介绍正则式函数基本语法及示例。
日志服务支持如下正则式函数：

函数名称	语法	说明
regexp_extract_all 函数	regexp_extract_all(x, regular expression)	提取目标字符串中符合正则表达式的子串，并返回所有子串的合集。
	regexp_extract_all(x, regular expression, n)	提取目标字符串中符合正则表达式的子串，并返回与目标捕获组匹配的子串合集。
regexp_extract 函数	regexp_extract(x, regular expression)	提取并返回目标字符串中符合正则表达式的第一个子串。
	regexp_extract(x, regular expression, n)	提取目标字符串中符合正则表达式的子串，然后返回与目标捕获组匹配的第一个子串。
regexp_like 函数	regexp_like(x, regular expression)	判断目标字符串是否符合正则表达式。
regexp_replace 函数	regexp_replace(x, regular expression)	删除目标字符串中符合正则表达式的子串，返回未被删除的子串。
	regexp_replace(x, regular expression, replace string)	替换目标字符串中符合正则表达式的子串，返回被替换后的字符串。
regexp_split 函数	regexp_split(x, regular expression)	使用正则表达式分割目标字符串，返回被分割后的子串合集。

regexp_extract_all 函数

regexp_extract_all 函数用于提取目标字符串中符合正则表达式的子串合集。

语法

- 提取目标字符串中符合正则表达式的子串，并返回所有子串的合集。

```
regexp_extract_all(x, regular expression)
```

- 提取目标字符串中符合正则表达式的子串，然后返回与目标捕获组匹配的子串合集。

```
regexp_extract_all(x, regular expression, n)
```

参数说明

参数	说明
x	参数值为 varchar 类型。
regular expression	正则表达式，例如 `\d+` 。同时支持捕获组，例如 `(\d)(\d)(\d)` 。
n	第 n 个捕获组。n 为从1开始的整数。

返回值类型

Array 类型。

示例

提取 http_protocol 字段中所有的数字，其中 http_protocol 的值为 HTTP/1.1。

- 查询和分析语句

```
* | SELECT regexp_extract_all(http_protocol, '\d+')
```

- 查询和分析结果

```
[1,1]
```

regexp_extract 函数

regexp_extract 函数用于提取目标字符串中符合正则表达式的第一个子串。

语法

- 提取并返回目标字符串中符合正则表达式的第一个子串。

```
regexp_extract(x, regular expression)
```

- 提取目标字符串中符合正则表达式的子串，然后返回与目标捕获组匹配的第一个子串。

```
regexp_extract(x, regular expression, n)
```

参数说明

参数	说明
x	参数值为 varchar 类型。
regular expression	正则表达式，例如 `\d+` 。同时支持捕获组，例如 `(\d)(\d)(\d)` 。
n	第 n 个捕获组。n 为从1开始的整数。

返回值类型

Varchar 类型。

示例

提取 **http_protocol** 字段值中的第一个数字，其中 **http_protocol** 的值为 HTTP/1.1。

- 查询和分析语句

```
* | SELECT regexp_extract(http_protocol, '\d+')
```

- 查询和分析结果

```
1
```

regexp_like 函数

regexp_like 函数用于判断目标字符串是否符合正则表达式。

语法

```
regexp_like(x, regular expression)
```

参数说明

参数	说明
x	参数值为 varchar 类型。

regular expression	正则表达式。

返回值类型

Boolean 类型。

示例

判断 server_protocol 字段值中是否包含数字，其中 server_protocol 的值为 HTTP/1.1。

- 查询和分析语句

```
* | select regexp_like(server_protocol, '\d+')
```

- 查询和分析结果

```
TRUE
```

regexp_replace 函数

regexp_replace 函数用于删除或替换目标字符串中符合正则表达式的子串。

语法

- 删除目标字符串中符合正则表达式的子串，返回未被删除的子串。

```
regexp_replace(x, regular expression)
```

- 替换目标字符串中符合正则表达式的子串，返回被替换后的字符串。

```
regexp_replace(x, regular expression, replace string)
```

参数说明

参数	说明
x	参数值为 varchar 类型。
regular expression	正则表达式。
replace string	用于替换的子串.

返回值类型

String 类型。

示例

删除 server_protocol 字段值中的版本号部分，并统计不同通信协议对应的请求数量，其中 server_protocol 的值为 HTTP/1.1。

- 查询和分析语句

```
* | select regexp_replace(server_protocol, '.\d+') AS server_protocol, count(*) AS count GROUP BY
server_protocol
```

- 查询和分析结果

server_protocol	count
HTTP	357

regexp_split 函数

regexp_split 函数用于分割目标字符串，返回被分割后的子串合集。

语法

```
regexp_split(x, regular expression)
```

参数说明

参数	说明
x	参数值为 varchar 类型。
regular expression	正则表达式。

返回值类型

Array 类型。

示例

使用正斜线（/）分割 server_protocol 字段的值，其中 server_protocol 的值为 HTTP/1.1。

- 查询和分析语句

```
* | select regexp_split(server_protocol, '/')
```

- 查询和分析结果

```
["HTTP","1.1"]
```

数组函数

最近更新时间：2025-06-26 18:56:52

本文档为您介绍数组相关函数的语法及示例。

函数名称	语句	含义
下标运算符 []	[x]	[]用于获取数组中的某个元素，等同于 element_at 函数。
array_agg 函数	array_agg(x)	以数组形式返回 x 中的所有值。
array_distinct 函数	array_distinct(x)	数组去重，获取数组中的唯一元素。
array_except 函数	array_except(x,y)	计算 x，y 两个数组的差集。
array_intersect 函数	array_intersect(x, y)	计算 x，y 两个数组的交集。
array_join 函数	array_join(x, delimiter)	使用指定的连接符将数组中的元素拼接为一个字符串。如果数组中包含 null 元素，则 null 元素将被忽略。说明： 使用 array_join 函数时，返回结果最大为1KB，超出1KB的数据会被截断。
	array_join(x, delimiter, null_replacement)	把字符串数组用 delimiter 连接，拼接成字符串，null 值用 null_replacement 替代。说明：使用 array_join 函数时，返回结果最大为1KB，超出1KB的数据会被截断。
array_max 函数	array_max(x)	获取数组中的最大值。
array_min 函数	array_min(x)	获取数组中的最小值。
array_position 函数	array_position(x, element)	获取指定元素的下标，下标从1开始。如果指定元素不存在，则返回0。
array_remove 函数	array_remove(x, element)	删除数组中指定的元素。
array_sort 函数	array_sort(x)	对数组元素进行升序排序。如果有 null 元素，则 null 元素排在最后。
array_union 函数	array_union(x, y)	计算两个数组的并集。
cardinality 函数	cardinality(x)	计算数组中元素的个数。
concat 函数	concat(x, y...)	将多个数组拼接为一个数组。
contains 函数	contains(x, element)	判断数组中是否包含指定元素。如果包含，则返回 true。
element_at 函数	element_at(x, y)	返回数组中的第 y 个元素。
filter 函数	filter(x, lambda_expression)	结合 Lambda 表达式，用于过滤数组中的元素，只返回满足 Lambda 表达式的元素。
flatten 函数	flatten(x)	将二维数组转换为一维数组。
reduce 函数	reduce(x, lambda_expression)	根据 Lambda 表达式的定义，对数组中的各个元素进行相加计算，然后返回计算结果。
reverse 函数	reverse(x)	对数组中的元素进行反向排列。
sequence 函数	sequence(x, y)	通过指定的起始值返回一个数组，其元素为起始值范围内一组连续且递增的值。递增间隔为默认值1。
	sequence(x, y, step)	通过指定的起始值返回一个数组，其元素为起始值范围内一组连续且递增的值。自定义递增间隔。

版权所有：腾讯云计算（北京）有限责任公司

shuffle 函数	shuffle(x)	对数组元素进行随机排列。
slice 函数	slice(x, start, length)	获取数组的子集。
transform 函数	transform(x, lambda_expression)	将 Lambda 表达式应用到数组的每个元素中。
zip 函数	zip(x, y)	将多个数组合并为一个二维数组，且各个数组中下标相同的元素组成一个新的数组。
zip_with 函数	zip_with(x, y, lambda_expression)	根据 Lambda 表达式中的定义将两个数组合并为一个数组。

下标运算符 []

下标运算符用于返回数组中的第 x 个元素。等同于 element_at 函数。

语法

```
[x]
```

参数说明

参数	说明
x	数组下标，从1开始。参数值为 bigint 类型。

返回值类型

返回指定元素的数据类型。

示例

返回 number 字段值中的第2个元素。

- 字段样例

```
array:[12,23,26,48,26]
```

- 查询和分析语句

```
* | SELECT cast(json_parse(array) as array(bigint))[2]
```

- 查询和分析结果

```
23
```

array_agg 函数

array_agg 函数会以数组形式返回 x 中所有的值。

语法

```
array_agg(x)
```

参数说明

参数	说明

x	参数值为任意数据类型。

返回值类型

array 类型。

示例

以数组形式返回 status 字段的值。

- 查询和分析语句

```
* | SELECT array_agg(status) AS array
```

- 查询和分析结果

```
[200,200,200,404,200,200]
```

array_distinct 函数

array_distinct 函数用于删除数组中重复的元素。

语法

```
array_distinct(x)
```

参数说明

参数	说明
x	参数值为 array 类型。

返回值类型

array 类型。

示例

删除 array 字段值中重复的元素。

- 字段样例

```
array:[12,23,26,48,26]
```

- 查询和分析语句

```
* | SELECT array_distinct(cast(json_parse(array) as array(bigint)))
```

- 查询和分析结果

```
[12,23,26,48]
```

array_except 函数

array_except 函数用于计算两个数组的差集。

语法

```
array_except(x, y)
```

参数说明

参数	说明
x	参数值为 array 类型。
y	参数值为 array 类型。

返回值类型

array 类型。

示例

计算数组[1，2，3，4，5]和[1，3，5，7]的差集。

- 查询和分析语句

```
* | SELECT array_except(array[1,2,3,4,5],array[1,3,5,7])
```

- 查询和分析结果

```
[2,4]
```

array_intersect 函数

array_intersect 函数用于计算两个数组的交集。

语法

```
array_intersect(x, y)
```

参数说明

参数	说明
x	参数值为 array 类型。
y	参数值为 array 类型。

返回值类型

array 类型。

示例

计算数组[1，2，3，4，5]和[1，3，5，7]的交集。

- 查询和分析语句

```
* | SELECT array_intersect(array[1,2,3,4,5],array[1,3,5,7])
```

- 查询和分析结果

```
[1,3,5]
```

array_join 函数

array_join 函数使用指定的连接符将数组中的元素拼接为一个字符串。

语法

- 使用指定的连接符将数组中的元素拼接为一个字符串。如果数组中包含 null 元素，则 null 元素将被忽略。

```
array_join(x, delimiter)
```

- 使用指定的连接符将数组中的元素拼接为一个字符串。如果数组中包含 null 元素，则 null 元素将被替换为 null_replacement。

```
array_join(x, delimiter, null_replacement)
```

参数说明

参数	说明
x	参数值为 array 类型。
delimiter	连接符，可以为字符串。
null_replacement	用于替换 null 元素的字符串。

返回值类型

varchar 类型。

示例

使用空格将数组[null, '中国','sh']中的元素拼接为一个字符串，其中 null 元素替换为 ‘region’ 。

- 查询和分析语句

```
* | SELECT array_join(array[null,'中国','sh'],'/','region')
```

- 查询和分析结果

```
region/中国/sh
```

array_max 函数

array_max 函数用于获取数组中的最大值。

语法

```
array_max(x)
```

参数说明

参数	说明
x	参数值为 array 类型。

返回值类型

与参数值中元素的数据类型一致。

示例

获取数组中[12,23,26,48,26]的最大值48。

- 字段样例

```
array:[12,23,26,48,26]
```

- 查询和分析语句

```
* | SELECT array_max(try_cast(json_parse(array) as array(bigint))) AS max_number
```

- 查询和分析结果

```
48
```

array_min 函数

array_min 函数用于获取数组中的最小值。

语法

```
array_min(x)
```

参数说明

参数	说明
x	参数值为 array 类型。

返回值类型

与参数值中元素的数据类型一致。

示例

获取数组中[12,23,26,48,26]的最小值12。

- 字段样例

```
array:[12,23,26,48,26]
```

- 查询和分析语句

```
* | SELECT array_min(try_cast(json_parse(array) as array(bigint))) AS min_number
```

- 查询和分析结果

```
12
```

array_position 函数

array_position 函数用于获取指定元素的下标，下标从1开始。如果指定元素不存在，则返回0。

语法

```
array_position(x, element)
```

参数说明

参数	说明
x	参数值为数组类型。
element	数组中的一个元素。

返回值类型

bigint 类型。

示例

返回数组[23,46,35]中46的下标。

- 查询和分析语句

```
* | SELECT array_position(array[23,46,35],46)
```

- 查询和分析结果

```
2
```

array_remove 函数

array_remove 函数用于删除数组中指定的元素。

语法

```
array_remove(x, element)
```

参数说明

参数	说明
x	参数值为数组类型。
element	数组中的一个元素。

返回值类型

array 类型。

示例

删除数组[23,46,35]中23。

- 查询和分析语句

```
* | SELECT array_remove(array[23,46,35],23)
```

- 查询和分析结果

```
[46,35]
```

array_sort 函数

array_sort 函数用于对数组元素进行升序排序。

语法

```
array_sort(x)
```

参数说明

参数	说明
x	参数值为 array 类型。

返回值类型

array 类型。

示例

对数组['b', 'd', null, 'c', 'a']进行升序排序。

- 查询和分析语句

```
* | SELECT array_sort(array['b','d',null,'c','a'])
```

- 查询和分析结果

```
["a","b","c","d",null]
```

array_union 函数

array_union 函数用于计算两个数组的并集。

语法

```
array_union(x, y)
```

参数说明

参数	说明
x	参数值为 array 类型。
y	参数值为 array 类型。

返回值类型

array 类型。

示例

计算数组[1,2,3,4,5]和[1,3,5,7]的并集。

- 查询和分析语句

```
* | SELECT array_union(array[1,2,3,4,5],array[1,3,5,7])
```

- 查询和分析结果

```
[1,2,3,4,5,7]
```

cardinality 函数

cardinality 函数用于计算数组中元素的个数。

语法

```
cardinality(x)
```

参数说明

参数	说明
x	参数值为 array 类型。

返回值类型

bigint 类型。

示例

计算 number 字段值中元素的个数。

- 字段样例

```
array:[12,23,26,48,26]
```

- 查询和分析语句

```
* | SELECT cardinality(cast(json_parse(array) as array(bigint)))
```

- 查询和分析结果

```
5
```

concat 函数

concat 函数用于将多个数组拼接成一个数组。

语法

```
concat(x, y...)
```

参数说明

参数	说明
x	参数值为 array 类型。
y	参数值为 array 类型。

返回值类型

array 类型。

示例

将数组['red','blue']和['yellow','green']拼接为一个数组。

- 查询和分析语句

```
* | SELECT concat(array['red','blue'],array['yellow','green'])
```

- 查询和分析结果

```
["red","blue","yellow","green"]
```

contains 函数

contains 函数用于判断数组中是否包含指定元素。如果包含，则返回 true。

语法

```
contains(x, element)
```

参数说明

参数	说明

x	参数值为数组类型。
element	数组中的一个元素。

返回值类型

boolean 类型。

示例

判断 array 字段值中是否包含23。

- 字段样例

```
array:[12,23,26,48,26]
```

- 查询和分析语句

```
* | SELECT contains(cast(json_parse(array) as array(varchar)),'23')
```

- 查询和分析结果

```
TRUE
```

element_at 函数

element_at 函数用于返回数组中的第 y 个元素。

语法

```
element_at(x, y)
```

参数说明

参数	说明
x	参数值为 array 类型。
element	数组下标，从1开始。参数值为 bigint 类型。

返回值类型

任意数据类型。

示例

返回 number 字段值中的第2个元素。

- 字段样例

```
array:[12,23,26,48,26]
```

- 查询和分析语句

```
* | SELECT element_at(cast(json_parse(number) AS array(varchar)), 2)
```

- 查询和分析结果

```
23
```

filter 函数

filter 函数和 Lambda 表达式结合，用于过滤数组中的元素。只返回满足 Lambda 表达式的元素。

语法

```
filter(x, lambda_expression)
```

参数说明

参数	说明
x	参数值为 array 类型。
lambda_expression	Lambda 表达式。更多信息，请参见 Lambda 函数 。

返回值类型

array 类型。

示例

返回数组[5,-6,null,7]中大于0的元素，其中 `x -> x > 0` 为 Lambda 表达式。

- 查询和分析语句

```
* | SELECT filter(array[5,-6,null,7],x -> x > 0)
```

- 查询和分析结果

```
[5,7]
```

flatten 函数

flatten 函数用于将二维数组转换为一维数组。

语法

```
flatten(x)
```

参数说明

参数	说明
x	参数值为 array 类型。

返回值类型

array 类型。

示例

将数组array[1,2,3,4],array[4,3,2,1]转换为一维数组。

- 查询和分析语句

```
* | SELECT flatten(array[array[1,2,3,4],array[4,3,2,1]])
```

- 查询和分析结果

```
[1,2,3,4,4,3,2,1]
```

版权所有：腾讯云计算（北京）有限责任公司

reduce 函数

reduce 函数将根据 Lambda 表达式中的定义，对数组中的各个元素进行相加计算，然后返回计算结果。

语法

```
reduce(x, lambda_expression)
```

参数说明

参数	说明
x	参数值为 array 类型。
lambda_expression	Lambda 表达式。更多信息，请参见 Lambda 函数 。

返回值类型

bigint 类型。

示例

返回数组[5, 20, 50]中各个元素相加的结果。

- 查询和分析语句

```
* | SELECT reduce(array[5,20,50],0,(s, x) -> s + x, s -> s)
```

- 查询和分析结果

```
75
```

reverse 函数

reverse 函数用于对数组中的元素进行反向排列。

语法

```
reverse(x)
```

参数说明

参数	说明
x	参数值为 array 类型。

返回值类型

array 类型。

示例

将数组[1,2,3,4,5]中的元素反向排序。

- 查询和分析语句

```
* | SELECT reverse(array[1,2,3,4,5])
```

- 查询和分析结果

```
[5,4,3,2,1]
```

sequence 函数

sequence 函数通过指定的起始值返回一个数组，其元素为起始值范围内一组连续且递增的值。

语法

- 递增间隔为默认值1。

```
sequence(x, y)
```

- 自定义递增间隔。

```
sequence(x, y, step)
```

参数说明

参数	说明
x	参数值为 bigint 类型、timestamp 类型（Unix 时间戳、日期和时间表达式）。
y	参数值为 bigint 类型、timestamp 类型（Unix 时间戳、日期和时间表达式）。
step	数值间隔。 当参数值为日期和时间表达式时，step 格式如下： • interval 'n' year to month，表示间隔为 n 年。 • interval 'n' day to second，表示间隔为 n 天。

返回值类型

array 类型。

示例

示例1：返回0 – 10之间的偶数。

- 查询和分析语句

```
* | SELECT sequence(0,10,2)
```

- 查询和分析结果

```
[0,2,4,6,8,10]
```

示例2：返回2017-10-23到2021-08-12之间的日期，间隔为1年。

- 查询和分析语句

```
* | SELECT sequence(from_unixtime(1508737026),from_unixtime(1628734085),interval '1' year to month )
```

- 查询和分析结果

```
["2017-10-23 05:37:06.0","2018-10-23 05:37:06.0","2019-10-23 05:37:06.0","2020-10-23 05:37:06.0"]
```

示例3：返回1628733298,1628734085之间的 Unix 时间戳，间隔为60秒。

- 查询和分析语句

```
* | SELECT sequence(1628733298,1628734085,60)
```

- 查询和分析结果

```
[1628733298,1628733358,1628733418,1628733478,1628733538,1628733598,1628733658,1628733718,1628733778,162
8733838,1628733898,1628733958,1628734018,1628734078]
```

shuffle 函数

shuffle 函数用于对数组元素进行随机排列。

语法

```
shuffle(x)
```

参数说明

参数	说明
x	参数值为 array 类型。

返回值类型

array 类型。

示例

对数组[1,2,3,4,5]中的元素进行随机排序。

- 查询和分析语句

```
* | SELECT shuffle(array[1,2,3,4,5])
```

- 查询和分析结果

```
[5,2,4,1,3]
```

slice 函数

slice 函数用于返回数组的子集。

语法

```
slice(x, start, length)
```

参数说明

参数	说明
x	参数值为 array 类型。
start	指定索引开始的位置。 • 如果 start 为负数，则从末尾开始。 • 如果 start 为正数，则从头部开始。
length	指定子集中元素的个数。

返回值类型

array 类型。

示例

返回数组[1,2,4,5,6,7,7]的子集，从第三个元素开始返回，子集元素个数为2。

- 查询和分析语句

```
* | SELECT slice(array[1,2,4,5,6,7,7],3,2)
```

- 查询和分析结果

```
[4,5]
```

transform 函数

transform 函数用于将 Lambda 表达式应用到数组的每个元素中。

语法

```
transform(x, lambda_expression)
```

参数说明

参数	说明
x	参数值为 array 类型。
lambda_expression	Lambda 表达式。更多信息，请参见 Lambda 函数 。

返回值类型

array 类型。

示例

将数组[5,6]中的各个元素加1，然后返回。

- 查询和分析语句

```
* | SELECT transform(array[5,6],x -> x + 1)
```

- 查询和分析结果

```
[6,7]
```

zip 函数

zip 函数将多个数组合并为一个二维数组，且各个数组中下标相同的元素组成一个新的数组。

语法

```
zip(x, y)
```

参数说明

参数	说明
x	参数值为 array 类型。
y	参数值为 array 类型。

返回值类型

array 类型。

示例

将数组[1,2]和[3,4]合并为一个二维数组。

- 查询和分析语句

```
* | SELECT zip(array[1,2], array[3,4])
```

- 查询和分析结果

```
["{1, 3}","{2, 4}"]
```

zip_with 函数

zip_with 函数将根据 Lambda 表达式中的定义将两个数组合并为一个数组。

语法

```
zip_with(x, y, lambda_expression)
```

参数说明

参数	说明
x	参数值为 array 类型。
y	参数值为 array 类型。
lambda_expression	Lambda 表达式。更多信息，请参见 Lambda 函数 。

返回值类型

array 类型。

示例

使用 Lambda 表达式 `(x, y) -> x + y` 使数组[1,2]和[3,4]中的元素分别相加后，以数组类型返回相加的结果。

- 查询和分析语句

```
* | SELECT zip_with(array[1,2], array[3,4],(x,y) -> x + y)
```

- 查询和分析结果

```
[4,6]
```

类型转换函数

最近更新时间：2025-05-28 10:55:32

本文介绍类型转换函数的基本语法及示例。
如果您在查询与分析数据时，需要区分更细维度的数据类型，您可以在查询与分析语句中使用类型转换函数转换数据的数据类型。

函数名称	语法	说明
cast 函数	cast(x as type)	转换 x 的数据类型。 使用 cast 函数转换数据类型时，如果某个值转换失败，将终止整个查询与分析操作。
try_cast 函数	try_cast(x as type)	转换 x 的数据类型。 使用 try_cast 函数转换数据类型时，如果某个值转换失败，该值返回 NULL，并跳过该值继续处理。
typeof 函数	typeof(x)	返回 x 的数据类型。

说明：
日志中可能有脏数据时，建议使用 try_cast 函数，避免因脏数据造成整个查询与分析操作失败。

cast 函数

cast 函数用于转换 x 的数据类型。使用 cast 函数转换数据类型时，如果某个值转换失败，将终止整个查询与分析操作。

语法

```
cast(x as type)
```

参数说明

参数	说明
x	参数值可以为任意类型。
type	SQL 数据类型，可选值为 bigint、varchar、double、boolean、timestamp、decimal、array 或 map。 索引数据类型和 SQL 数据类型的映射关系，请参见 附录：数据类型映射关系 。

说明：
- type 为 timestamp 时，x 需为毫秒级时间戳或 ISO 8601格式的时间字符串，例如：1597807109000或2019-12-25T16:17:01+08:00。
- 将UNIX时间戳转为timestamp时，采用UTC+0时区，如需转换为其它时区可使用具备指定时区功能的函数（例如 `from_unixtime(__TIMESTAMP__/1000, 'Asia/Shanghai')` ）或手动添加时区偏移（例如 `cast(__TIMESTAMP__+8*60*60*1000 as timestamp)` 或 `cast(__TIMESTAMP__ as timestamp)+ INTERVAL 8 HOUR` ）。

返回值类型

由您配置的 type 参数决定。

示例

1. 将数值0.01转换为 bigint 格式。

```
* | select cast(0.01 as bigint)
```

2. 将日志服务附带的日志采集时间 `__TIMESTAMP__` 转换为 `TIMESTAMP` 类型 。

```
* | select cast(__TIMESTAMP__ as timestamp)
```

try_cast 函数

try_cast 函数用于转换x的数据类型。使用 try_cast 函数转换数据类型时，如果某个值转换失败，该值返回 NULL，并跳过该值继续处理。

语法

```
try_cast(x as type)
```

参数说明

参数	说明
x	参数值可以为任意类型。
type	SQL 数据类型，可选值为 bigint、varchar、double、boolean、timestamp、decimal、array 或 map。 索引数据类型和 SQL 数据类型的映射关系，请参见 附录：数据类型映射关系 。

返回值类型

由您配置的 type 参数决定。

示例

将 **remote_user** 字段值转换为 varchar 格式。

```
* | select try_cast(remote_user as varchar)
```

附录：数据类型映射关系

索引数据类型和 SQL 数据类型的对应关系如下表所示：

索引的数据类型	SQL 的数据类型
long	bigint
text	varchar
double	double
json	varchar

条件表达式

最近更新时间：2024-08-15 15:04:21

本文介绍条件表达式的语法和示例。

表达式	语法	说明
CASE WHEN 表达式	CASE WHEN condition1 THEN result1 [WHEN condition2 THEN result2] [ELSE result3] END	过条件判断，对数据进行归类。
IF 表达式	IF(condition, result1)	如果 condition 为 true，则返回 result1，否则返回 null。
	IF(condition, result1, result2)	如果 condition 为 true，则返回 result1，否则返回 result2。
NULLIF 表达式	NULLIF(expression1, expression2)	比较两个表达式的值是否相等。如果相等，则返回 null，否则返回第一个表达式的值。
TRY 表达式	TRY(expression)	捕获异常信息，使得系统继续执行查询和分析操作。
COALESCE 表达式	COALESCE(expression1, expression2...)	获取多个表达式中的第一个非 NULL 值。

CASE WHEN 语法

CASE WHEN 语法用于对数据进行归类。

语法

```
CASE WHEN condition1 THEN result1
            [WHEN condition2 THEN result2]
            [ELSE result3]
END
```

参数说明

参数	说明
condition	条件表达式。
result	返回结果。

示例

示例1

从 http_user_agent 字段值中提取浏览器信息，归为 Chrome、Safari 和 unknown 三种类型并计算三种类型对应的访问 PV。

```
* |
SELECT
  CASE
    WHEN http_user_agent like '%Chrome%' then 'Chrome'
    WHEN http_user_agent like '%Safari%' then 'Safari'
    ELSE 'unknown'
  END AS http_user_agent,
  count(*) AS pv
GROUP BY
  http_user_agent
```

示例2

统计不同请求时间的分布情况。

```
* |
SELECT
  CASE
    WHEN request_time < 0.001 then 't0.001'
    WHEN request_time < 0.01 then 't0.01'
    WHEN request_time < 0.1 then 't0.1'
    WHEN request_time < 1 then 't1'
    ELSE 'overtime'
  END AS request_time,
  count(*) AS pv
GROUP BY
  request_time
```

IF 语法

IF 语法用于对数据进行归类，类似于 CASE WHEN 表达式。

语法

- 如果 condition 为 true，则返回 result1，否则返回 null。

```
IF(condition, result1)
```

- 如果 condition 为 true，则返回 result1，否则返回 result2。

```
IF(condition, result1, result2)
```

参数说明

参数	说明
condition	条件表达式。
result	返回结果。

示例

计算状态码为200的请求占所有请求的比例。

```
* |
SELECT
  sum(IF(status = 200, 1, 0)) * 1.0 / count(*) AS status_200_percentag
```

NULLIF 表达式

NULLIF 表达式用于比较两个表达式的值是否相等。如果相等，则返回 null，否则返回第一个表达式的值。

语法

```
NULLIF(expression1, expression2)
```

参数说明

参数	说明
expression	任何有效的标量表达式。

版权所有：腾讯云计算（北京）有限责任公司

示例

判断 server_addr、http_host 两个字段的值是否相同。当不相同时，返回 server_addr 字段的值。

```
* | SELECT NULLIF(server_addr,http_host)
```

TRY 表达式

TRY 表达式用于捕获异常信息，使得系统继续执行查询和分析操作。

语法

```
TRY(expression)
```

参数说明

参数	说明
expression	任何类型的表达式。

示例

当执行 regexp_extract 函数发生异常时，try函数会捕获异常信息并继续查询和分析操作，返回查询和分析结果。

```
* |
SELECT
  TRY(regexp_extract(uri, '.*\/(index.*)', 1))
    AS file, count(*)
    AS count
GROUP BY
  file
```

COALESCE 表达式

COALESCE 表达式用于获取多个表达式中的第一个非 NULL 值。

语法

```
COALESCE(expression1, expression2...)
```

参数说明

参数	说明
expression	任何有效的标量表达式。

示例

```
* | select COALESCE(null, 'test')
```

数学计算函数

最近更新时间：2024-07-11 16:43:41

本文介绍数学计算函数的语法及示例。
日志服务（Cloud Log Service，CLS）日志分析功能支持对于以 int、long、double 为类型的字段通过数学统计函数和数学计算函数进行日志分析。

说明
- 数学计算函数支持运算符+-*/%
- 以下函数中的 x、y 可以为数字、日志字段或计算结果为数字的表达式

基本语法

函数名称	说明
abs(x)	计算数字的绝对值。
cbrt(x)	计算数字的立方根。
sqrt(x)	计算数字的平方根。
cosine_similarity(x,y)	计算向量 x 和 y 之间的余弦相似度。 例如 `* | SELECT cosine_similarity(MAP(ARRAY['x','y'], ARRAY[1.0,0.0]), MAP(ARRAY['x','y'], ARRAY[0.0,1.0]))` 返回0。
degrees(x)	将弧度转换为度。
radians(x)	将度转换为弧度。
e()	计算数字的自然对数。
exp(x)	返回自然对数的指数。
ln(x)	计算数字的自然对数。
log2(x)	计算以2为底的对数。
log10(x)	计算以10为底的对数。
log(x,b)	计算以 b 为底的对数。
pi()	返回包含14个小数位的 π 值。
pow(x,b)	计算数字的 b 次幂。
rand()	返回随机数。
random(0,n)	返回[0,n)之间的随机数。
round(x)	返回四舍五入后的取值。
round(x, N)	保留数字的 N 位小数。
floor(x)	向下取整数。
ceiling(x)	向上取整数。
truncate(x)	截断数字的小数部分。
acos(x)	计算数字的反余弦。
asin(x)	计算数字的反正弦。

atan(x)	计算数字的反正切。
atan2(y,x)	计算两个数字相除的结果的反正切。
cos(x)	计算数字的余弦。
sin(x)	计算数字的正弦。
cosh(x)	计算数字的双曲余弦。
tan(x)	计算数字的正切。
tanh(x)	计算数字的双曲正切。
infinity()	返回正无穷的数值。
is_nan(x)	判断目标值是否为 NaN（Not a Number）。
nan()	返回一个 NaN 值（Not a Number）。
mod(x, y)	用于计算 x 与 y 相除的余数。
sign(x)	返回 x 的符号，通过1、0、-1表示。
width_bucket(x, bound1, bound2, n)	将一段数值（bound1 – bound2）划分成大小相同的 n 个 Bucket，返回 x 所属的 Bucket。 例如 `* | select timeCost,width_bucket(timeCost,10,1000,5)`
width_bucket(x, bins)	使用数组（bins）指定 Bucket 的范围，返回 x 所属的 Bucket。 例如 `* | select timeCost,width_bucket(timeCost,array[10,100,1000])`
to_base(x, radix)	根据 BASE 编码将 x 转为 radix 进制的字符串。
from_base(x, radix)	根据 BASE 编码将字符串 string 转为 radix 进制的数字。

示例

同比今天和昨天的访问 PV，并使用百分数表示。查询和分析语句如下：

```
* | SELECT diff [1] AS today, round((diff [3] -1.0) * 100, 2) AS growth FROM (SELECT compare(pv, 86400) as
diff FROM (SELECT COUNT(*) as pv FROM log))
```

数学统计函数

最近更新时间：2024-07-11 16:43:41

本文介绍数学统计函数的语法及示例。

基本语法

函数名称	说明
corr(key1, key2)	计算两列的相关度。计算结果范围为[0,1]。
covar_pop(key1, key2)	计算两列的总体协方差。
covar_samp(key1, key2)	计算两列的样本协方差。
regr_intercept(key1, key2)	返回输入值的线性回归截距。key1 是依赖值，key2 是独立值。
regr_slope(key1, key2)	返回输入值的线性回归斜率。key1 是依赖值，key2 是独立值。
stddev(key)	计算 key 列的样本标准差。与 stddev_samp 函数同义。
stddev_samp(key)	计算 key 列的样本标准差。
stddev_pop(key)	返回 key 列的总体标准差。
variance(key)	计算 key 列的样本方差，与 var_samp 函数同义。
var_samp(key)	计算 key 列的样本方差。
var_pop(key)	计算 key 列的总体方差。

示例

计算两列数据的相关度。查询和分析语句如下：

```
* | select corr(timeCost,SamplingRate)
```

二进制字符串函数

最近更新时间：2024-07-11 16:43:41

本文将为您介绍二进制字符串函数的语法。
二进制字符串类型 varbinary 有别于字符串类型 varchar。

语句	说明
连接函数 ||	`a || b` 结果为 `ab` 。
length(binary) → bigint	返回二进制的长度。
concat(binary1, …, binaryN) → varbinary	连接二进制字符串，等同于||。
to_base64(binary) → varchar	把二进制字符串转换成 base64。
from_base64(string) → varbinary	把 base64 转换成二进制字符串。
to_base64url(binary) → varchar	转化成 URL 安全的 base64。
from_base64url(string) → varbinary	从 URL 安全的 base64 转化成二进制字符串。
to_hex(binary) → varchar	把二进制字符串转化成十六进制表示。
from_hex(string) → varbinary	从十六进制转化成二进制。
to_big_endian_64(bigint) → varbinary	把数字转化成大端表示的二进制。
from_big_endian_64(binary) → bigint	把大端表示的二进制字符串转化成数字。
md5(binary) → varbinary	计算二进制字符串的 md5。
sha1(binary) → varbinary	计算二进制字符串的 sha1。
sha256(binary) → varbinary	计算二进制字符串的 sha256 hash。
sha512(binary) → varbinary	计算二进制字符串的 sha512。
xxhash64(binary) → varbinary	计算二进制字符串的 xxhash64。

位运算函数

最近更新时间：2024-05-29 14:28:51

本文介绍位运算函数的基本语法及示例。

函数名称	语句	说明
bit_count 函数	bit_count(x, bits)	统计 x 的二进制表示中1的个数。
bitwise_and 函数	bitwise_and(x, y)	以二进制的形式对 x，y 进行与运算。
bitwise_not 函数	bitwise_not(x)	以二进制的形式对 x 的所有位进行取反运算。
bitwise_or 函数	bitwise_or(x, y)	以二进制形式对 x，y 进行或运算。
bitwise_xor 函数	bitwise_xor(x, y)	以二进制形式对 x，y 进行异或运算。

bit_count 函数

bit_count 函数用于统计 x 中1的个数。

语法

```
bit_count(x, bits)
```

参数说明

参数	说明
x	参数值为 bigint 类型。
bits	位数，例如64位。

返回值类型

bigint 类型

示例

计算数字24的二进制数，并返回其二进制数中1的个数。

- 查询和分析语句

```
* | SELECT bit_count(24, 64)
```

- 查询和分析结果

```
2
```

bitwise_and 函数

bitwise_and 函数以二进制形式对 x 和 y 进行与运算。

语法

```
bitwise_and(x, y)
```

参数说明

参数	说明

x	参数值为 bigint 类型。
y	参数值为 bigint 类型。

返回值类型

bigint 类型

示例

以二进制形式对数字3和5进行与运算。

- 查询和分析语句

```
* | SELECT bitwise_and(3, 5)
```

- 查询和分析结果

```
1
```

bitwise_not 函数

bitwise_not 函数以二进制形式对 x 的所有位进行取反运算。

语法

```
bitwise_not(x)
```

参数说明

参数	说明
x	参数值为 bigint 类型。

返回值类型

bigint 类型

示例

以二进制形式对数字4的所有位进行取反运算。

- 查询和分析语句

```
* | SELECT bitwise_not(4)
```

- 查询和分析结果

```
-5
```

bitwise_or 函数

bitwise_or 函数以二进制形式对 x 和 y 进行或运算。

语法

```
bitwise_or(x, y)
```

参数说明

参数	说明
x	参数值为 bigint 类型。
y	参数值为 bigint 类型。

返回值类型

bigint 类型

示例

以二进制形式对数字3和5进行或运算。

- 查询和分析语句

```
* | SELECT bitwise_or(3, 5)
```

- 查询和分析结果

```
7
```

bitwise_xor 函数

bitwise_xor 函数以二进制形式对 x 和 y 进行异或运算。

语法

```
bitwise_xor(x, y)
```

参数说明

参数	说明
x	参数值为 bigint 类型。
y	参数值为 bigint 类型。

返回值类型

bigint 类型

示例

以二进制形式对数字3和5进行异或运算。

- 查询和分析语句

```
* | SELECT bitwise_xor(3, 5)
```

- 查询和分析结果

```
6
```

Lambda 函数

最近更新时间：2024-05-29 14:28:51

本文将为您介绍 Lambda 函数对日志数据进行分析处理的语法及示例。
日志服务支持您在 SQL 分析语句中定义 Lambda 表达式，并将该表达式传递给指定函数，丰富函数的表达。

语法

Lambda 表达式需与函数一起使用，例如 filter 函数、reduce 函数、transform 函数、zip_with 函数。Lambda 表达式的语法如下：

```
parameter -> expression
```

参数	说明
parameter	用于传递参数的标识符。

expression	表达式，大多数的 Mysql 表达式都可以在 Lambda 表达式使用。例如：
	`x -> x + 1`
	`(x, y) -> x + y`
	`x -> regexp_like(x, 'a+')`
	`x -> x[1] / x[2]`
	`x -> if(x > 0, x, -x)`
	`x -> coalesce(x, 0)`
	`x -> cast(x AS JSON)`
	`x -> x + try(1 / 0)`

示例

示例1：使用 Lambda 表达式 x-> x is not null

返回数组[5，null，7，null]中非 null 的元素。

- 查询和分析语句

```
* | SELECT filter(array[5, null, 7, null], x -> x is not null)
```

- 查询和分析结果

```
[5,7]
```

示例2：使用 Lambda 表达式0，(s, x) -> s + x, s -> s

返回数组[5，20，50]中各个元素相加的结果。

- 查询和分析语句

```
* | SELECT reduce(array[5, 20, 50], 0, (s, x) -> s + x, s -> s)
```

- 查询和分析结果

```
75
```

示例3：使用 Lambda 表达式(k, v) -> v> 10

将两个数组映射为一个 Map 且 Map 中的键值大于10。

- 查询和分析语句

```
* | SELECT map_filter(map(array['class01', 'class02', 'class03'], array[11, 10, 9]), (k,v) -> v > 10)
```

- 查询和分析结果

```
{"class01":11}
```

示例4：使用 Lambda 表达式(x, y) -> (y, x)

将对换两个数组的元素位置，并提取数组中索引相同的元素组成一个新的二维数组。

- 查询和分析语句

```
* | SELECT zip_with(array['a', 'b', 'c'], array['d', 'e', 'f'], (x, y) -> concat(x, y))
```

- 查询和分析结果

```
["ad","be","cf"]
```

示例5：使用 Lambda 表达式 x -> coalesce（x, 0）+1

将数组[5, null, 6]的各个元素加1，然后返回。如果数组中包含 null 元素，则转换为0，再加1。

- 查询和分析语句

```
* | SELECT transform(array[5, NULL, 6], x -> coalesce(x, 0) + 1)
```

- 查询和分析结果

```
[6,1,7]
```

其他示例

```
* | SELECT filter(array[], x -> true)
* | SELECT map_filter(map(array[],array[]), (k, v) -> true)
* | SELECT reduce(array[5, 6, 10, 20], -- calculates arithmetic average: 10.25
            cast(row(0.0, 0) AS row(sum double, count integer)),
            (s, x) -> cast(row(x + s.sum, s.count + 1) AS row(sum double, count integer)),
            s -> if(s.count = 0, null, s.sum / s.count))
* | SELECT reduce(array[2147483647, 1], cast(0 AS bigint), (s, x) -> s + x, s -> s)
* | SELECT reduce(array[5, 20, null, 50], 0, (s, x) -> s + x, s -> s)
* | SELECT transform(array[array[1, null, 2], array[3, null]], a -> filter(a, x -> x is not null))
```

窗口函数

最近更新时间：2024-08-15 15:04:21

本文介绍窗口函数的基础语法和示例。
窗口函数用于对指定的多行数据进行计算并返回计算的结果，与 GROUP BY 的区别在于其仅会将计算结果附加到每一行数据上，而不会对行本身进行合并。

语法

```
window_function (expression) OVER (
   [ PARTITION BY part_key ]
   [ ORDER BY order_key ]
   [ { ROWS | RANGE } BETWEEN frame_start AND frame_end ] )
```

参数说明

参数	说明
window_function	窗口值计算方法，支持通用聚合函数、排序函数和取值函数。
PARTITION BY	窗口分区依据。
ORDER BY	窗口分区内多行数据排序依据。
{ ROWS | RANGE } BETWEEN frame_start AND frame_end	窗口帧，即窗口分区内每行数据计算值时使用到的数据范围（行），未指定时代表窗口分区内的所有行。 使用示例： rows between current row and 1 following：当前行及后一行。 rows between 1 preceding and current row：当前行及前一行。 rows between 1 preceding and 1 following：前一行至后一行（共三行）。 rows between current row and unbounded following：当前行及后续所有行。 rows between unbounded preceding and current row：当前行及前面的所有行。

通用聚合函数

支持所有的 通用聚合函数，例如 sum()，avg()等，通用聚合函数将针对每行数据计算其在窗口帧内的统计值。

排序函数

排序函数不能使用窗口帧

函数	说明
rank()	对每个窗口分区分别进行排名并返回值。相同的值拥有相同的排名，所以排名可能不是连续的。例如，有两个相同值的排名为1，则下一个值的排名为3。
dense_rank()	与 rank() 类似，区别在于该函数排名是连续的。例如，有两个相同值的排名为1，则下一个值的排名为2。
cume_dist()	统计窗口分区内各个值的累计分布，即窗口分区内值小于等于当前值的行数占窗口内总行数的比例。
ntile(n)	将窗口分区内数据按照顺序分为n组，如果分区内的行数没有被平均分成 n 组，则从第一组开始，每组分配一个剩余值。例如：数据有6行，需要分为4组，则每行数据的组号为 1、1、2、2、3、4。
percent_rank()	计算每行数据在窗口分区内的百分比排名，计算方式为(r - 1) / (n - 1)，其中r为通过 rank() 获取到排名值，n为窗口分区内的总行数
row_number()	返回窗口分区内每行数据（根据排序规则排序后）的序号，从1开始，每行均为唯一值。

取值函数

函数	说明
first_value(key)	返回窗口分区内的第一个值。

last_value(key)	返回窗口分区内最后一个值。
nth_value(key, offset)	返回窗口分区内指定偏移后的值，offset 从1开始计算。如果 offset 为 null 或超过了窗口分区内行的数量，则返回null。offset 不允许为0或负数。
lead(key[, offset[, default_value]])	返回窗口分区内当前行向后指定偏移后的值，offset 从0开始计算，即当前行。offset 默认为1，如果 offset 为 null，则返回 null。如果偏移后的行超出了窗口分区，则返回 default_value，未指定时返回 null。 使用该函数时必须制定窗口分区内的排序规则（ORDER BY），并且不能使用窗口帧。
lag(key[, offset[, default_value]])	与 lead(key[, offset[, default_value]]) 函数类似，区别在于该函数会返回当前行向前指定偏移后的值。

示例

示例1：查询最近1小时每个接口最慢的5个请求及其请求 ID

选择时间范围为近1小时，并执行如下查询和分析语句，其中 action 为接口名称，timeCost 为接口响应时间，seqId 为请求ID。

- 查询和分析语句

```
* | select * from (select action,timeCost,seqId,rank() over (partition by action order by timeCost
desc) as ranking order by action,ranking,seqId) where ranking<=5 limit 10000
```

- 查询和分析结果

action	timeCost	seqId	ranking
ModifyXXX	151	d75427b3-c562-6d7a-354f-469963aab689	1
ModifyXXX	104	add0d353-1099-2c73-e9c9-19ad02480474	2
CreateXXX	1254	c7d591f0-2da6-292c-8abf-98a0716ff8c6	1
CreateXXX	970	d920cf7a-7e7b-524b-68e9-a957c454c328	2
CreateXXX	812	16357f6d-33b3-83ea-0ae3-b1a2233d4858	3
CreateXXX	795	0efdab5e-af5f-4a4a-0618-7961420d17a1	4
CreateXXX	724	fb0481f2-dcfc-9500-cb44-a139b774aceb	5
DescribeXXX	55242	4129dcda-46d7-9213-510e-f58cba29daf5	1
DescribeXXX	17413	e36cdeb0-cbc5-ce2b-dec7-f485818ab6c7	2
DescribeXXX	10171	cd6228f7-4644-ba45-f539-0fce7b09455b	3
DescribeXXX	9475	48b6f6e3-6d08-5a31-cd68-89006a346497	4
DescribeXXX	9337	940b5398-e2ae-9141-801b-b7f0ca548875	5

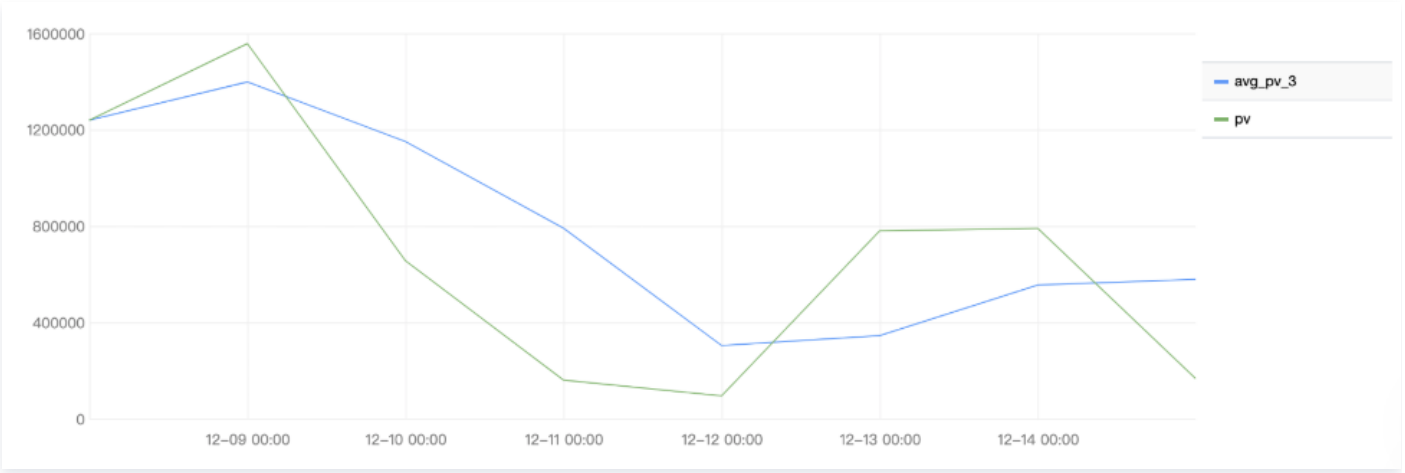
示例2：查询应用吞吐量的3日移动平均值变化趋势

选择查询和分析的时间范围为最近7天，并执行如下查询和分析语句，其中 pv 为每天的应用吞吐量，avg_pv_3 是3天移动平均后的应用吞吐量。

- 查询和分析语句

```
* | select avg(pv) over(order by analytic_time rows between 2 preceding and current row) as
avg_pv_3,pv,analytic_time from (select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 day) as
analytic_time, count(*) as pv group by analytic_time order by analytic_time)
```

- 查询和分析结果


1600000
1200000
800000
400000
0
12-09 00:00
12-10 00:00
12-11 00:00
12-12 00:00
12-13 00:00
12-14 00:00
avg_pv_3
pv

空间几何函数

最近更新时间：2024-07-11 16:43:41

本文将为您介绍空间几何函数。

基本概念

空间几何函数支持 Well-Known Text（WKT）及 Well-Known Binary（WKB）格式描述的几何实体，相关概念可参考 Well-known text representation of geometry - Wikipedia 。

几何实体	Well-Known Text（WKT）格式
点	POINT (0 0)
线段	LINESTRING (0 0, 1 1, 1 2)
多边形	POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))
多点	MULTIPOINT (0 0, 1 2)
多线段	MULTILINESTRING ((0 0, 1 1, 1 2), (2 3, 3 2, 5 4))
多个多边形	MULTIPOLYGON (((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1)), ((-1 -1, -1 -2, -2 -2, -2 -1, -1 -1)))
几何实体集合	GEOMETRYCOLLECTION (POINT(2 3), LINESTRING (2 3, 3 4))

几何实体默认采用平面几何，在平面几何中两点之间的最短距离为直线。几何实体也支持球面几何，球面中两点之间的最短距离为大圆弧。使用 to_spherical_geography() 可将平面几何实体转换为球面几何实体。
例如：
`ST_Distance(ST_Point(-71.0882, 42.3607), ST_Point(-74.1197, 40.6976))` 计算平面上的两个点的距离，值为3.4577。
`ST_Distance(to_spherical_geography(ST_Point(-71.0882, 42.3607)), to_spherical_geography(ST_Point(-74.1197, 40.6976)))` 计算球面上两个点的距离，值为312822.179。
计算长度时（例如：ST_Distance()和ST_Length()），单位为米，计算面积时（例如：ST_Area()），单位为平方米。

构造几何实体

函数	返回值类型	说明
ST_Point(double, double)	Point	构造一个点。
ST_LineFromText(varchar)	LineString	从 WKT 格式的文本中构造一个线段。
ST_Polygon(varchar)	Polygon	从 WKT 格式的文本中构造一个多边形。
ST_GeometryFromText(varchar)	Geometry	从 WKT 文本中构造一个几何实体。
ST_GeomFromBinary(varbinary)	Geometry	从 WKB 中构造一个几何实体。
ST_AsText(Geometry)	varchar	把一个空间几何实体转变成 WKT 格式。
to_spherical_geography(Geometry)	SphericalGeography	将平面几何实体转换为球面几何实体。
to_geometry(SphericalGeography)	Geometry	将球面几何实体转换为平面几何实体。

空间关系判断

函数	返回值类型	说明
ST_Contains(Geometry, Geometry)	boolean	当第二个实体的所有点都不在第一个实体外部，并且第一个实体至少有一个内部点在第二个实体内部时，返回 true。如果第二个实体正好在第一个实体的边上，返回 false。
ST_Crosses(Geome	boole	当两个实体有共同内部点时，返回 true。

try, Geometry)	an	
ST_Disjoint(Geometry, Geometry)	boolean	当两个实体没有任何交集时，返回 true。
ST_Equals(Geometry, Geometry)	boolean	当两个实体完全相同时，返回 true。
ST_Intersects(Geometry, Geometry)	boolean	当两个实体在两个空间上共享时，返回 true。
ST_Overlaps(Geometry, Geometry)	boolean	当两个实体维度相同，并且不是包含关系时，返回 true。
ST_Relate(Geometry, Geometry)	boolean	当两个实体相关时，返回 true。
ST_Touches(Geometry, Geometry)	boolean	当两个实体仅仅边界有联系，没有共同内部点时，返回 true。
ST_Within(Geometry, Geometry)	boolean	当第一个实体完全在第二个实体内部时，返回 true。如果边界有交集，返回 false。

Operations

函数	返回值类型	说明
geometry_nearest_points(Geometry, Geometry)	row(Point, Point)	返回两个几何实体之间最相近的两个点。
geometry_union(array(Geometry))	Geometry	将多个几何实体合并为一个几何实体。
ST_Boundary(Geometry)	Geometry	返回几何实体的闭包。
ST_Buffer(Geometry, distance)	Geometry	返回一个多边形，该多边形距离输入参数 Geometry 的距离是 distance。
ST_Difference(Geometry, Geometry)	Geometry	返回两个空间实体的不同的点的集合。
ST_Envelope(Geometry)	Geometry	返回空间实体的边界多边形。
ST_ExteriorRing(Geometry)	Geometry	返回多边形的外部环。
ST_Intersection(Geometry, Geometry)	Geometry	返回两个空间实体的交集点。
ST_SymDifference(Geometry, Geometry)	Geometry	返回两个空间实体不同的点，组成的新的空间实体。获取两个几何对象不相交的部分。

Accessors

函数	返回值类型	说明
ST_Area(Geometry)	double	在平面几何中计算多边形面积。
ST_Area(SphericalGeography)	double	在球面几何中计算多边形面积。
ST_Centroid(Geometry)	Geometry	返回几何实体的中心点。
ST_CoordDim(Geometry)	bigint	返回几何实体的坐标维度。
ST_Dimension(Geometry)	bigint	返回几何实体的固有维度，必须小于或等于坐标维度。
ST_Distance(Geometry, Geometry)	double	计算平面几何中两个实体之间的最小距离。
ST_Distance(SphericalGeography, SphericalGeography)	double	计算球面几何中两个实体之间的最小距离。

版权所有：腾讯云计算（北京）有限责任公司

ST_IsClosed(Geometry)	boolean	当实体是一个闭合空间时，返回 true。
ST_IsEmpty(Geometry)	boolean	当参数是一个空的几何实体集合或者多边形或者点时，返回 true。
ST_IsRing(Geometry)	boolean	当参数是一条线，并且是闭合的简单的线时，返回 true。
ST_Length(Geometry)	double	在平面几何中，计算一个线段或者多条线段的长度。
ST_Length(SphericalGeography)	double	在球面几何中，计算一个线段或者多条线段的长度。
ST_XMax(Geometry)	double	返回几何体边框的 X 最大值。
ST_YMax(Geometry)	double	返回几何体边框的 Y 最大值。
ST_XMin(Geometry)	double	返回几何体边框的 X 最小值。
ST_YMin(Geometry)	double	返回几何体边框的 Y 最小值。
ST_StartPoint(Geometry)	point	返回线段类型几何体的第一个点。
ST_EndPoint(Geometry)	point	返回线段类型几何体的最后一个点。
ST_X(Point)	double	返回点类型的 X 轴。
ST_Y(Point)	double	返回点类型的 Y 轴。
ST_NumPoints(Geometry)	bigint	计算几何实体的点的个数。
ST_NumInteriorRing(Geometry)	bigint	返回多边形内部的环的个数。

版权所有：腾讯云计算（北京）有限责任公司

SQL 案例

最近更新时间：2025-03-03 14:50:32

操作场景

使用 SQL 对日志进行统计分析极为灵活，能够满足绝大多数的统计分析需求，但是 SQL 语法较为复杂，不易上手，本文介绍常用 SQL 案例以供参考。

样例数据

本文以 Nginx Demo 日志为例，介绍常用 SQL。您可参考 使用 Demo 日志快速体验 CLS 开启 Nginx Demo 日志主题，实际执行本文中的 SQL。Nginx Demo 日志内容如下：

```
{
    "remote_addr": "124.78.118.162",
    "method": "GET",
    "upstream_addr": "169.254.128.14:60002",
    "upstream_response_length": "48",
    "body_bytes_sent": "59",
    "time_local": "2021-12-18T05:16:06+00:00",
    "version": "HTTP/1.1",
    "url": "/_ignition/execute-solution",
    "http_user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/78.0.3904.108 Safari/537.36",
    "remote_user": "-",
    "req_id": "37585e290318470aa57b148282aed99f",
    "upstream_status": "400",
    "request_time": "0.002",
    "sys_address": "11.149.171.195",
    "request_length": "272",
    "http_referer": "-",
    "sys_datasource": "gz.1.1.v1.2.19",
    "proxy_upstream_name": "default-kubernetes-443",
    "upstream_response_time": "0.000",
    "time": "1706701448003",
    "timestamp": "2024-01-31T19:44:08+08:00",
    "status": "400"
}
```

运维监控类案例

请求数 Top URL

```
* | select count(*) as "请求数量",
    url
    group by url
    order by "请求数量" desc
    limit 50
```

语句解释：

- `count(*)`：统计所有日志条数，即请求数量。
- `group by url`：按照 url 分组，统计每个 url 对应的请求数量。
- `order by "请求数量" desc`：按照请求数量降序排序，优先展示请求数量多的 url。
- `limit 50`：只返回查询结果中的前50条，即请求数量最多的50个 url。

请求数 Top URL（去除 URL 中的参数部分）

```
* | select count(*) as "请求数量",
```

```
    url_extract_path(url) as url_path
    group by url_path
    order by "请求数量" desc
    limit 50
```

语句解释：

- `url_extract_path(url)`：去除 url 中的参数部分，仅保留 path，避免 url 参数变化影响统计。
- 其它语法含义同上例。

错误请求占比

```
* | select round(count_if(status >= 400)*1.0 / count(*),2) as "错误请求占比"
```

语句解释：

- `count_if(status >= 400)`：统计满足指定条件的日志条数，此处以 http 状态码 status >= 400代表错误请求。
- `*1.0`：乘以1.0是为了将 count_if 的结果转换为 double 类型，以便后续进行除法运算时保留精度，否则整数相除结果仍为整数，将丢失精度。
- `count(*)`：统计所有日志条数，即总请求数量。
- `round`：保留小数位数，此处为保留2位小数。

错误请求数量按 URL 分布

```
status >= 400 | select count(*) as "错误请求数量",url
    group by url
    order by "错误请求数量" desc
    limit 50
```

语句解释：

- `status >= 400`：管道符 `|` 前的语句为检索条件，该条件代表仅对 http 状态码 status >= 400的日志进行统计分析。
- `count(*)`：统计所有日志条数，因为管道符 `|` 前已有检索条件，此处实际上统计的是所有错误请求日志条数。
- `group by url`：按照 url 分组，统计每个 url 对应的错误请求数量。
- `order by "错误请求数量" desc`：按照错误请求数量降序排序，优先展示错误请求数量多的 url。
- `limit 50`：只返回查询结果中的前50条，即错误请求数量最多的50个 url。

错误请求占比变化趋势

```
* | select round(count_if(status >= 400)*1.0 / count(*),2) as "错误请求占比",
    histogram(__TIMESTAMP__,INTERVAL ${__interval}) as time
    group by time limit 10000
```

语句解释：

- `count_if(status >= 400)`：统计满足指定条件的日志条数，此处以 http 状态码 status >= 400代表错误请求。
- `*1.0`：乘以1.0是为了将 count_if 的结果转换为 double 类型，以便后续进行除法运算时保留精度，否则整数相除结果仍为整数，将丢失精度。
- `count(*)`：统计所有日志条数，即总请求数量。
- `round`：保留小数位数，此处为保留2位小数。
- `histogram(__TIMESTAMP__,INTERVAL ${__interval})`：按时间查看指标变化趋势时，需指定趋势图中每个数据点之间的间隔，例如查询最近60分钟的变化趋势，数据点之间的间隔为30秒。此处：
  - `__TIMESTAMP__` 代表日志时间，即请求的发生时间。
  - `${__interval}` 代表数据点之间的间隔，是一个变量，将根据查询时间范围的长度动态计算最合适的间隔。也可手动指定，例如修改为 `1 MINUTE`，代表间隔为1分钟。
- `group by time`：使用 histogram 指定好时间间隔后，按该间隔分别统计每个间隔的错误请求占比。
- `limit 10000`：SQL 未指定 limit 时，默认仅返回100条查询结果，为获取完整的查询结果，将 limit 指定为更大的值。

各状态码请求数量变化趋势

```
* | select count(*) as cnt,status,
    histogram(__TIMESTAMP__,interval ${__interval}) as time
    group by time,status limit 10000
```

语句解释：

- `count(*)`：统计所有日志条数，即总请求数量。
- `histogram(__TIMESTAMP__,INTERVAL ${__interval})`：按时间查看指标变化趋势时，需指定趋势图中每个数据点之间的间隔，例如查询最近60分钟的变化趋势，数据点之间的间隔为30秒。此处：
  - `__TIMESTAMP__` 代表日志时间，即请求的发生时间。
  - `${__interval}` 代表数据点之间的间隔，是一个变量，将根据查询时间范围的长度动态计算最合适的间隔。也可手动指定，例如修改为 `1 MINUTE`，代表间隔为1分钟。
- `group by time`：使用 histogram 指定好时间间隔后，按该间隔分别统计每个间隔的错误请求占比。
- `limit 10000`：SQL 未指定 limit 时，默认仅返回100条查询结果，为获取完整的查询结果，将 limit 指定为更大的值。

各状态码请求占比变化趋势

```
* | select time,status,
    cnt*1.0/sum(cnt) over (partition by time) as pct
    from (
      select count(*) as cnt,status,
      histogram(__TIMESTAMP__,interval ${__interval}) as time
      group by time,status)
    limit 10000
```

语句解释：

- `select count(*) as cnt......group by time,status`：该查询与 各状态码请求数量变化趋势 案例一致，统计各状态码请求数量变化趋势。
- `sum(cnt) over (partition by time)`：根据子查询的结果，把每个时间间隔下不同状态码的请求数量相加，作为该时间间隔下的请求总数。
- `cnt*1.0/sum(cnt) over (partition by time) as pct`：将每个状态码的请求数量除以请求总数，获得该状态码的请求占比。其中 `*1.0` 用于将 cnt 转换为 double 类型，以便后续进行除法运算时保留精度，否则整数相除结果仍为整数，将丢失精度。

P95 耗时最高的 URL

```
* | select approx_percentile(request_time,0.95) as P95,
    url
    group by url
    order by P95 desc
    limit 50
```

语句解释：

- `approx_percentile(request_time,0.95)`：对 request_time 进行统计，获取其位于95%分位的值，即95%的请求耗时低于该值。
- `group by url`：按照 url 分组，统计每个 url 对应 P95。
- `order by P95 desc`：按照 P95降序排序，优先展示 P95高的 url。
- `limit 50`：只返回查询结果中的前50条，即 P95最高的50个 url。

P95 耗时变化趋势

```
* | select approx_percentile(request_time,0.95) as P95,
    histogram(__TIMESTAMP__,INTERVAL ${__interval}) as time
    group by time limit 10000
```

语句解释：

- `approx_percentile(request_time,0.95)`：对 request_time 进行统计，获取其位于95%分位的值，即95%的请求耗时低于该值。
- `histogram(__TIMESTAMP__,INTERVAL ${__interval})`：按时间查看指标变化趋势时，需指定趋势图中每个数据点之间的间隔，例如查询最近60分钟的变化趋势，数据点之间的间隔为30秒。此处：

版权所有：腾讯云计算（北京）有限责任公司

- `__TIMESTAMP__` 代表日志时间，即请求的发生时间。
- `${__interval}` 代表数据点之间的间隔，是一个变量，将根据查询时间范围的长度动态计算最合适的间隔。也可手动指定，例如修改为 `1 MINUTE`，代表间隔为1分钟。

- `group by time`：使用 histogram 指定好时间间隔后，按该间隔分别统计每个间隔的 P95耗时。
- `limit 10000`：SQL 未指定 limit 时，默认仅返回100条查询结果，为获取完整的查询结果，将 limit 指定为更大的值。

各 URL P95 耗时变化趋势（仅查看耗时最高的10个 URL）

```
* | select approx_percentile(request_time,0.95) as P95,
    histogram(__TIMESTAMP__,INTERVAL ${__interval}) as time,
    url
    where url in  (select url group by url order by approx_percentile(request_time,0.95) desc limit 10)
    group by time,url
    order by P95 desc limit 10000
```

语句解释：

- `where url in  (select url group by url order by approx_percentile(request_time,0.95) desc limit 10)`：URL 很多是，直接查看所有 URL 各自的耗时变化趋势会使得趋势图非常凌乱，通过该过滤条件仅查看耗时最高的10个 URL 。
- 其它语法含义同上例。

网络总流量最高的 url

```
* | select sum(body_bytes_sent) as "网络流量",
    url
    group by url
    order by "网络流量" desc
    limit 50
```

语句解释：

- `sum(body_bytes_sent)`：对 body_bytes_sent 进行求和，获取网络总流量。
- `group by url`：按照 url 分组，统计每个 url 对应的网络总流量。
- `order by "网络流量" desc`：按照 P95降序排序，优先展示网络总流量高的 url。
- `limit 50`：只返回查询结果中的前50条，即网络总流量最高的50个 url。

按照版本号范围过滤日志

```
* | select *
where array_join(transform(split(version, '.'), x -> lpad(x, 10, '0')), '.') >
array_join(transform(split('2.0.0', '.'), x -> lpad(x, 10, '0')), '.')
```

语句解释：

版本号 version 为字符串，例如“2.0.1”，无法直接使用检索条件按范围对其进行过滤，因此需在 SQL 中进行过滤。上面的 SQL 可过滤版本号大于等于“2.0.0”的日志。

- `split(version, '.')`：使用符号 `.` 将版本号拆分为多个部分，结果为数组。例如将 `2.0.1` 转换为 `["2","0","1"]` 。
- `transform(split(version, '.'), x -> lpad(x, 10, '0'))`：使用 transform 函数在数组中每个数字左侧补0，使长度均为10，避免版本号中每部分长度不一致影响后续比较。例如将 `2.0.1` 转换为 `["0000000002","0000000000","0000000001"]` 。
- `array_join`：使用符号 `.` 将数组拼接为字符串，由于已经使用 transform 函数将版本号中每部分的长度均统一为了10，此时的字符串可直接按字符进行比较。

> ⚠ **注意：**
>
> - 版本号格式需一致，例如有些是 `2.0` 而有些是 `2.0.0` 。
> - 版本号不能包含非数字，例如 `2.0.0-alpha` 。
> - 版本号每部分长度不能超过10。

运营分析类案例

各 URL PV 及 UV

```
* | select count(*) as PV, approx_distinct(remote_addr) as UV, url
    group by url
    order by PV desc
    limit 50
```

语句解释：

- `approx_distinct(remote_addr)`：对客户端 IP remote_addr 去重统计数量作为 UV。
- `group by url`：按照 url 分组，统计每个 url 对应的 PV 及 UV。
- `order by PV desc`：按照 PV 降序排序，优先展示 PV 高的 url。
- `limit 50`：只返回查询结果中的前50条，即 PV 最高的50个 url。

UV 国内省份分布

```
* | select approx_distinct(remote_addr) as UV, ip_to_province(remote_addr) as province
    where ip_to_province(remote_addr) != 'NULL' and ip_to_country(remote_addr) = '中国'
    group by province
    order by UV desc
    limit 1000
```

语句解释：

- `approx_distinct(remote_addr)`：对客户端 IP remote_addr 去重统计数量作为 UV。
- `ip_to_province(remote_addr)`：根据客户端 IP remote_addr 获取客户端所在的省份。
- `where ip_to_province(remote_addr) != 'NULL' and ip_to_country(remote_addr) = '中国'`：过滤掉无省份信息的 IP（例如内网 IP）并只保留国内 IP。
- `group by province`：按照 province 分组，统计每个 province 对应的 UV。
- `order by UV desc`：按照 UV 降序排序，优先展示 PV 高的 province。
- `limit 50`：只返回查询结果中的前50条，即 UV 最高的50个 province。

UV 环比

```
* |
select compare[1] as today, compare[2] as yesterday, round(compare[3],4) as ratio
from (
    select compare(UV, 86400) as compare
    from (
        select approx_distinct(remote_addr) as UV
    )
)
```

语句解释：

- `approx_distinct(remote_addr)`：对客户端 IP remote_addr 去重统计数量作为 UV。
- `compare(UV, 100) as compare`：对子查询中的 UV 计算其环比值，86400 代表计算其与86400秒（即1天）前的比值，结果为数组，按顺序分别为当前 UV（`compare[1]`）、一天前的 UV（`compare[2]`）、当前 UV 与一天前的比值（`compare[3]`）。

PV 按移动端设备类型分布

```
* | select
    case when
           lower(http_user_agent) like '%iphone%' OR
           lower(http_user_agent) like '%ipod%' OR
           lower(http_user_agent) like '%ipad%' OR
           lower(http_user_agent) like '%ios%'
```

```
        then 'IOS'
        when
          lower(http_user_agent) like '%android%'
        then 'Android'
        else 'other'
    end as type ,
    count(*) as PV
    group by type
    HAVING type != 'other'
```

语句解释：

- `case when`：根据 http_user_agent 中包含的关键字将其分别归类至不同的设备类型（IOS、Android、other）。
- `lower(http_user_agent)`：将 http_user_agent 转换为小写，方便匹配关键字。
- `count(*) as PV`：统计所有日志条数，即 PV。
- `group by type`：按照设备类型分组，统计每个设备类型对应的 PV。
- `HAVING type != 'other'`：针对统计的结果进行二次过滤，去除 other 类型的设备。

Top 访问量 IP

```
* | select count(*) as PV, remote_addr as IP
    group by IP
    order by PV desc
    limit 50
```

语句解释：

- `count(*) as PV`：统计所有日志条数，即 PV。
- `group by IP`：按 IP 分组，统计每个 IP 对应的 PV。
- `order by PV desc`：按照 PV 降序排序，优先展示 PV 高的 IP。
- `limit 50`：只返回查询结果中的前50条，即 PV 最高的50个 IP。

采样分析

最近更新时间：2024-08-15 15:04:21

操作场景

使用 统计分析 功能时，若原始日志条数非常多或统计分析语句（SQL）较为复杂，统计分析可能出现缓慢甚至超时的情况。此时可使用采样分析功能对原始日志进行采样，再进行统计分析。
日志服务通过随机采样的方式对原始日志进行采样，根据统计学相关原理，在合理的采样率及较大的样本数下，采样后的统计结果相比全量精确统计误差较低，能够满足大多数海量数据的统计分析需求。

采样方式

统计分析默认为“精确分析”，即不采样。需要采样时，可选择如下两类采样方式：

- 自动采样：根据原始日志条数多少自动决定采样率，使得样本数约为100万，以该样本进行统计分析。例如原始日志为100亿条时，使用自动采样获取100万样本，对应的采样率为1:10,000。
- 固定采样率采样：人工设定采样率，根据该采样率从原始日志中获取样本。例如原始日志为100亿条时，按照1:100,000采样，样本数为10万。采样率最大为1（即不采样），最小为1:1000,000。

无论采用哪种采样方式，在统计分析结果中均可查看最终实际的采样率。

估算真实值

采样分析结果能够很大程度上反应真实值，根据值的计算方式，可参见如下方法估算实际值：

- 平均值相关的计算方式：例如 `avg` 、`geometric_mean` 等，直接以样本统计结果代表真实值。
- 和值相关的计算方式：例如 `count(*)` 、`sum` 、`count_if` 等，以样本统计结果除以采样率代表真实值。例如，按照1:10,000的采样率统计接口 pv(`count(*)`)，样本统计结果为232，则真实值约为232 / (1:10000) = 2320000。

例如按照1:10,000采样时，检索分析语句为：

```
* | select count(*)/(1.0/10000) as pv,avg(response_time) as response_time_avg
```

- pv 计算： `count(*)` 为统计日志条数，属于和值计算，需以样本统计结果除以采样率代表真实值。其中， `(1.0/10000)` 即为采样率，计算获得的 pv 为真实 pv 的估算值。
- response_time_avg 计算： `avg(response_time)` 为计算响应时间 `response_time` 的平均值，属于平均值计算，可直接作为真实 response_time_avg 的估算值。

采样分析准确度

统计学上使用置信水平和置信区间衡量采样结果的准确度，置信水平代表采样结果的可信度，置信区间代表指定置信水平下真实值所在的区间。例如某一样本统计结果置信水平为95%，置信区间为[212,478]，则真实值有95%的概率位于212 – 478之间。
置信水平越高，置信区间越大。常用置信水平为95%（统计学中一般认为概率低于5%为小概率事件），对应的置信区间计算方式为：

$$\left[\bar{x} - 1.961\frac{s}{\sqrt{n}}, \bar{x} + 1.961\frac{s}{\sqrt{n}}\right]$$

其中，$\bar{x}$ 为样本平均值，s 为样本标准差，n 为样本数。
例如使用自动采样进行统计分析：

```
* | select avg(response_time) as x,count(response_time) as n,stddev_samp(response_time) as s
```

样本统计分析结果为：

- x：即样本平均值，544.4656932942215；
- n：即样本数，995097；
- s：即样本标准差，1382.618439585749。

则置信水平为95%时， `avg(response_time)` 的真实值位于[541.75,547.18]内，而实际上该案例通过精确统计得到的 `avg(response_time)` 为545.16，估算值非常接近真实值。

> **说明：**
> 以上为严格的采样准确度衡量方式，实际使用过程中，样本数大于1000时，采样获得的结果一般具备较高的准确度。

按维度统计分析时（即group by），由于样本会按指定维度拆分为多个组，并在每个组内分别计算统计值，此时单个组内的样本数将低于总样本数，会导致样本数较少的组的统计值准确度较低。
例如使用采样进行统计分析：

```
* | select avg(response_time) as response_time,count(*) as sampleCount,url group by url order by count(*) desc
```

样本统计分析结果为：

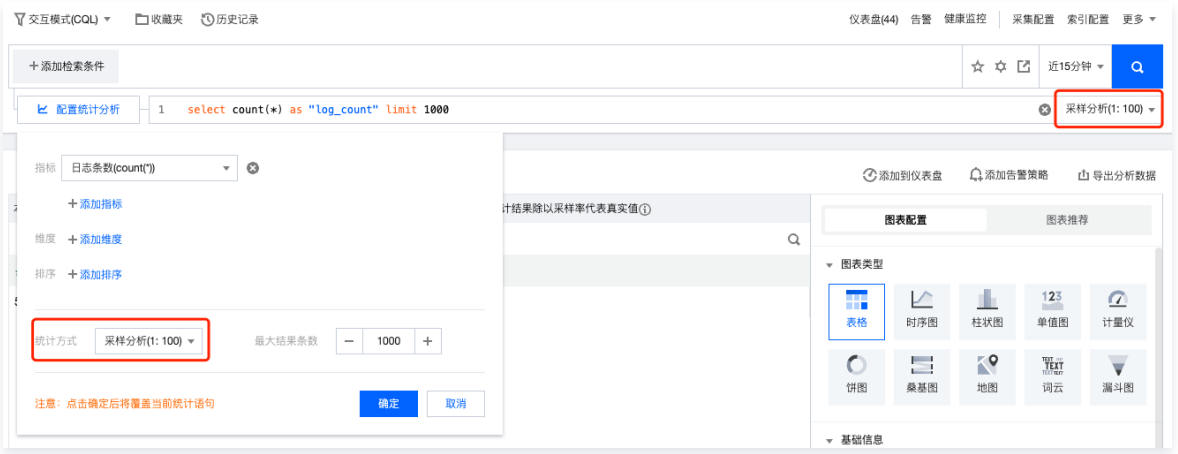
url	response_time	sampleCount
/user	45.23	7845
/user/list	78.45	6574
/user/login	45.85	5235
/user/logout	45.48	1245
/book/new	125.78	987
/book/list	17.23	658
/book/col	10.21	23
/order	12.13	2

其中，`/book/col` 和 `/order` 两个 url 的样本数过低，统计结果准确度较低，不具备参考性。如需对这两个 url 获取更加准确的统计结果，可提高整体的采样率，或单独对其进行统计分析。例如：

```
url:"/book/col" OR url:"/order" | select avg(response_time) as response_time,count(*) as sampleCount,url group by url order by count(*) desc
```

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**检索分析**，进入日志主题检索分析页面。
3. 在页面顶部，选择需要检索分析的日志主题。
4. 输入检索分析语句（需包含 SQL）后，在右侧**检索分析**按钮左侧设定是否需要采样分析以及对应的采样率。
   - 交互模式：



   - 语句模式：



5. 单击右侧 执行检索分析。完成检索分析后，图表顶部将显示本次分析采用的采样率。

版权所有：腾讯云计算（北京）有限责任公司

关联外部数据

最近更新时间：2025-01-08 14:47:12

操作场景

对日志进行统计分析时，有时需要将日志与外部数据关联起来才能进行完整的业务统计，例如以下使用场景：

- 日志中仅保存了用户 ID，统计分析时需根据用户 ID 在用户信息数据库中查询用户的等级、地域和类型等信息，例如按用户性别分别统计用户访问次数。
- 日志中仅保存了业务系统 ID，根据 ID 很难快速对应到具体的业务名称，需要根据 ID 与名称的映射关系将业务系统 ID 翻译为业务名称。
- 在仪表盘中除了查看日志数据，还需要查看其它存储在数据库中的数据。

日志服务支持将 MySQL 及对象存储中的 CSV 文件作为数据库表关联至日志主题，然后使用 SQL 语句将两部分数据进行关联分析。

前提条件

已开通日志服务并 创建日志主题，日志主题需为标准存储，低频存储不支持使用 SQL 进行统计分析。

注意事项

- **关联 MySQL 时**，请注意网络/数据安全及 MySQL 数据库性能：
  - 网络安全：建议使用内网地址访问 MySQL，数据源类型为**云数据库 MySQL** 和**云数据库 TDSQL-C MySQL** 时，CLS 将自动采用内网方式访问。
  - 数据安全：建议创建专门的数据库账号供 CLS 访问 MySQL，尽可能的缩小账号对应的资源及操作权限，CLS 仅需查询权限，无需编辑及删除权限，配置方式请参见 云数据库 MySQL 修改账号权限 。请妥善保管账号信息，不要泄露。
  - 性能：如果 MySQL 内数据量较大，建议不要直接将生产环境主数据库关联至 CLS，避免通过 CLS 对数据库发起过于复杂的 SQL 查询，影响生产环境稳定性，建议使用 只读实例 。
- **关联对象存储 CSV 文件**时，建议使用与日志主题相同地域的存储桶，跨地域访问存储桶内的文件将会在 COS 侧产生外网下行流量及相应费用。对象存储访问权限建议使用“私有读写”，避免数据被第三方访问。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，进入日志主题列表页面。
3. 单击需要关联外部数据的日志主题 ID/名称，进入日志主题管理页面。
4. 选择**关联外部数据**页签，单击**添加外部数据**，在弹出对话框中配置相关参数。

云数据库 MySQL / TDSQL-C MySQL

使用云数据库 MySQL / TDSQL-C MySQL 时，采用该配置方式：

参数名称	描述
名称	作为 CLS SQL 中的表名，支持小写字母、数字和_，且不能以_开头和结尾，长度为3 - 60字符，在所属地域内名称不能重复。
备注	可选，不超过255个字符
数据源类型	云数据库 MySQL 或 云数据库 TDSQL-C MySQL
地域	选择云数据库所在的地域
MySQL 实例	选择云数据库实例
账户名	访问 MySQL 使用的账户名
密码	访问 MySQL 使用的密码
数据库名	需要关联的 MySQL 数据库名称
表名	需要关联的 MySQL 数据库中的表名称
使用范围	• 仅当前日志主题可访问：仅当前日志主题可通过 SQL 访问该 MySQL 的数据 • 当前日志集内日志主题可访问：当前日志主题所属日志集内的日志主题都可以通过 SQL 访问该 MySQL 的数据

版权所有：腾讯云计算（北京）有限责任公司

配置完成后，单击**验证访问配置**，验证上述配置是否正确。验证通过后单击**确定**完成添加。

对象存储 COS（CSV 格式文件）

添加对象存储 COS（CSV 格式文件）时配置如下参数：

参数名称	描述
名称	作为 CLS SQL中的表名，支持小写字母、数字和_，且不能以_开头和结尾，长度为3至60字符，在所属地域内名称不能重复。
备注	可选，不超过255个字符
数据源类型	对象存储 COS（CSV 格式文件）
存储桶地域	选择 COS 文件所属地域
COS 存储桶	选择 COS 文件所属存储桶
文件名称	填写 COS 文件名称
压缩模式	暂时仅支持**不压缩**

可单击**预览数据**，验证上述配置是否正确，同时从 CSV 文件中获取前5行数据（含表头）作为样例，进一步配置如下参数：

参数名称	描述
字段类型	支持 text、long 及 double，请根据实际数据类型选择
访问范围	• 仅当前日志主题可访问：仅当前日志主题可通过 SQL 访问该 MySQL 的数据 • 当前日志集内日志主题可访问：当前日志主题所属日志集内的日志主题都可以通过 SQL 访问该 MySQL 的数据

单击**确定**完成添加。

自建 MySQL

MySQL 直接部署在 CVM 中或其它平台上时，采用该配置方式：

参数名称	描述
名称	作为 CLS SQL 中的表名，支持小写字母、数字和_，且不能以_开头和结尾，长度为3 - 60字符，在所属地域内名称不能重复。
备注	可选，不超过255个字符
数据源类型	自建 MySQL
访问方式	内网地址或公网地址
地域	使用内网地址时需填写，选择 MySQL 所在的地域
所属网络	使用内网地址时需填写，选择 MySQL 所在的 VPC
网络服务类型	使用内网地址时需填写： • 若您的 MySQL 需通过 CLB 访问，请选择 CLB • 若您的 MySQL 所在服务器可直接访问，请选择 CVM
访问地址	例如 gz-cdb-xxxxx.sql.tencentcdb.com
MySQL 端口	数据库端口，例如3306
账户名	访问 MySQL 使用的账户名
密码	访问 MySQL 使用的密码

数据库名	需要关联的 MySQL 数据库名称
表名	需要关联的 MySQL 数据库中的表名称
使用范围	• 仅当前日志主题可访问：仅当前日志主题可通过 SQL 访问该 MySQL 的数据 • 当前日志集内日志主题可访问：当前日志主题所属日志集内的日志主题都可以通过 SQL 访问该 MySQL 的数据

单击验证访问配置，验证上述配置是否正确。验证通过后单击确定完成添加。

使用 SQL 关联查询外部数据及日志数据

从 SQL 角度来看，日志主题内的日志数据及外部数据均为数据库表，日志数据对应的表名为 log ，外部数据对应的表名为添加外部数据时设置的名称。因此可分别使用如下 SQL 单独查询日志数据及外部数据：

- **日志数据：`* | select * from log`，其中`from log`可以省略，即`* | select *`，同时查询日志数据及外部数据时，建议您不要省略，以提高 SQL 可读性。**
- **外部数据：假设外部数据名称为`userinfo`，则对应 SQL 为`* | select * from userinfo`。**

同时查询日志数据及外部数据需使用 JOIN、UNION 等语法，例如：

```
* | select * from log left join userinfo on log.user_id=userinfo.id
```

> **注意：**
> **查询外部数据时，`|` 前的检索条件及本次查询指定的时间范围对外部数据不生效，仅对当前日志主题的日志数据生效。**

使用案例

某日志主题为请求日志，包含如下字段：

```
"status_code": "404"
"local_time": "2023-06-05 19:59:01"
"refer": "_",
"user_id": "15"
"ip": "66.131.53.125"
"url": "\"GET /class/111.html HTTP/1.1\""
```

统计分析该日志时，期望能够按照访问用户性别分别统计访问次数，但访问日志中并没有用户性别字段，无法直接统计，需要在用户信息数据库（MySQL）中根据用户 ID（user_id）查询。
用户信息数据库表如下：

id	Name	Gender	Age	Email	Phone	Address
1	John Doe	Male	32	johndoe@example.com	1234567890	123 Main St
2	Jane Smith	Female	28	janesmith@example.com	9876543210	456 Elm St
3	Michael Johnson	Male	45	michaeljohnson@example.com	5551234567	789 Oak St
4	Sarah Davis	Female	38	sarahdavis@example.com	7894561230	321 Pine St
5	David Wilson	Male	51	davidwilson@example.com	1237894560	654 Maple St
6	Emily Anderson	Female	29	emilyanderson@example.com	4567890123	987 Cherry St
7	Matthew Thompson	Male	37	matthewthompson@example.com	7890123456	321 Plum St
8	Olivia Martinez	Female	26	oliviamartinez@example.com	2345678901	654 Orange St
9	Alexander Taylor	Male	42	alexandertaylor@example.com	9012345678	987 Grape St

版权所有：腾讯云计算（北京）有限责任公司

10	Emma Clark	Female	31	emmaclark@example.com	3456789012	123 Lemon St

将该数据库表关联至日志主题后，即可使用如下语句将日志中的 `user_id` 转换为用户性别 `Gender`：

```
* | select ip,url,user_id,Name,Gender from log left join userinfo on log.user_id=userinfo.id
```

查询结果如下图，其中 ip、url、user_id 来源于日志，Name、Gender 则来源于用户信息库表。

t ip	t url	# user_id	t Name	t Gender
86.125.132.187	"GET /class/115.html ...	8	Olivia Martinez	Female
10.163.123.86	"GET /class/114.html ...	8	Olivia Martinez	Female
163.132.168.168	"GET /anyway/666 H...	19	William Garcia	Male
131.163.86.125	"GET /class/111.html ...	8	Olivia Martinez	Female
168.163.168.125	"GET /class/116.html ...	8	Olivia Martinez	Female
25.66.125.10	"GET /class/114.html ...	25	Elizabeth Moore	Female
86.37.10.123	"GET /class/111.html ...	8	Olivia Martinez	Female
86.10.66.37	"GET /class/111.html ...	8	Olivia Martinez	Female
132.10.125.19	"GET /class/112.html ...	7	Matthew Thompson	Male
132.25.187.66	"GET /class/116.html ...	6	Emily Anderson	Female

在此基础上，进一步编写 SQL，即可按照访问用户性别分别统计访问次数：

```
* | select count(*) as pv,Gender from (select ip,url,user_id,Name,Gender from log left join userinfo on
log.user_id=userinfo.id) group by Gender
```

查询结果如下图，其中 Gender 列中的 null 表示部分日志没有查询到用户性别，可能是日志中无 user_id，也可能为 user_id 在用户信息库中无记录。

# pv	t Gender
73	Female
25	null
52	Male

规格限制

关联外部数据库的规格与限制，请参见 规格与限制 。

费用说明

- 关联对象存储 CSV 文件时，如果对象存储桶与日志主题不在同一个地域，每次查询该数据都会在 COS 侧产生外网下行流量及相应费用，CLS 侧无费用。
- 关联 MySQL 时，如果使用外网连接，每次查询该数据都会在对应的资源侧（例如云数据库 MySQL 或 CVM）产生外网访问流量，可能产生相应费用（取决于您购买的实例配置），CLS 侧无费用。

版权所有：腾讯云计算（北京）有限责任公司

跨主题关联分析

最近更新时间：2025-07-01 16:21:52

操作场景

日志数据分散在多个主题中时，可使用 SQL 跨主题进行统计分析，例如以下场景：

- 微服务架构下，每个服务的日志存储在不同主题中，需按照请求 ID 汇总查询各个服务的耗时。
- 日志分为报错日志和业务访问日志，存储在不同主题中，需根据报错日志中的用户 ID、订单 ID 等信息在访问日志中查询关联的访问记录。

前提条件

- 日志接入标准存储，低频存储不支持使用 SQL 进行统计分析。
- 最大支持同时关联分析5个日志主题。

> **注意：**
> 该功能目前以白名单方式供部分用户使用。如需开通白名单，请 联系我们 。

操作步骤

跨主题关联分析通过 SQL 语句实现，在日志检索分析、仪表盘、定时 SQL、告警策略中均可使用。以日志检索分析为例，操作步骤参考 检索分析语法规则操作步骤 。

语法规则

- 在 SQL 中，每个日志主题相当于一个数据库表，表名为主题 ID，可使用 JOIN 、UNION 、INTERSECT 、EXCEPT 等语法进行关联分析。
- 主题 ID 中包含特殊字符 `-` ，在 SQL 中需使用双引号包裹，例如 `"0c037c7c-dbf0-4646-bb8b-9e414e5928f7"` 。
- 查询当前主题中的数据时，可使用 `log` 代表当前主题，也可省略。例如 `* | select * from log` 、`* | select *` 。

示例

场景描述

某业务系统包含两个服务，订单管理服务的日志存储在主题 A 中，支付服务的日志存储在主题 B 中，需根据请求 ID 查询各个服务的耗时。
日志主题 A 的 ID 为 `"0c037c7c-dbf0-4646-bb8b-9e414e5928f7"` ，日志数据如下。

```
{
    "RequestId": "4f7g2k9m",
    "timeCost": 856,
    "timestamp": "2024-01-31T19:44:08+08:00",
},
{
    "RequestId": "6m4v1ztf",
    "timeCost": 1452,
    "timestamp": "2024-01-31T19:45:08+08:00",
}
```

日志主题 B 的 ID 为 `"8c05119c-c3e8-4e8f-95b7-81eab4702b0a"` ，日志数据如下。

```
{
    "reqid": "4f7g2k9m",
    "costms": 247,
    "timestamp": "2024-01-31T19:44:08+08:00",
},
{
    "reqid": "6m4v1ztf",
    "costms": 365,
    "timestamp": "2024-01-31T19:45:08+08:00",
```

```
}
```

查询语句

在订单管理服务日志主题 A 中执行该查询语句。

```
* |
SELECT
  order_service.RequestId,
  order_service.timeCost AS "订单管理服务耗时",
  pay_service.costms AS "支付服务耗时"
FROM
  log AS order_service
  JOIN "8c05119c-c3e8-4e8f-95b7-81eab4702b0a" AS pay_service
  ON order_service.RequestId = pay_service.reqid
LIMIT 100
```

查询结果

RequestId	订单管理服务耗时	支付服务耗时
4f7g2k9m	856	247
6m4v1ztf	1452	365

索引配置

最近更新时间：2025-06-25 19:05:52

操作场景

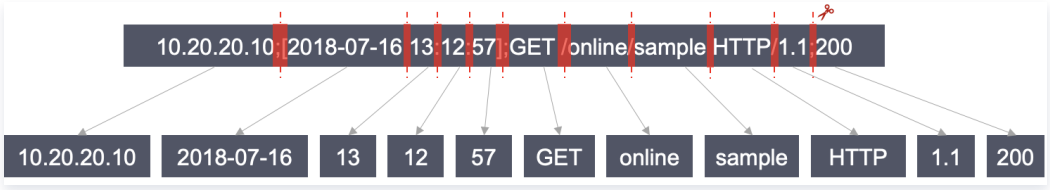
索引配置是使用日志服务（Cloud Log Service，CLS）进行检索分析的必要条件，只有开启索引才能对日志进行检索分析。而且不同的索引规则也会产生不同的日志检索分析效果，本文档会详细介绍索引的配置规则及原理。

功能介绍

索引配置的核心是对原始日志进行“分词”，以快速且便捷地根据特定的检索条件检索日志。同时在索引配置中还可以针对特定的字段“开启统计”，便于使用 SQL 对日志进行统计分析。
检索一个较长的日志时，通常只使用其中一部分内容进行检索。例如：需要检索出包含 `sample` 的如下完整日志：

```
10.20.20.10;[2018-07-16 13:12:57];GET /online/sample HTTP/1.1;200
```

由于日志全文很长，除 `sample` 外还有很多其他内容，其与检索条件并不是直接相等，因此不能直接使用 `sample` 作为检索条件来检索该日志。为满足该检索需求，需要将日志全文切分为多个片段，每个片段称之为一个“词”，而这个过程称之为“分词”。
例如：按符号对上述示例日志进行切分，只要出现了 `@&()='",;:<>[]{}/ \n\t\r\\` 范围内的符号，就切分日志，将得到如下的词：



检索条件为 `sample` 时，上述分词后的日志包含 `sample`，即认为符合检索条件。
检索条件本身也会进行分词，例如下图2种检索条件：



- 检索条件A：`"/online/sample"`
  - 双引号表示日志需**同时存在**这两个词，且两个词**顺序严格一致**才符合检索条件。
  - 上述示例日志同时包含 `online` 和 `sample`，且词的顺序与检索条件一致，符合该检索条件。
- 检索条件B：`"/online/login"`

  上述示例日志不包含 `login`，**不符合**该检索条件。

日志服务包含以下两类索引：

类别	描述
全文索引	全文索引将日志全文整体切分为多个分词进行索引构建，检索时直接通过关键词进行检索（即全文检索），例如使用 `error` 检索所有出现过 error 的日志。
键值索引	键值索引将原始日志按字段（即 key:value）分别切分为多个分词进行索引构建，检索时基于键值方式进行检索（即键值检索），例如使用 `level:error AND timeCost:>1000` 检索日志级别(level)为 error 且耗时(timeCost)大于 1000ms 的日志。 某些日志中还包含一类特殊的“元数据字段”，这类字段配置索引的方式与普通字段一致。

注意：

- 索引配置较为复杂，建议您使用 操作步骤 中的推荐配置，能够满足大多数使用需求。
- 索引关闭时采集的日志数据无法被检索，从开启索引到支持日志检索分析存在约一分钟延迟。
- 索引规则编辑（包括新增/编辑/删除字段、调整分词符配置等在内的所有操作）后仅对新写入的日志生效，已有数据不会更新。

版权所有：腾讯云计算（北京）有限责任公司

- 下划线开头为系统保留用法，若您的日志中包含下划线开头的字段，开启索引后，将会在字段前加上@符号。
- 低频存储 仅支持全文索引，不支持键值索引。

全文索引

配置项	功能描述
全文分词符	对原始日志进行分词的字符集合，仅支持英文符号，控制台默认分词符为 `@&?|#()='",;:<>[]{}/ \n\t\r\\` 。
大小写敏感	检索时是否对大小写敏感。例如日志为 Error，若大小写敏感，则使用 error 无法检索到该条日志。
是否包含中文	日志中包含中文且需要对中文进行检索时可开启该功能。例如日志为“用户登录接口超时”，若未开启该功能，搜索 `"超时"` 无法检索到该日志，只有完整地搜索 `"用户登录接口超时"` 才能检索到该日志，开启该功能后便可通过搜索“超时”检索到该日志。

键值索引

配置项		功能描述
整体配置	大小写敏感	检索时是否对大小写敏感。例如日志为“level:Error”，若大小写敏感，则使用 `level:error` 无法检索到该条日志。
	自动配置	是否启用键值索引自动配置，启用后将自动将日志内的字段添加到键值索引中，包括日志中后续新增的字段。字段添加规则请参见 自动配置 。
字段配置	字段名称	结构化日志 中的字段名称，单个日志主题键值索引最多可添加300个字段。 • 支持除 `*\",` 外的所有英文字母、符号及数字，不能以 `_` 开头(`__CONTENT__` 字段例外) • 不允许同时包含 JSON 父子级字段，例如 `a` 及 `a.b` 。
	字段别名	使用别名代替原始字段名称进行检索分析，例如为 `__SOURCE__` 字段添加别名 `ip` 。 • 别名仅支持字母、数字和下划线，且需以字母开头。 • 仅支持 CQL 语法，不支持 Lucene 语法。
	字段类型	字段的数据类型，包括 text、long、double 三种类型。 • text 类型支持使用通配符进行模糊检索，不支持范围比较。 • long、double 类型支持范围比较，不支持模糊检索。
	分词符	对字段值进行分词的字符集合，仅支持英文符号，控制台默认分词符为 `@&?|#()='",;:<>[]{}/ \n\t\r\\` 。
	包含中文	字段中包含中文且需要对中文进行检索时可开启该功能。例如日志为“message:用户登录接口超时”，若未开启该功能，使用 `message:"超时"` 无法检索到该日志，只有使用 `message:"用户登录接口超时"` 才能检索到该日志，开启该功能后便可使用 `message:"超时"` 检索该日志。
	开启统计	开启后可对该字段使用 SQL 进行统计分析，text 类型字段开启统计时，如果值过长，仅前32766个字符参与统计运算。 开启统计不会额外产生费用，建议您开启。

说明：
已开启全文索引时，键值索引不产生任何额外索引流量/存储费用，建议您尽量多地为字段开启键值索引，以获得更好的检索分析体验。

为满足最基本的日志检索要求，日志服务针对 日志 中部分内置保留字段自动创建了键值索引，但不会产生索引流量，不会增加费用，具体如下：

内置保留字段	说明	键值索引配置
`__FILENAME__`	日志采集的文件名，可以利用该字段检索特定文件下的日志，例如 `__FILENAME__:"/var/log/access.log"` 检索 `/var/log/access.log` 文件的日志。	text 类型、无分词符、不包含中文、已开启统计
`__SOURCE__`	日志采集的源 IP，可以利用该字段检索特定机器下的日志，例如 `__SOURCE__:192.168.10.10` 检索 `192.168.10.10` 机器的日志。	text 类型、无分词符、不包含中文、已开启统计

__HOSTNAME__	日志来源机器名称，可以利用该字段检索特定机器下的日志，例如 __HOSTNAME__:localhost 检索 localhost 机器的日志。 注意：需使用2.7.4及以上版本的 Loglistener 才会采集该字段。	text 类型、无分词符、不包含中文、已开启统计
__TIMESTAMP__	日志时间戳（毫秒级别 Unix 时间戳），按时间范围检索日志时，将自动使用该时间对日志进行检索，无需在检索语句中指定。在控制台显示为“日志时间”。	long 类型，已开启统计
__INDEX_STATUS__	值非空时代表当前日志创建索引 出现异常，原始日志被存储在 __RAWLOG__ 中，__INDEX_STATUS__ 为创建索引异常原因。 使用 __INDEX_STATUS__:* 可以查询所有索引创建异常的日志。	text 类型、无分词符、不包含中文、已开启统计

自动配置

开启**自动配置**开关后，将自动把日志内的字段添加到键值索引字段列表中，包括日志中新增的字段。
添加字段时，将根据字段值自动判断字段类型，将字段类型设置为 text、long 或 double，并使用如下配置。如果以下配置不满足使用需求（例如字段类型不正确），可在添加后的字段上手动修改。

字段类型	分词符	包含中文	开启统计
text	@&?|#()='",;:<>[]{}/ \n\t\r\\	包含	开启
long	不涉及	不涉及	开启
double	不涉及	不涉及	开启

说明：
日志字段名称仅支持字母、数字、下划线和-./@，且不能以下划线开头，不符合该规则的字段不会添加到键值索引字段列表中。

日志字段为 json 字符串时，默认不会解析 json。如需将 json 中的子级字段添加至键值索引，请手动添加其中的任意子级字段，后续将自动把该子级字段的同级别字段添加到键值索引字段列表中。
例如原始日志共三个字段：

```
key1:textValue
key2:123
key3:{"ip":"123.123.123.132","url":"class/132.html","detail":{"status_code":"500","id":13}}
```

自动添加的键值索引字段列表为：

字段名称	字段类型	分词符	包含中文	开启统计
key1	text	@&?|#()='",;:<>[]{}/ \n\t\r\\	包含	开启
key2	long	不涉及	不涉及	开启
key3	text	@&?|#()='",;:<>[]{}/ \n\t\r\\	包含	开启

如果手动将字段 key3 修改为 key3.ip，将自动把 ip 同级的字段 url 及 detail 添加到键值索引字段列表中，最终的列表为：

字段名称	字段类型	分词符	包含中文	开启统计
key1	text	@&?|#()='",;:<>[]{}/ \n\t\r\\	包含	开启
key2	long	不涉及	不涉及	开启
key3.ip	text	@&?|#()='",;:<>[]{}/ \n\t\r\\	包含	开启
key3.url	text	@&?|#()='",;:<>[]{}/ \n\t\r\\	包含	开启
key3.detail	text	@&?|#()='",;:<>[]{}/ \n\t\r\\	包含	开启

高级设置

版权所有：腾讯云计算（北京）有限责任公司

为满足部分特殊使用场景，索引配置提供如下高级设置。实际使用过程中，建议您采用推荐配置。通过控制台新建索引配置时，也会默认采用推荐配置。

配置项	含义	推荐配置
内置保留字段包含至全文索引	**包含**：全文索引包含 `__FILENAME__` 、 `__HOSTNAME__` 及 `__SOURCE__` 这三个内置字段，同时支持全文检索及键值检索，例如 `"/var/log/access.log"` 、 `__FILENAME__:"/var/log/access.log"` 。 **不包含**：全文索引不包含上述三个内置字段，只能使用键值检索，例如 `__FILENAME__:"/var/log/access.log"` 。	包含
元数据字段包含至全文索引	**包含**：全文索引包含所有元数据字段（前缀为 `__TAG__` 的字段），可直接使用全文检索的方式检索相关日志字段，例如 `ap-beijing` 。 **不包含**：全文索引不包含任何元数据字段，只能使用键值检索的方式检索相关日志字段，例如 `__TAG__.region:ap-beijing` 。低频日志主题不支持键值检索，此时将无法检索这些字段。 **仅包含开启键值索引元数据字段**：全文索引包含开启了键值索引的元数据字段，不包含未开启键值索引的元数据字段，低频日志主题无该选项。	包含
日志创建索引异常存储规则	日志创建索引过程中如果出现异常，CLS 为了确保日志不丢失，会将原始日志存储在 `__RAWLOG__` 中。如果此时仅部分字段创建索引失败，可将失败的部分存储在指定字段中（默认为 RAWLOG_FAIL_PART），请参见 创建索引出现异常是什么意思 。	启用

操作步骤

编辑索引配置

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，进入日志主题列表页。
3. 单击需要配置索引的日志主题 ID/名称，进入日志主题管理页。
4. 选择**索引配置**然后单击**编辑**，进入编辑索引配置页面，配置项说明请参见 全文索引 及 键值索引 。

> **说明：**
> - 首次开启索引配置时，推荐**使用推荐配置**，完成索引的自动配置。
>
> 
>
> - 低频主题仅支持全文索引。

导入索引配置

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，进入日志主题列表页。
3. 单击需要配置索引的日志主题 ID/名称，进入日志主题管理页。
4. 选择**索引配置**页签，单击**导入配置规则**。
5. 在对话框中，选择需要导入的日志主题索引配置，单击**确定**，将选中的日志主题的索引配置填写至当前日志主题的索引配置中。
6. 确认无误后，单击**确定**，即可保存当前日志主题的索引配置。

费用说明

开启索引后会产生索引流量及索引存储费用，请参见 计费概述 。如需降低相关费用，请参见 节省产品使用成本 。

规格与限制

索引的规格与限制，请参见 规格与限制 。

常见问题

创建索引出现异常是什么意思？__RAWLOG__及__INDEX_STATUS__是什么？

日志创建索引过程中，如果原始日志格式异常或索引配置与原始日志不匹配，可能会导致日志创建索引异常，此时 CLS 为了确保日志不丢失，会将原始日志存储在 `__RAWLOG__` 中，作为兜底的异常处理方式。 `__INDEX_STATUS__` 则为创建索引异常的原因。

- `__RAWLOG__` 仅支持全文检索（需开启全文索引），不支持键值检索、键值索引和统计分析。
- `__INDEX_STATUS__` 支持全文检索及键值检索，使用 `__INDEX_STATUS__:*` 可以查询所有索引创建异常的日志。

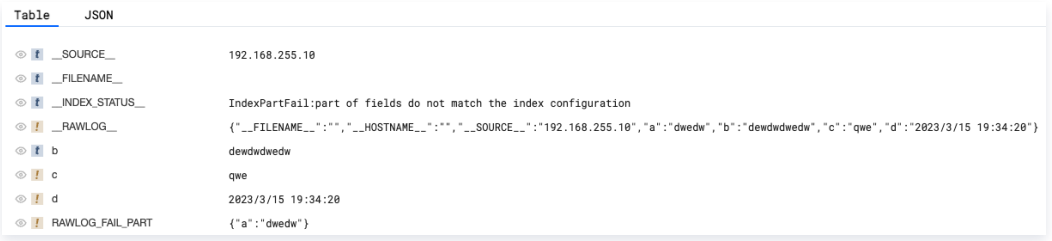
> **说明：**
> 产生异常的日志，开启全文索引后，仍按照原始日志全文计算索引流量、索引存储及对应费用，不产生额外费用。

创建索引异常包含两类情况：

1. 日志中所有的字段均未按照索引配置成功创建索引，此时日志中仅具备 `__RAWLOG__` 字段，只能使用全文检索。



2. 日志中部分字段未成功创建索引，部分字段成功创建索引，此时日志中既具备 `__RAWLOG__` 字段，又具备创建索引成功的部分字段。



在**索引配置-高级设置**中还可以将异常部分的字段存储在指定字段中（默认为 RAWLOG_FAIL_PART，支持配置键值索引及统计分析）。



CLS 会不断优化索引配置对原始日志的兼容性，尽量避免日志出现上述异常情况。随着产品迭代升级，具体的异常处理规则可能会发生变化。

重建索引

最近更新时间：2024-08-20 16:27:22

操作场景

由于索引规则编辑后仅对新写入的日志生效，如果需要按照最新的索引规则对历史数据重新创建索引，可使用重建索引功能，常见的需要使用重建索引的场景如下：

- 键值索引新增字段，历史数据也需要使用该字段进行检索。
- 调整分词符配置，历史数据也需要按照新的分词符进行检索。
- 字段之前未开启统计，在索引配置中开启后，历史数据也需要对该字段进行统计（即 SQL）。

重建索引会将指定时间范围内的原始日志重新构建一遍索引，将产生索引流量费用（不产生写流量费用及索引存储费用）。数据量较大时会消耗较长时间并产生较高的费用，建议您尽量避免频繁地修改索引配置并重建索引。

前提条件

已开启索引，且最新的索引配置能满足后续的检索分析需求。

注意事项

- 单个日志主题同时仅允许运行一个重建索引任务，单个日志主题最多同时拥有10个重建索引任务记录，需删除不再需要的任务记录后才能新建索引任务。
- 同一时间范围内的日志，仅允许重建一次索引，需删除之前的任务记录后才能再次重建。
- 删除重建索引任务记录将恢复重建索引前的索引数据。
- 所选时间范围对应日志写流量不能超出5TB。
- 重建索引时间范围以日志时间为准，日志上传时间与重建索引时间范围有超过1小时的偏差时（例如16:00上传了一条02:00的日志到 CLS，重建00:00~12:00的日志索引）不会被重建且后续无法进行检索。新上报一条日志到已经被重建的日志时间范围时，也不会被重建且后续无法进行检索。



操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**，进入日志主题列表页面。
3. 单击需要重建索引的日志主题 ID/名称，进入日志主题管理页面。
4. 选择**索引配置**页签， 单击底部的**重建索引**。



5. 选择需要重建的日志时间范围，单击**开始重建**。

> **说明：**
> 此处将根据选择的时间范围预估需要重建的日志数据对应的写流量及耗时，如果写流量过大、耗时过长，建议您尽可能地缩小需要重建的日志时间范围，以缩短耗时，同时也有助于降低重建索引带来的成本。开始重建后，可以查看任务的运行进度。

6. 重建完成后，可在列表中查看已经完成的任务。在列表中可删除指定任务，删除任务将恢复重建索引前的索引数据。

多主题检索

最近更新时间：2024-10-18 10:20:11

操作场景

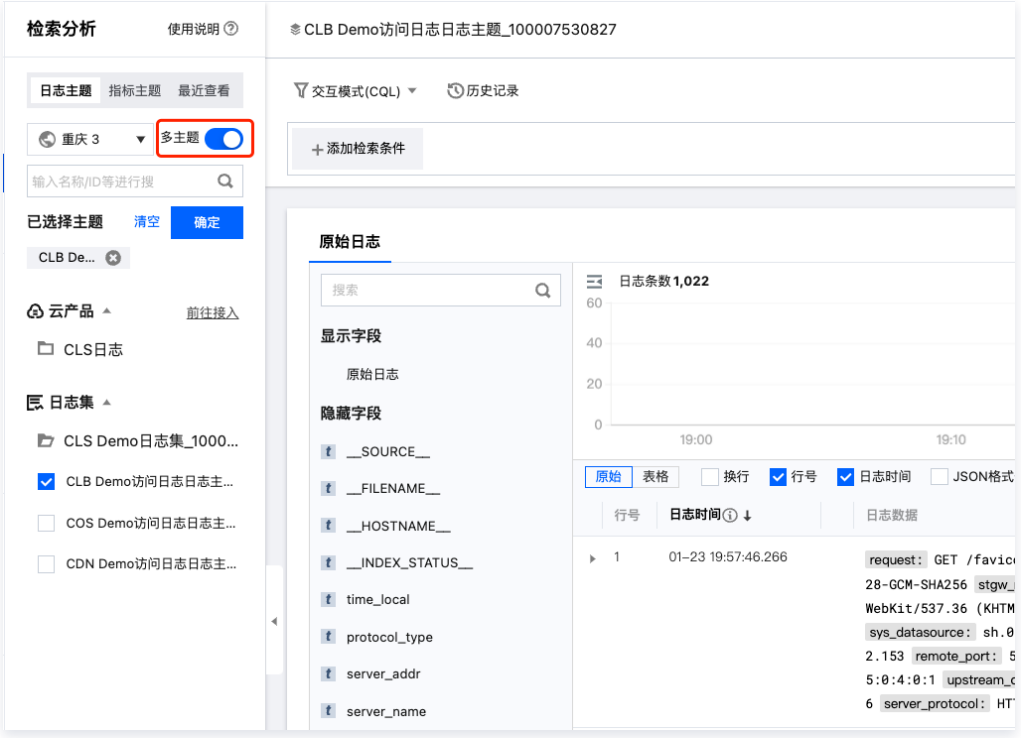
多主题检索用于集中检索分布在多个日志主题中的日志，例如通过 RequestID 跨日志主题检索相关日志。本文为您介绍如何使用多主题检索。

使用限制

- 支持跨日志集选择日志主题，但不支持跨地域选择日志主题。
- 最多可同时检索50个日志主题。
- 仅支持检索原始日志，暂不支持使用 SQL 进行统计分析。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**检索分析**，进入检索分析页面。
3. 开启左上角的**多主题**，进入多主题检索模式，然后选择需要检索的多个日志主题，单击**确定**。



4. 输入检索分析语句，在右侧选择时间范围，然后单击 ，执行检索。

> **说明：**
> 多个主题之间索引配置不一致时，检索语句可能在部分日志主题下执行报错，此时仅展示未报错日志主题的数据及报错信息。例如 A 主题具备字段 KeyA 并开启键值索引，B 主题不具备该字段，使用检索语句 `KeyA:xxx` 执行检索时，B 主题执行将报错，仅能查看 A 主题相关日志。

上下文检索

最近更新时间：2025-06-27 11:06:31

本文介绍如何在日志服务控制台查看指定日志在原始文件中的上下文信息。

操作场景

日志上下文检索是指把目标日志数据在原始文件中的前若干条日志（上文）或后若干条日志（下文）检索查询出来。通过查看指定日志的上下文信息，您可以在业务故障排查中快速查找故障信息，方便定位问题。

功能优势

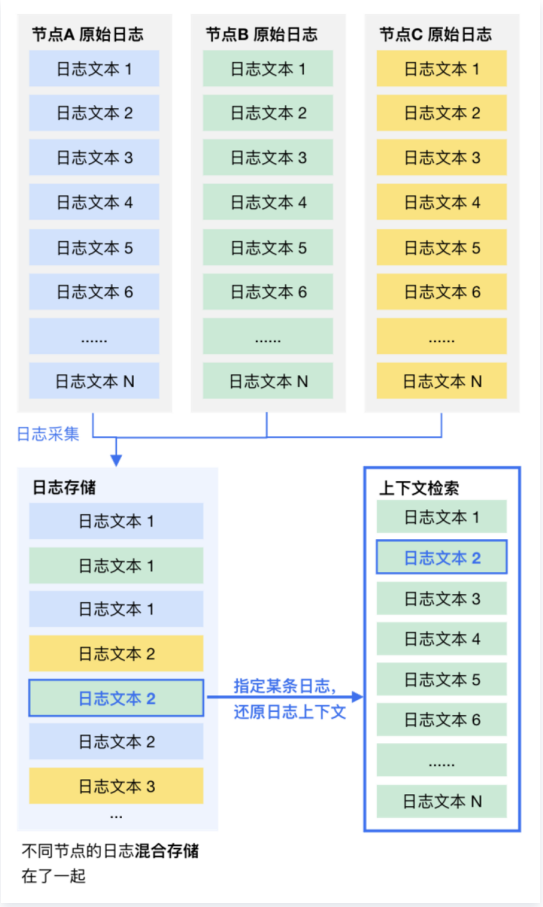
- 免除登录机器的繁琐步骤，在检索分析页面、快速查看任意机器，文件的指定日志的上下文信息。
- 结合事件发生的时间线索，在检索分析页指定时间段快速定位可疑日志后再进行上下文查询，快速定位问题。
- 不用担心服务器存储空间不足或日志文件轮转造成的数据丢失，在检索分析页上随时可以查看历史数据。

前提条件

- 上下文检索分析仅支持2.3.5以上版本使用，建议 安装或升级至最新版本 。
- 只有通过以下方式采集到的日志才支持上下文功能：
  - LogListener
  - GO SDK
  - JAVA SDK
  - fluent-bit-go-cls SDK
  - log4j SDK
  - log4j2 SDK
  - logback SDK
- 已开启并配置索引，详情请参见 配置索引 。

场景示例

订单交易轨迹：用户登录 > 浏览商品 > 选择物品 > 加入购物车 > 下单 > 订单支付 > 支付扣款 > 生成订单
使用场景：若用户的某一单交易失败，可先通过订单号查询错误日志，再根据查到的错误日志还原上下文，找到失败原因。例如扣款失败。
在日志服务中，可以按照以下步骤排查：
1. 在日志服务控制台的**检索分析**页面，结合事件发生的时间线索，输入**关键字**订单 ID 找到订单失败日志。
2. 以查到的日志为基准，向上或向下滚动直到发现与之相关的上下文日志信息。

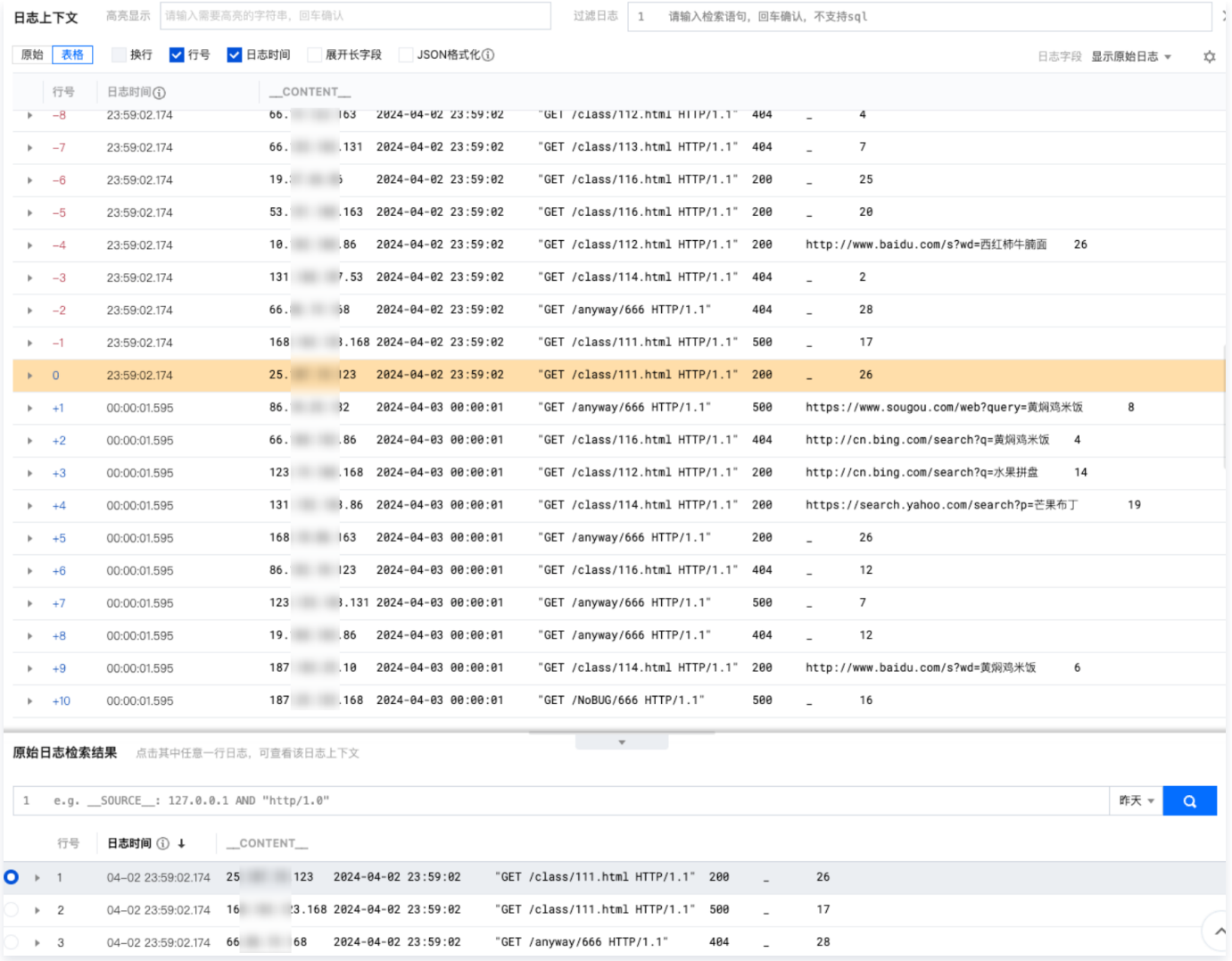


操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**检索分析**，进入检索分析管理页面。
3. 根据实际需求，选择**地域**和**日志主题**。
4. 在**原始日志**页签下，找到目标日志，单击 ，查看该条日志的上下文。

> **注意：**
> 需符合 前提条件 中采集方式的日志才支持上下文检索，可查看到该按钮，其他采集方式的日志将自动隐藏该按钮。

5. 在**日志上下文**页面，行号为0的黄色背景日志为当前目标日志，其上下分别为该日志的上下文。



6. 您可以在此页面进行如下操作：
   - 使用鼠标在当前页面上下滚动查看更多上下文日志，每次滚动默认加载50条日志，可点击右上角的设置按钮来设置每次加载日志条数。
   - 在**高亮显示**文本框中，输入关键字进行高亮显示，有多个关键字时将分别使用不同的颜色进行高亮。
   - 在**过滤日志**文本框中，输入检索语句对上下文日志进行过滤。
   - 在底部的**原始日志检索结果**中，可修改检索语句以查找其它需要查看上下文的目标日志，然后选中该目标日志，即可在上方查看该目标日志的上下文。

自定义跳转

最近更新时间：2025-07-03 17:35:32

操作场景

针对在检索分析页面查询到的原始日志，日志服务支持单击字段值进行自定义跳转，方便进行进一步的操作，例如：

- 根据 request_id 在其它日志主题检索相关日志。
- 根据 TraceID 在 应用性能监控 中查询调用链信息。
- 根据 user_id 在内部用户管理平台查询用户信息。



自定义跳转支持三种操作类型：

跳转类型	适用场景
打开外部 URL	打开指定的 URL，并携带日志中的指定字段作为 URL 中的参数，例如根据 user_id 在内部用户管理平台查询用户信息。
查看应用性能监控链路追踪详情	已在 应用性能监控 中将调用链与 CLS 日志 关联 时，可通过 TraceID 跳转至应用性能监控页面中查看调用链详情。
检索其他日志主题	检索指定日志主题，并携带日志中的指定字段作为检索条件，例如根据 request_id 在其他日志主题检索相关日志。

操作步骤

添加自定义跳转

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**检索分析**，进入检索分析页面。

3. 查询到日志后，单击需要添加跳转的字段值，在弹出的菜单中单击+，添加新的跳转。



4. 在弹框中填写如下配置信息：
   - 跳转类型：根据需要选择对应的跳转类型。
   - 触发字段：单击某个字段的值触发跳转，例如上图中的“stgw_request_id” 。
   - 配置名称：自定义跳转的名称，例如“查看下游服务日志”。
   - 其它配置项：与跳转类型有关，详见下表：

打开外部 URL

- URL：需要打开的 URL 地址，以 http:// 或 https:// 开头，URL 中支持使用 变量 动态获取日志字段值等信息，例如 `{{__currentValue__}}` 代表当前被单击的值。
- 新窗口打开：是否跳转至新窗口。

查看应用性能监控链路追踪详情

- 业务系统：选择已经在应用性能监控中与 CLS 日志 关联 完毕的 业务系统 。
- Trace ID：日志中哪部分值为Trace ID。
  - 触发字段字段值：适用于Trace ID 作为单独字段的情况。
  - 触发字段被单击分词：适用于Trace ID 未作为单独字段，被包含至长文本中的情况，例如下面的例子中 `5e72e39e55209533-1` 为Trace ID。

```
request [5e72e39e55209533-1] for operation PRODUCER rest demo-register-url-prefix-
server.RegisterUrlPrefixEndpoint.getName begin
```

- 新窗口打开：是否跳转至新窗口。

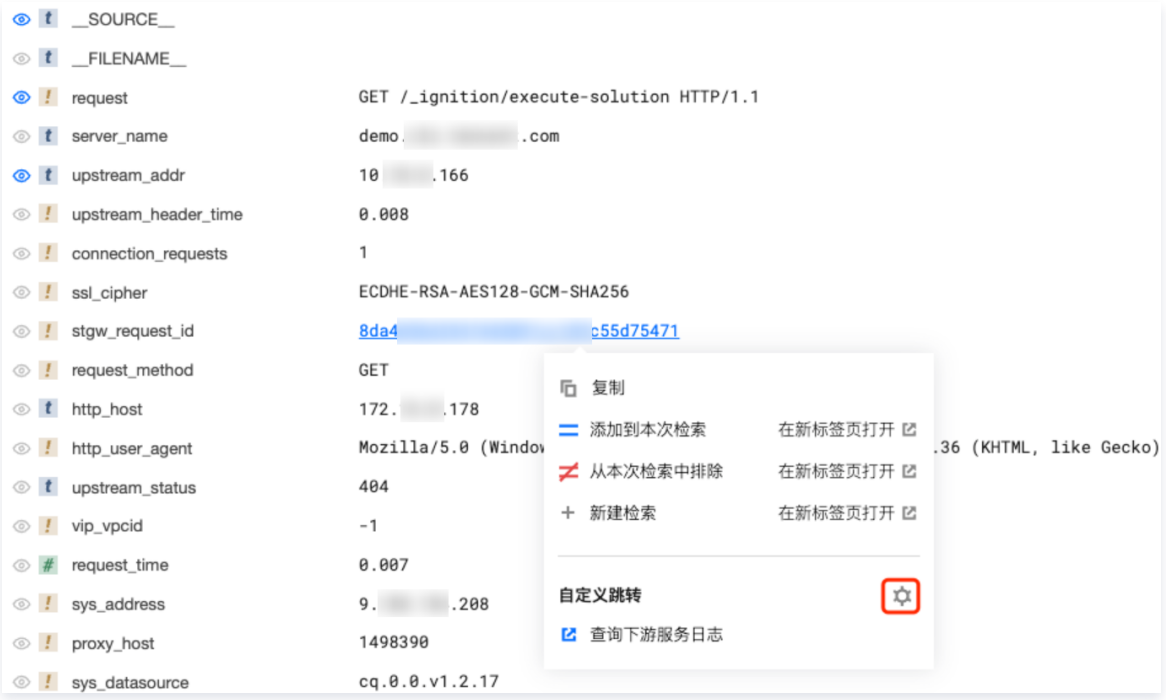
检索其他日志主题

- 日志主题：需要检索的日志主题。
- 时间范围：需要检索的时间范围，建议“沿用当前时间范围”，也可以自定义。
- 沿用当前检索条件：是否需要沿用当前的检索条件。

- 新增检索条件：是否需要新增检索条件，可使用 变量 动态获取日志字段值等信息，例如使用 `stgw_request_id:{{__currentValue__}}` 将当前被单击的值作为键值检索条件，在检索其它日志主题时，将自动转化为 `stgw_request_id:"8da469b42947445891cc10fc55d75471"` 出现在检索语句中。
- 新窗口打开：是否跳转至新窗口。

编辑/删除自定义跳转

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**检索分析**，进入检索分析页面。
3. 查询到日志后，单击需要编辑/删除跳转的字段值，在弹出的菜单中单击管理自定义跳转。



4. 在弹框中单击自定义跳转配置名称，在右侧表单中可编辑该配置。单击名称右侧可删除该配置。
5. 单击**应用**保存配置。

变量说明

- `${__CurrentValue}` 表示当前被单击的字段值，字段被分词时，该变量指的是分词后的词，例如下图，kube-scheduler 后方为分词符 `/` ，鼠标悬停时，仅高亮了 kube-scheduler，单击触发自定义跳转时，对应的 `${__CurrentValue}` 为 kube-scheduler。



- `${__TopicId}` 表示当前主题ID，例如 `a85bbd1c-233f-xxxx-aeda-70cbd9f8715a` 。
- `${__StartTime}` 和 `${__EndTime}` 表示当前查询时间范围的起止 Unix 时间戳。
- 可使用 `${}` 包裹字段名称作为变量，表示该字段的完整值，例如上图中可使用 `${userAgent}` 代表字段 userAgent 的值， `${userAgent}` = `kube-scheduler/v1.20.6 (linux/amd64) kubernetes/1cb721e/leader-election` 。

下载日志

最近更新时间：2024-11-21 14:37:11

操作场景

日志服务提供了日志数据导出的功能，您可以将采集到的日志数据下载至本地查看。
下载日志功能如下：

- 单次下载日志条数最多为5000万条。
- 支持指定检索条件，检索时间范围。
- 支持 JSON，CSV 导出格式。

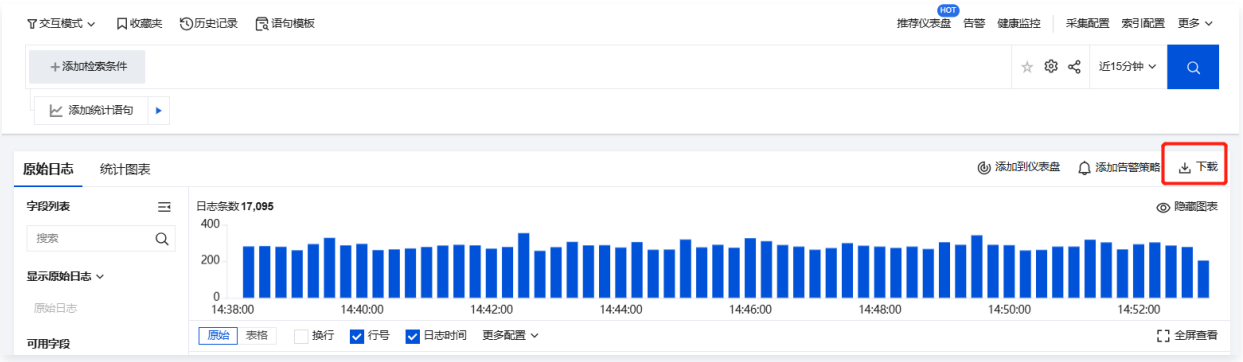
> **注意：**
> 下载日志的语句不支持 SQL，如需下载 SQL 结果，请在**统计图表** TAB 页的右上角，单击**导出分析数据**。

费用说明

下载日志会产生外网读流量费用，CLS 将按照 gzip 压缩后的大小来计量。例如：您的原始日志大小100GB，压缩后计量约为40GB - 70GB（源日志差异导致），外网读流量费用是0.8元/GB，则计费约为32 - 56元。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**检索分析**，进入检索分析管理页面。
3. 单击“日志集”和“日志主题”的下拉框，选择待检索的日志主题。
4. 单击日志时间，选择需要检索的日志时间。您可以选择最近时间、相对时间，也可以自定义时间范围。
5. 输入检索条件，单击 检索相关日志。
6. 待检索到日志数据后，单击 > **下载日志**。



> **说明：**
> 若未检索到日志数据，则无法下载日志。

7. 在弹出的窗口中，确认需下载日志的时间范围和检索语句，并根据如下信息设置下载日志数据。
   - 数据格式：提供了 “CSV 格式”和 “JSON 格式”两种格式，请按需选择。
   - 日志排序：提供了“升序”和“降序”两种规则，默认按时间降序。
   - 选择字段：打开开关之后，您可添加/删除需要下载的字段，字段总个数 <= 300个，不可使用逗号作为字段名。
   - 日志数量：默认导出“全部日志”，您也可以选择“自定义日志数量”，自定义导出所需日志数量。

> **说明：**
> - 需要下载的日志条数超出5000万时，建议缩小日志检索范围（例如采用更加精确的检索条件、缩小查询时间范围）或仅下载指定数量的日志。

- 如果确实需要下载超出5000万的日志，可通过限定日志查询时间范围创建多个下载任务，分批进行下载。

8. 单击**导出**，切换为**导出记录**界面。任务状态共包含“等待中”、“文件生成中”和“文件已生成”三个状态。新创建的下载任务会在等待片刻后进入“文件生成中”状态，并开始在后台生成数据文件。如有同时多个下载任务，系统会按照创建时间依次执行，创建较晚的任务会处于“等待中”。
9. 待任务状态变为**文件已生成**后，单击**下载**，即可导出日志。

注意：
导出的日志文件仅保留3天。

兼容 Elasticsearch 接口

最近更新时间：2025-03-31 16:26:32

操作场景

Elasticsearch 是一个开源的分布式搜索和分析引擎，日志服务兼容其部分接口。当您从 Elasticsearch 迁移到日志服务时，如果您之前使用的一些 Elasticsearch 接口在日志服务的兼容范围内，您可以继续使用这些接口，从而降低迁移的复杂度。
日志服务的兼容接口基于 Elasticsearch 7.10版本。

访问方式

- 外网地址：https://${region}.cls.tencentcs.com/elasticsearch/
- 内网地址：https://${region}.cls.tencentyun.com/elasticsearch/

其中 `${region}` 为 地域简称，请使用日志主题所在的地域。
接口采用 Basic Auth 进行鉴权，将 API 密钥 中的 SecretId 及 SecretKey 分别作为 username 及 password。

> **说明：**
> 建议您单独创建一个子账号，使用该账号的 SecretId 及 SecretKey。该账号仅授予如下接口权限即可，配置方式请参见 子账号授权 。

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

兼容接口

Search API

```
GET /${topic_id}/_search
POST /${topic_id}/_search
```

其中 `${topic_id}` 为日志主题 ID，相当于 Elasticsearch 中的 index。

兼容 DSL

日志需开启索引才能使用对应的 DSL，详情请参见 索引配置 。

> **说明：**
> 按照日志时间范围查询数据时，请使用 `__TIMSTAMP__` 字段代替 Elasticsearch 中的 `@timestamp` 字段。

查询 DSL

- term 及 terms：指定的字段需开启键值索引。
- match_phrase：指定的字段需开启键值索引。
- range：指定的字段需开启键值索引。

聚合 DSL

terms：指定的字段需添加键值索引并开启统计。

通用参数

- 分页与排序：from、size、sort，用作 sort 的字段需添加键值索引并开启统计。
- 结果控制：track_total_hits。

示例

查询最近30分钟 `reqMethod` 为 `GET` 的日志。

```
curl -X POST "https://ap-chongqing.cls.tencentcs.com/elasticsearch/58b9496c-c6c6-48b3-xxx-50824857a13c/_search" \
  -H "Content-Type: application/json" \
  -u AKIDBFvNbcOwLWTXXXXXXXXXXXXXXXXXXXXX:bx5zf6JI3wb18XXXXXXXXXXXXXXXXXXXX \
  -d '{
    "track_total_hits": true,
    "size": 10,
    "query": {
      "bool": {
        "filter": [
          { "match": { "reqMethod": "GET" } },
          {
            "range": {
              "__TIMESTAMP__": {
                "gte": "now-30m",
                "lte": "now"
              }
            }
          }
        ]
      }
    }
  }'
```

查询最近30分钟日志全文包含关键词 `USER` 的日志，并按照 `resHttpCode` 分组统计。

```
curl -X POST "https://ap-chongqing.cls.tencentcs.com/elasticsearch/58b9496c-c6c6-48b3-xxxx-50824857a13c/_search" \
  -H "Content-Type: application/json" \
  -u AKIDBFvNbcOwLWTXXXXXXXXXXXXXXXXXXXXX:bx5zf6JI3wb18XXXXXXXXXXXXXXXXXXXX \
  -d '{
    "query": {
      "bool": {
        "filter": [
          {
            "multi_match": {
              "type": "best_fields",
              "query": "USER",
              "lenient": true
            }
          },
          {
            "range": {
              "__TIMESTAMP__": {
                "gte": "now-30m",
                "lte": "now"
              }
            }
          }
        ]
      }
```

```
    },
    "aggs": {
      "http_code_distribution": {
        "terms": {
          "field": "resHttpCode",
          "size": 100,
          "min_doc_count": 1
        }
      }
    }
  }'
```

检索分析（指标主题）
语法规则

最近更新时间：2024-10-16 19:58:31

指标主题使用 Prometheus 的 PromQL 语法查询指标数据，详细语法说明可参见 Prometheus 官方文档 。本文主要介绍部分常用语法及操作步骤帮助您快速了解基本使用方法。
指标查询包含两种查询方式：

- 范围查询：即 Prometheus Range queries，在查询时间范围内，按照时间间隔（step）查询每个时刻的指标数据，常用于查询指标变化趋势。

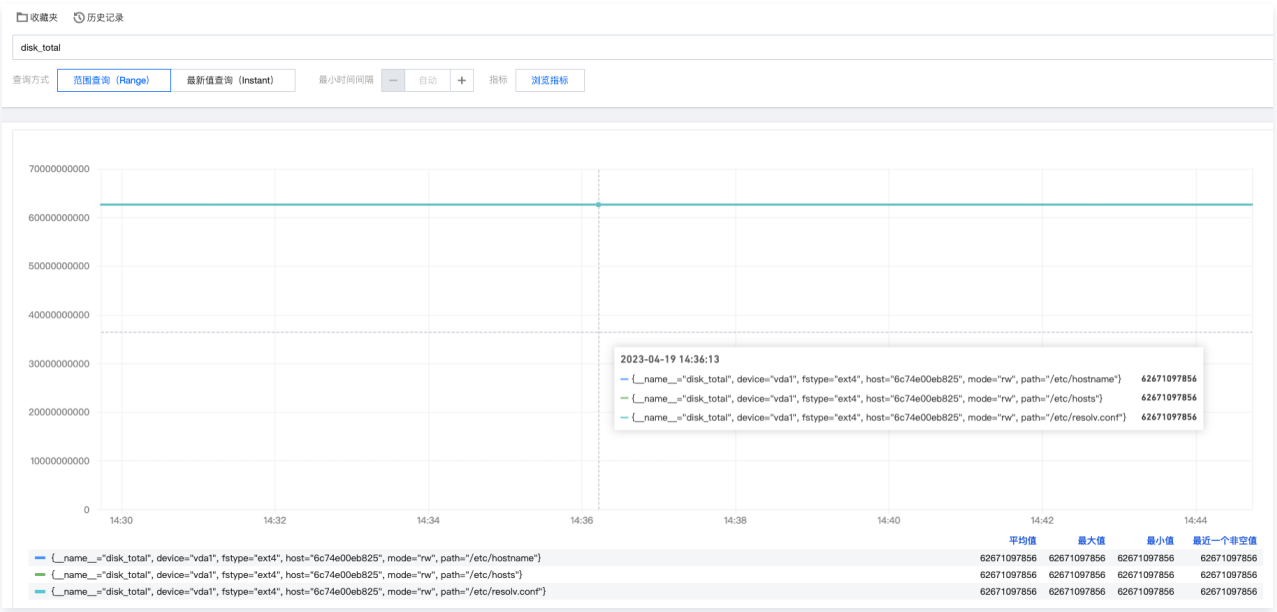
> **说明：**
> 查询时间间隔(step)小于指标采集间隔时，查询结果会出现重复数据，例如 CPU 使用率每15s上报一次，查询时间间隔为1s时会出现15个值相同的重复数据。

- 最新值查询：即 Prometheus Instant queries，返回查询时间范围结束时刻最新的指标数据，常用于查询当前指标值。如果结束时刻向前推5分钟内均无指标数据，则无相应的查询结果。

常用语法

1. 指定指标名称，查询指标名称下各个维度的指标值的变化趋势，例如查询磁盘容量（disk_total）：

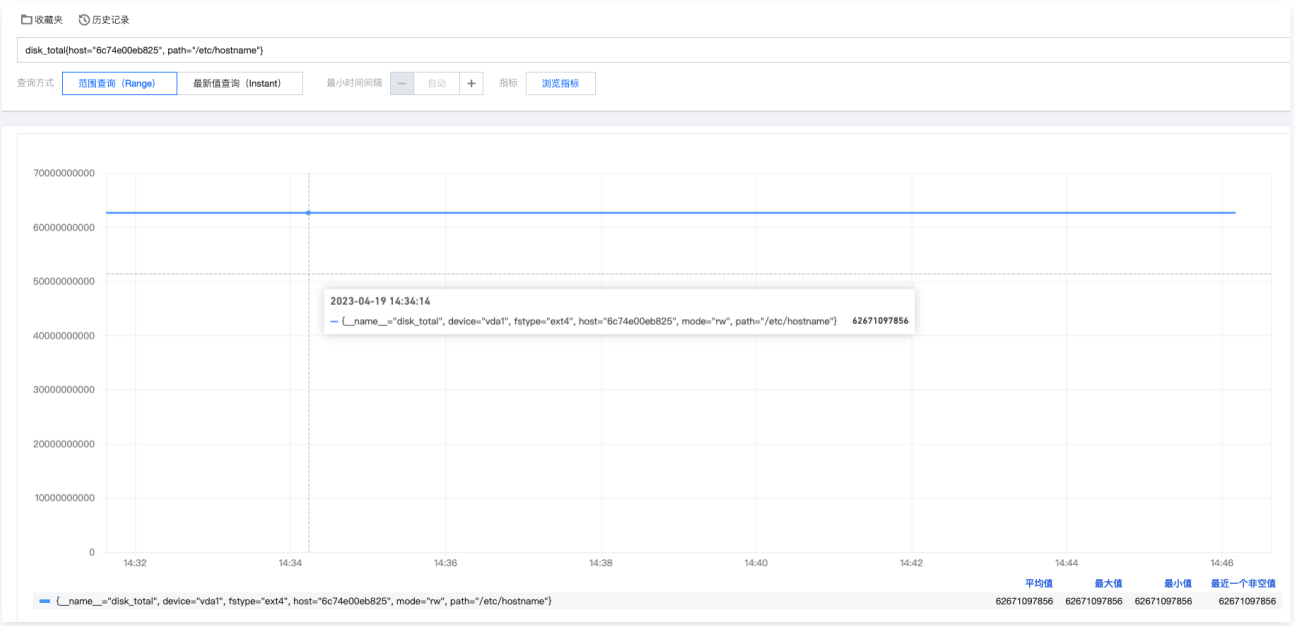
```
disk_total
```



2. 指定指标名称及维度，查询特定维度组合下的指标值变化趋势，例如主机名（host）为 6c74e00eb825，路径（path）为 /etc/hostname 的磁盘容量：

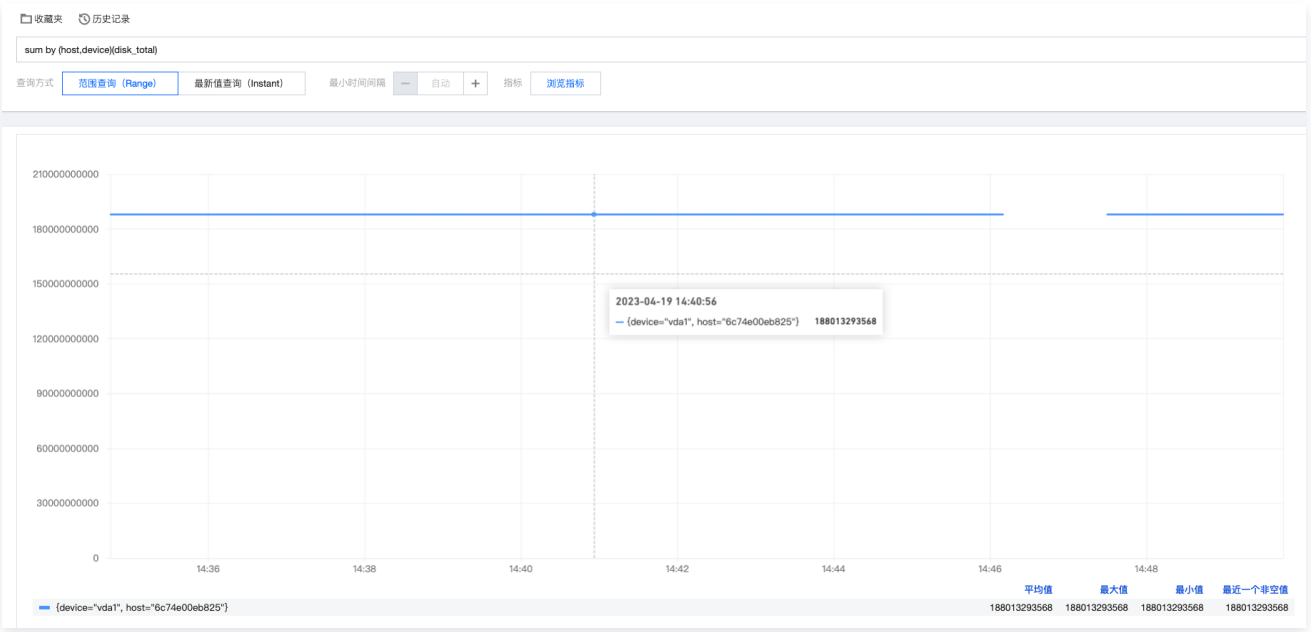
```
disk_total{host="6c74e00eb825", path="/etc/hostname"}
```

版权所有：腾讯云计算（北京）有限责任公司



3. 按照指定维度聚合指标数据，例如查询每台主机（host）、每个设备（device）的磁盘容量总和：

```
sum by (host,device)(disk_total)
```

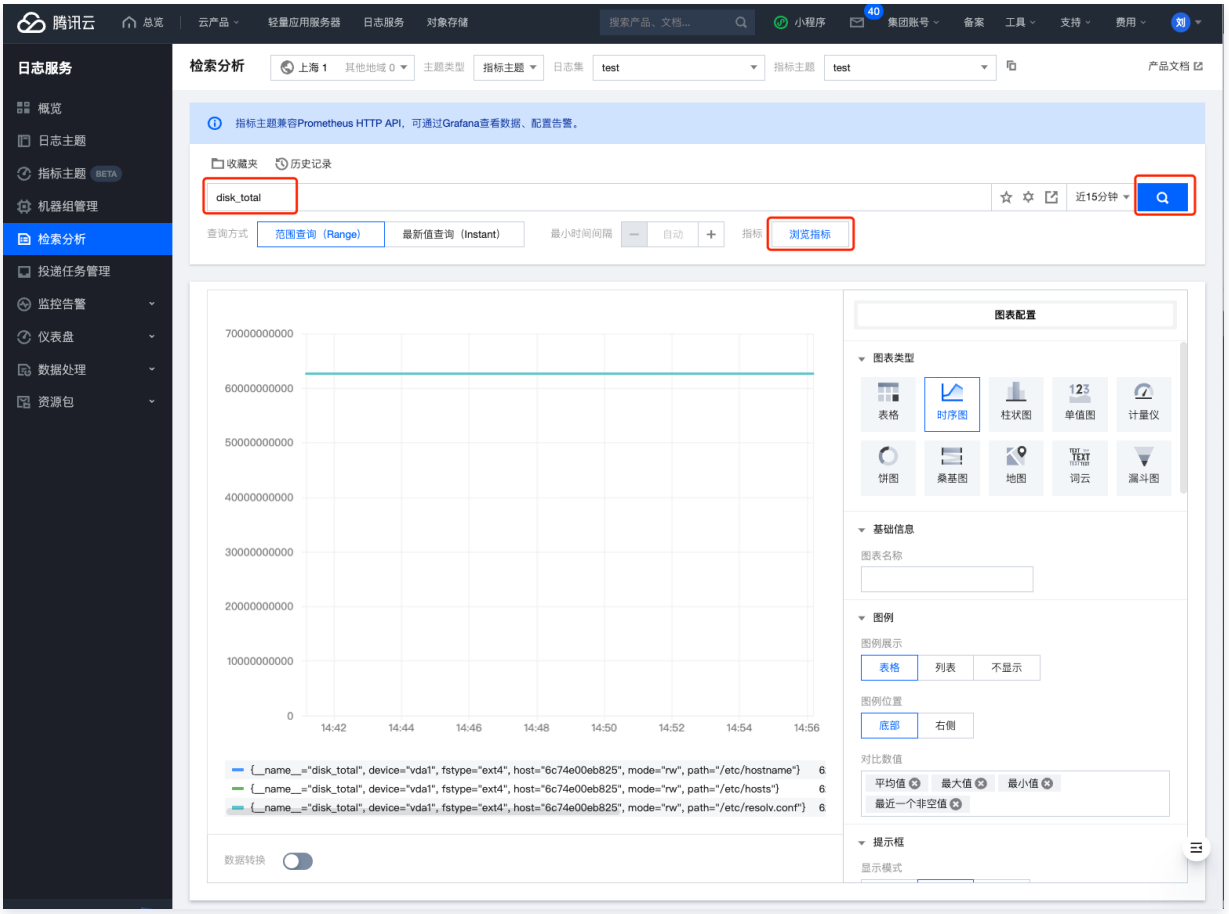


说明：

更多 PromQL 语法请查询 Prometheus 官方文档，也可参见 查询示例。

操作步骤

1. 登录 日志服务控制台。
2. 在左侧导航栏中，选择**检索分析**，在顶部选择指标主题所在的地域，主题类型选择“指标主题”，然后选择需要的指标主题。
3. 在输入框中输入查询语句，然后单击右侧 查询指标。您也可以单击**浏览指标**查询当前已有的指标名称。


腾讯云
总览
云产品
轻量应用服务器
日志服务
对象存储
搜索产品、文档...
小程序
集团账号
备案
工具
支持
费用
日志服务
概览
日志主题
指标主题
机器组管理
检索分析
投递任务管理
监控告警
仪表盘
数据处理
资源包
检索分析
上海 1
其他地域 0
主题类型
指标主题
日志集
test
指标主题
test
产品文档
指标主题兼容Prometheus HTTP API，可通过Grafana查看数据、配置告警。
收藏夹
历史记录
disk_total
近15分钟
查询方式
范围查询（Range）
最新值查询（Instant）
最小时间间隔
自动
指标
浏览指标
70000000000
60000000000
50000000000
40000000000
30000000000
20000000000
10000000000
0
14:42
14:44
14:46
14:48
14:50
14:52
14:54
14:56
{__name__="disk_total", device="vda1", fstype="ext4", host="6c74e00eb825", mode="rw", path="/etc/hostname"}
{__name__="disk_total", device="vda1", fstype="ext4", host="6c74e00eb825", mode="rw", path="/etc/hosts"}
{__name__="disk_total", device="vda1", fstype="ext4", host="6c74e00eb825", mode="rw", path="/etc/resolv.conf"}
数据转换
图表配置
图表类型
表格
时序图
柱状图
单值图
计量仪
饼图
桑基图
地图
词云
漏斗图
基础信息
图表名称
图例
图例展示
表格
列表
不显示
图例位置
底部
右侧
对比数值
平均值
最大值
最小值
最近一个非空值
提示框
显示模式

仪表盘
仪表盘概述

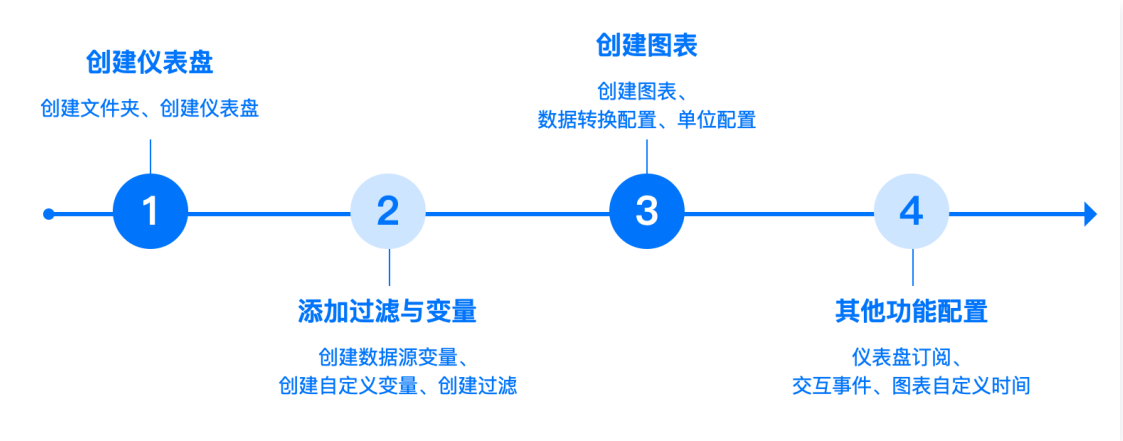
最近更新时间：2024-11-27 11:48:43

日志服务提供丰富的图表类型，支持用户使用 SQL 灵活统计日志内的系统及业务指标，并通过图表进行展示。用户还可以将图表分析结果保存到仪表盘，进行长期监控。

Demo 体验环境

您可点击 此处 前往仪表盘 Demo 体验环境。

使用路径



功能特性

使用路径	功能	功能描述	详细信息
1.创建仪表盘	创建文件夹	仪表盘支持创建文件夹，用以归类和收折仪表盘，便于管理	-
	创建仪表盘	日志服务支持多种创建仪表盘的方式： • 空白仪表盘：创建一个空白的仪表盘，需要用户手动添加图表并保存。 • 导入仪表盘：导入一个现有的 CLS 的仪表盘 Json 文件。Json 文件来自对已有仪表盘的导出。 • 从仪表盘模板创建（推荐）：CLS 提供了丰富的仪表盘模板，可以根据自己的场景与日志内容选择合适的模板一键启用，自动生成仪表盘	创建仪表盘
2.添加过滤与变量	过滤与变量	• 过滤：支持通过指定字段的字段值对仪表盘内所有图表的数据进行过滤。 • 变量：支持用户自己设置静态输入或动态查询的变量值，替换到检索语句、标题、文本。	过滤与变量
3.创建统计图表	表格	表格是最常见的数据展示类型，通过对数据结构化的整理，实现数据的对比与统计。大多数场景均适用。	表格
	时序图	时序图需要统计数据具备时序字段，依据时间顺序组织与聚合指标。可直观反映指标随时间的变化趋势。统计近一周，每天404错误出现的次数等趋势分析场景适用。	时序图
	柱状图	柱状图描述的是分类数据，直观表现每一个分类项的大小对比关系。统计近一天各错误码类型出现的次数等分类统计场景适用。	柱状图
	饼图	饼图描述的是不同分类的占比情况，通过扇区大小来衡量各分类项的占比情况。错误码占比情况分析等占比统计场景适用。	饼图
	单值图	单值图描述的是单个指标，一般选择具备有业务价值的关键性指标。统计天、周、月 PV、UV 等单指标场景适用。	单值图

	计量仪	计量仪描述的是单个指标，与单值图不同的是，计量仪一般搭配阈值使用，用来衡量该指标的状态。系统健康度监控等有分级标准的场景适用。	计量仪
	地图	地图通过图形的位置来表现数据的地理位置，通常来展示数据在不同地理区域上的分布情况。攻击 IP 地理分布等地理位置统计场景适用。	地图
	桑基图	桑基图是一种特定类型的流图，用于描述一组值到另一组值的流向。防火墙目的与源 IP 流量统计等场景适用。	桑基图
	词云图	词云是文本数据的视觉表示，用于展示文本数据的出现频率。在高频操作人员统计等审计场景适用。	词云图
	热力图	热力图支持通过对色块着色来显示数据的统计图表，适用于查看总体情况、发现异常值、显示多个变量之间的差异，以及检测它们之间是否存在任何相关性的场景。	热力图
	漏斗图	漏斗图适用于单流向单路径的业务流程，对各环节进行统计并用梯形面积表示某个环节业务量与上一个环节之间的差异。	漏斗图
	雷达图	雷达图是一种在多个变量间进行可视化和比较的图表。它通过从一个中心点出发的多条轴线，代表不同的数据类别或指标，然后在每条轴线上用点来表示数据值，最后将这些点连接起来形成一个多边形，从而展示数据之间的关系和趋势。	雷达图
	拓扑图	拓扑图是一种直观展示节点间关系的图表类型，在日志分析场景中可用于分析服务间的调用关系、系统架构情况以及数据流的方向等。	拓扑图
	日志	日志类型图表支持将原始日志保存到仪表盘查看，可以在仪表盘页面查看分析结果后，快捷的看到关联的日志内容，而无需跳转到检索分析页面。	日志
	文本	文本类型图表支持 MarkDown 语法，可以在仪表盘页面插入文本、图片链接、超链接等。	文本
	数据转换	数据转换支持用户对检索结果进行二次处理，包括字段类型修改、选择制图字段、合并分组等。可以在不修改 SQL 语句的情况下满足制图的需求。	数据转换
	单位换算	图表支持自动单位换算，选择一个原始单位后，当数值满足换算进率时，自动换算为更高一级别的单位。单位支持精度配置。	单位配置
4.其他功能配置	交互事件	图表自定义交互事件功能，支持单击图表内容触发交互打开检索分析页面、仪表盘页面、第三方 URL 等。	交互事件
	添加分组	仪表盘支持图表分组，通过分组可以对仪表盘图表内容进行归类和收折。	添加分组
	订阅仪表盘	日志服务支持订阅仪表盘，可以将仪表盘以图片的形式导出。用邮件、企业微信等方式定期发送给指定的对象。对于需要面向团队发送日、周、月报表的仪表盘，可使用该功能。	订阅仪表盘
	免密分享	免密分享支持将仪表盘或图表分享给其他人临时查看，无需登录鉴权，提高协作效率。	免密分享仪表盘
	图表时间配置	关闭使用全局时间后，图表时间独立控制，不再随全局时间的变化而变化，仪表盘的图表间可以有不同的时间范围，支持更丰富的对比场景。	图表时间配置

规格与限制

仪表盘的规格与限制，请您参见 仪表盘规格与限制 。

创建仪表盘

最近更新时间：2025-01-06 09:44:22

仪表盘提供数据分析监控的全局视图。您可以在仪表盘查看多个基于查询与分析结果的统计图表。本文介绍如何创建仪表盘及其相关操作。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**仪表盘 > 仪表盘列表**，进入仪表盘列表页面。
3. 在仪表盘列表页面，单击**创建文件夹**，新建一个自定义文件夹。（非必须操作）
4. 在仪表盘列表页面，单击**创建仪表盘**。在弹窗中选择创建仪表盘的方式：
   - **空白仪表盘**：创建一个空白的仪表盘，需要用户手动添加图表并保存。
   - **导入仪表盘**：
     - 导入 CLS 仪表盘文件：Json 文件来自对已有仪表盘的导出。
     - 导入 Grafana 的仪表盘文件：Json 文件从 Grafana 导出或从 Grafana Lab 下载。
   - **从仪表盘模板创建（推荐）**：CLS 提供了丰富的仪表盘模板，可以根据自己的场景与日志内容选择合适的模板一键启用，自动生成仪表盘。
5. 填写仪表盘相关信息。选择仪表盘关联的**标签**（非必须操作）。

配置项	说明
仪表盘名称	仪表盘名称不能重名
所属文件夹	用于分类仪表盘，支持文件夹模式显示仪表盘，支持按照文件夹过滤仪表盘
标签	用于仪表盘资源鉴权，支持按照标签过滤仪表盘，权限策略配置参见 CLS 访问策略模板

6. 单击**保存**，创建成功。

版权所有：腾讯云计算（北京）有限责任公司

创建统计图表
添加图表

最近更新时间：2025-06-24 15:47:31

日志服务（Cloud Log Service，CLS）支持对检索分析结果进行可视化图表统计，并支持将统计图表保存到仪表盘以方便持续监控查看，同时仪表盘也提供统计图表创建的入口，支持简易图表创建和自定义图表创建。

操作步骤

通过检索分析添加图表

> **注意：**
> 此方式需通过检索数据生成图表内容，详情请参见 日志分析 。

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**检索分析**，进入检索分析管理页面，填写检索分析语句。
3. 选择**统计图表**，单击**添加到仪表盘**。
4. 在弹出的窗口中，填写信息，单击**确定**。

表单元素	说明
图表名称	图表的名称，最多填写255个字符。
目标仪表盘	图表将要添加到的仪表盘。支持添加图表到已有仪表盘或选择新建一个仪表盘，并将图表添加进去。
仪表盘	仪表盘的名称，最多填写255个字符。
所属文件夹	选择仪表盘所属文件夹，仅在新建仪表盘时配置。

通过仪表盘添加图表

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**仪表盘** > **仪表盘列表**。
3. 选择目标仪表盘进入编辑页面。

> **说明：**
> 如果您没有仪表盘，请 创建仪表盘 。

4. 单击**添加图表**，图表编辑页面。
   - 自定义图表：通过自定义检索分析语句创建图表，则支持全部统计分析功能。
     - 选择数据源，并填写检索分析语句，日志主题与指标主题语法规则不同，请参见以下链接。
       - 日志主题：语法规则
       - 指标主题：语法规则
     - 图表支持添加多个语句绘图，详情请参见 多语句绘图示例 。
   - 简易图表：

图表元素	说明
选择字段	展示当前日志主题字段信息的列表，可以单击或直接鼠标拖拽添加到右侧条件输入框。
指标	指标是衡量事物某个特征的参数，一般为数值类型字段。可以从左侧字段列表拖入字段，也可以单击“+”号触发下拉菜单选择字段。字段填充后可以单击字段旁的设置图标，修改字段的聚合计算方式，系统默认使用 “平均值” 。
维度	维度是分析指标的视角分组，一般为字符串类型描述事物属性的字段。可以从左侧字段列表拖入字段，也可以单击“+”号触发下拉菜单选择字段。

版权所有：腾讯云计算（北京）有限责任公司

过滤	过滤字段用于数据的属性过滤。可以从左侧字段列表拖入字段，也可以单击“+”号触发下拉菜单选择字段。字段填充后可以单击字段旁的设置图标，修改字段的过滤方式，系统默认使用“字段存在” 。
排序	排序字段用于统计结果的排列。排列字段仅支持依据条件字段排列，其他非条件字段不可用。建议单击“+”号触发下拉菜单选择字段。字段填充后可以单击字段旁的设置图标，修改排序方法，系统默认使用 “升序” 。
行数限制	行数限制用于统计结果的数量过滤。设置后将按照排序从前往后显示设定数量的统计结果。行数限制取值范围为1 – 1000，系统默认使用1000。

5. 检查数据配置是否符合要求，系统将会根据图表数据结构要求自动填充。若数据类型不匹配，需要手动通过**数据转换**调整数据配置，详情请参见 数据转换 。
6. 单击**应用到仪表盘**。

图表的其他操作



添加的图表支持以下快捷操作：

- **拖拽图表尺寸大小**：长按图表右下角的箭头并拖拽，可以调整当前图表尺寸。
- **编辑图表**：单击进入图表编辑页面。
- **在检索分析查看**：单击跳转到检索分析页面，并打开该图表对应的统计分析。
- **添加到监控告警**：添加该图表的查询分析语句到告警策略配置。
- **全屏展示**：全屏展示该统计图表。
- **更多**：包括更多仪表盘操作，单击 ••• 打开：
  - 编辑图表
  - 预览检索分析语句
  - 添加到监控告警
  - 创建免密分享
  - 全屏查看
  - 复制到当前仪表盘
  - 复制到其他仪表盘
  - 保存为指标
  - 数据导出
  - 删除图表

表格

最近更新时间：2025-07-03 15:06:02

表格是最常见的数据展示类型，通过对数据结构化的整理，实现数据的对比与统计。大多数场景均适用。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**表格**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。

表格样式	• 自动换行：开启后文本超过表格宽度时将自动换行显示。不开启，则不自动换行，超出部分将被省略显示。
	• 列宽模式：表格的默认列宽。
	  ○ 自动：根据每列内容的文本长度，自动适配列宽。
	  ○ 等宽：每列宽度等宽。
	• 分页大小：配置表格分页的行数，取值范围20 – 1000。
	• 列最小宽度：配置表格列的最小列宽，取值范围50 – 500。
	• 对齐：设置表格内容在单元格里的对齐方式。默认配置下，指标类型的字段**右对齐**，维度类型的字段**左对齐**。
	• 列过滤：支持对列的内容进行过滤。

标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。
交互事件	表自定义交互事件功能，支持单击图表内容触发交互打开检索分析页面、仪表盘页面、第三方 URL 等。详情请参见 交互事件 。

阈值配置

配置项	说明
阈值配置	设置阈值点，可以使符合阈值的行或列高亮显示。

示例：在图表配置 > 阈值配置中配置阈值，选择阈值样式为文本高亮显示或背景高亮显示。
配置后查看表格，符合阈值的行或列高亮显示。

t status	# log_count
200	26743
302	23192
304	3236
307	3373
308	3226
400	12792
404	78479
431	92
499	195

字段配置

配置项	说明
字段配置	字段配置支持仅对所选字段生效的配置项，该配置会覆盖默认的图表配置。

示例：当表格有多个指标时，如想把 Error 日志数、错误率同时显示在同一行，并带上单位。Error 日志数单位为 “次数”，错误率单位为 “%”，单位不同。此时选择添加字段配置，选择“Error 日志数”，单位配置为“次数”，选择“错误率”，单位配置为“%”，实现不同的字段配置不同的效果。

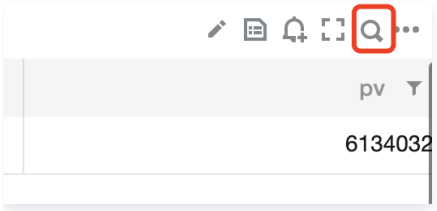
版权所有：腾讯云计算（北京）有限责任公司



图表操作

搜索表格内容

在表格顶部的搜索框中，输入搜索内容的关键词，单击搜索。



按照某一列排序

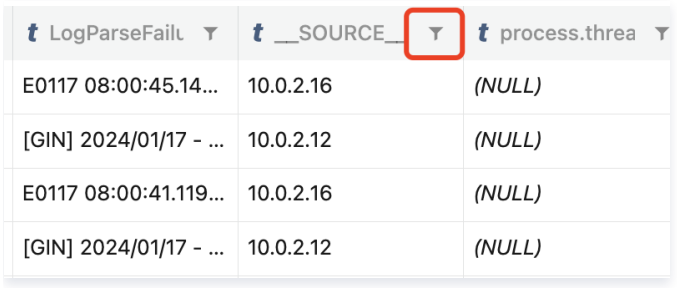
单击需要排序的列的表头字段名，将会按照升序排列。再次单击，切换为降序排列。



server	入流量(KB)	出流量(KB)	总流量(MB)
192.168.0.23	95.6	20002.93	19.63
192.168.0.206	94.68	19217.17	18.86
192.168.0.106	93.16	25394.2	24.89
192.168.0.224	92.6	17762.99	17.44
192.168.0.216	92.08	16423.64	16.13
192.168.0.43	87.37	25059.93	24.56
192.168.0.252	79.11	17114.31	16.79
192.168.0.134	78.75	16614.44	16.3
192.168.0.1	76.16	18497.78	18.14

按照某一列过滤

单击需要排序的列的表头的过滤按钮，选择字段值进行过滤。



LogParseFailu	__SOURCE__	process.threa
E0117 08:00:45.14...	10.0.2.16	*(NULL)*
[GIN] 2024/01/17 - ...	10.0.2.12	*(NULL)*
E0117 08:00:41.119...	10.0.2.16	*(NULL)*
[GIN] 2024/01/17 - ...	10.0.2.12	*(NULL)*

时序图

最近更新时间：2025-07-03 15:06:02

时序图需要统计数据具备时序字段，依据时间顺序组织与聚合指标。可直观反映指标随时间的变化趋势。统计近一周，每天404错误出现的次数等趋势分析场景适用。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**时序图**。
3. 按照需求配置图表。

通用配置

配置项	说明
图表名称	设置图表的显示名称，可为空。
图例	设置图表的图例内容，可以控制图例的样式与位置。同时也支持在图例中添加对比数据。
悬浮提示	设置鼠标悬浮触发的气泡提示的内容样式。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。
交互事件	自定义交互事件功能，支持单击图表内容触发交互打开检索分析页面、仪表盘页面、第三方 URL 等。详情请参见 交互事件 。

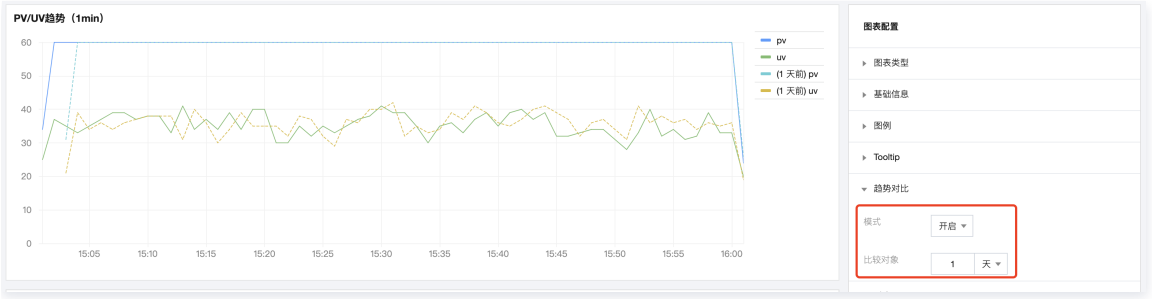
统计分析字段

时序图必需时间字段和指标字段，当碰到数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
时间字段	用作时序图 X 轴的时间类型字段，默认选择数据类型为 time 的字段。当没有 time 类型字段时，需手动选择指定字段。
维度字段	分类字段，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

趋势对比

配置项	说明
趋势对比	开启趋势对比后，可以选择和 X 小时、天、月、年以前的同周期数据进行对比。对比数据以虚线的方式显示在图表中。

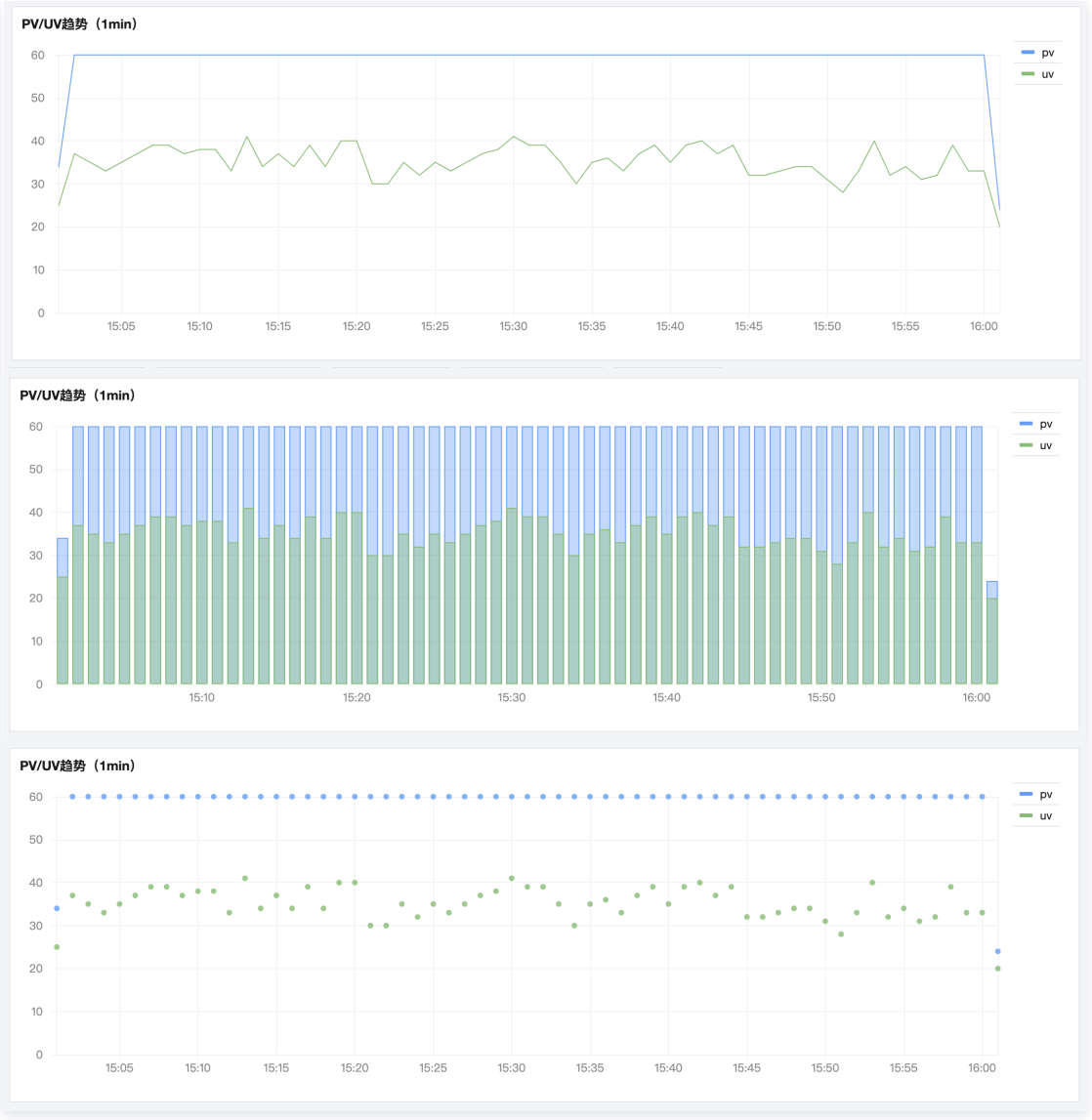


时序图配置

配置项	说明
时序图样式	• 绘制样式：数据在坐标轴上的显示样式，选择线则为折线图，选择柱则为直方图，选择点则为散点图。 • 连线方式：设置点与点之间的连线使用直线还是平滑曲线。 • 线宽：控制线条的粗细。

	• 填充：设置区域填充的样式。 • 不透明度：控制填充区域的不透明度 • 显示点：显示数据点，无数据则不显示。 • 点的大小：设置点的大小。 • 空值：线上空值点的处理，默认连接。 • 堆叠方式：是否将数据堆叠显示。 • 指标最大显示数量：默认显示的最大指标（线条）数量，超出的将被收折隐藏。
坐标轴	• Y 轴位置：显示/隐藏 Y 轴，及 Y 轴的位置。 • Y 轴最大值/Y 轴最小值：设置 Y 轴显示的最大值与最小值，大于最大值，小于最小值的坐标区域不显示。 • Y 轴尺度：控制 Y 轴的尺度，支持线性尺度与对数尺度。

- 绘制样式示例：



- 填充示例：

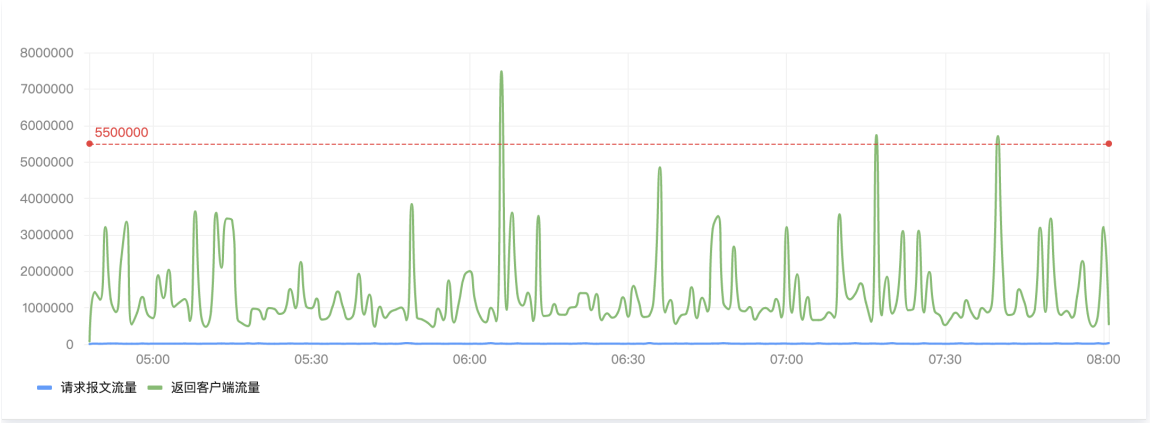

PV/UV趋势（1min）
60
50
40
30
20
10
0
15:55
16:00
16:05
16:10
16:15
16:20
16:25
16:30
16:35
16:40
16:45
16:50
pv
uv
PV/UV趋势（1min）
60
50
40
30
20
10
0
15:55
16:00
16:05
16:10
16:15
16:20
16:25
16:30
16:35
16:40
16:45
16:50
pv
uv

- **空值示例：**



阈值配置

配置项	说明
阈值配置	• 阈值点：设置阈值点，可以添加多个阈值区间，单击阈值对应色彩可打开色盘自定义颜色。 • 阈值展示：控制阈值展示的样式，包含阈值线、区域填充、阈值线和区域填充三种模式。选择关闭时，不使用阈值。



字段配置

配置项	说明
字段配置	字段配置支持仅对所选字段生效的配置项，该配置会覆盖默认的图表配置。

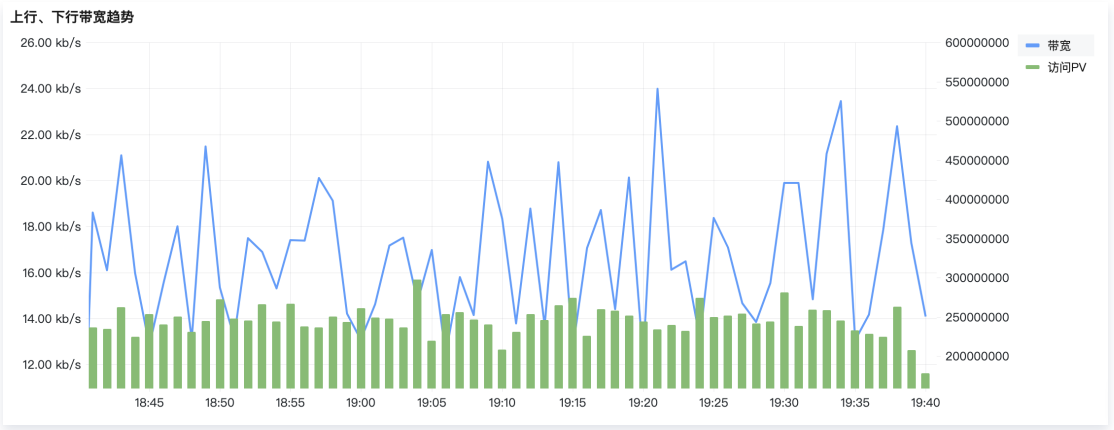
示例：

- **实现折线与直方图叠加效果**

  当表格有多个指标时，想使用折线图展示带宽，使用直方图显示访问 PV。可选择**添加字段配置**，选择**访问 PV**，单位配置为“次数”，绘制样式选择“柱”，填充选择“100”，实现目标效果。



折线与直方图叠加效果：

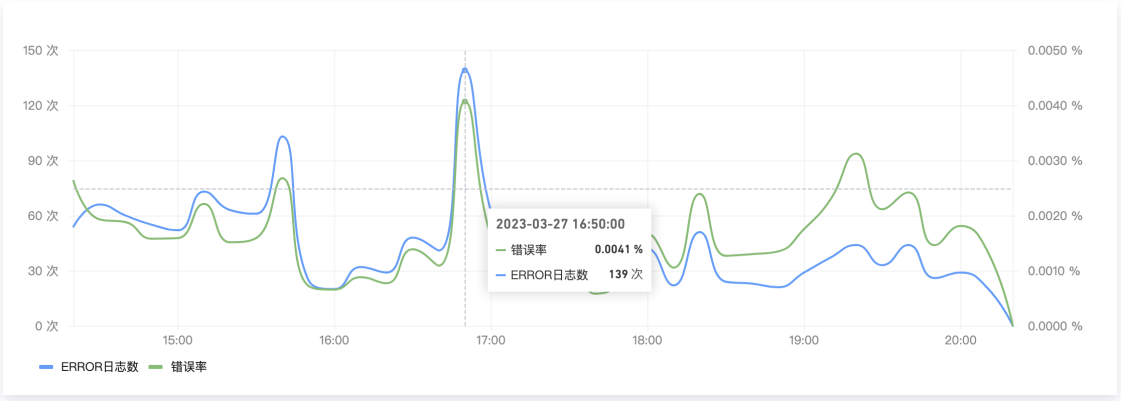


- **实现双坐标轴效果**

  当时序图有多个指标时，如想把 Error 日志数、错误率同时显示在同一图上，并带上单位。Error 日志数单位为 “次数”，错误率单位为 “%”，单位不同。此时选择**添加字段配置**，选择“Error 日志数”，单位配置为“次数”，选择“错误率”，单位配置为“%”，实现双坐标轴的效果。

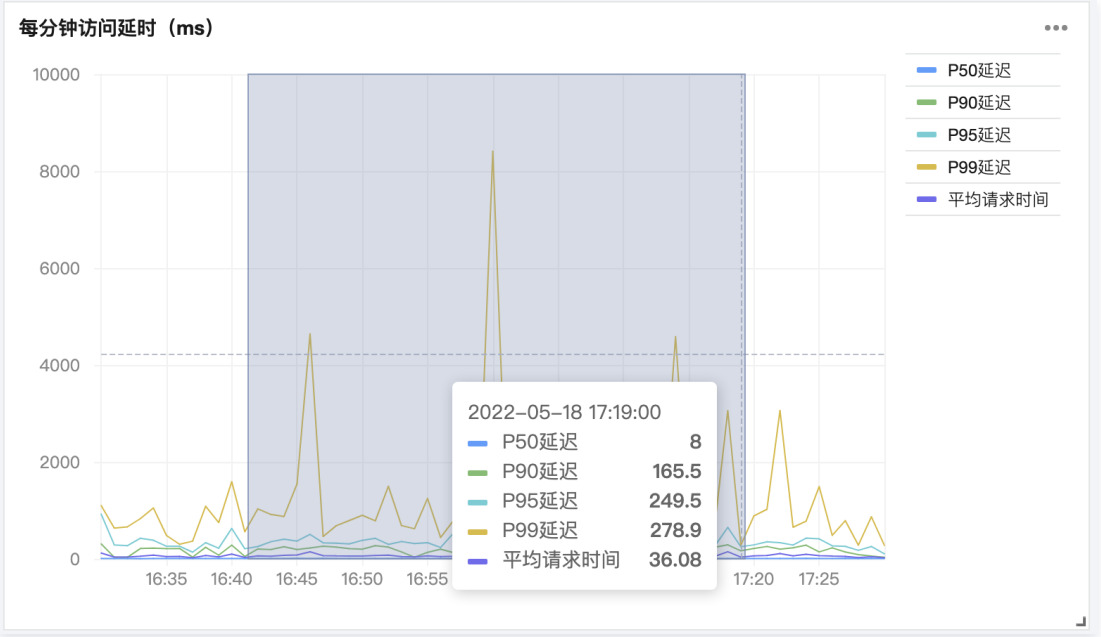


您可查看双坐标轴的效果：



图表操作

框选时间范围



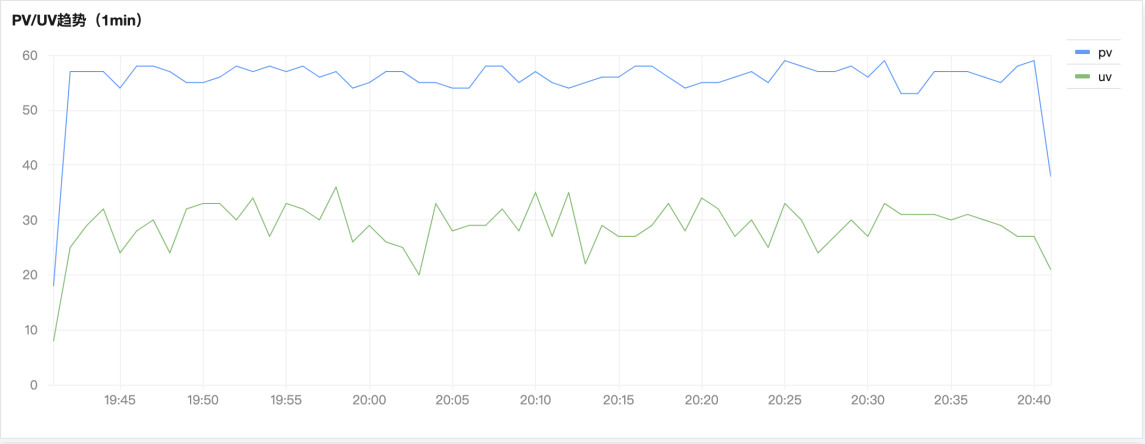
鼠标悬浮在图表上，长按后拖拽触发框选，时间范围将会使用框选区域的时间。适用于异常点的时间范围下钻等场景。

语句示例

时序图制图需要具备时间类型的字段，使用需依赖各种函数来对时间字段进行处理。更多的时序图时间函数请参见 时间与日期函数 。

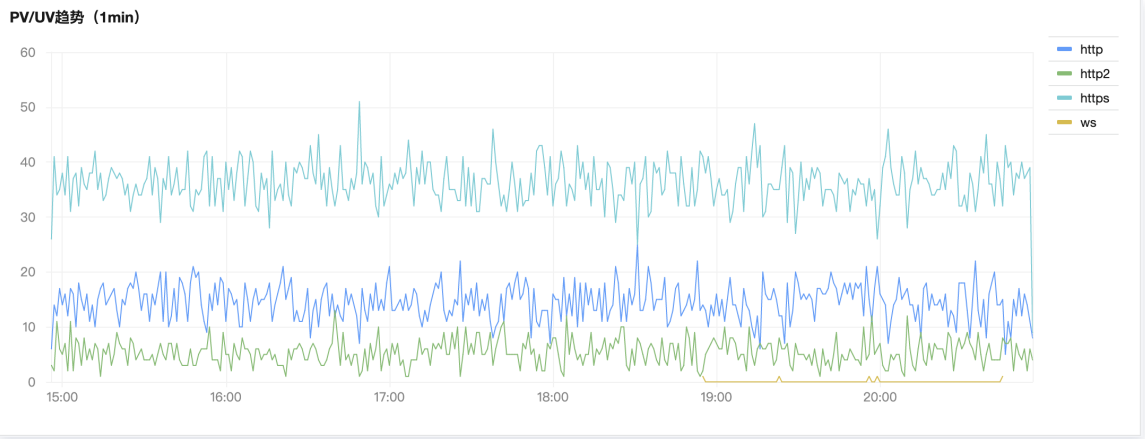
计算每分钟的 PV 与 UV：

```
* | select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 minute) as time, count(*) as pv,count(
distinct remote_addr) as uv group by time order by time desc limit 10000
```



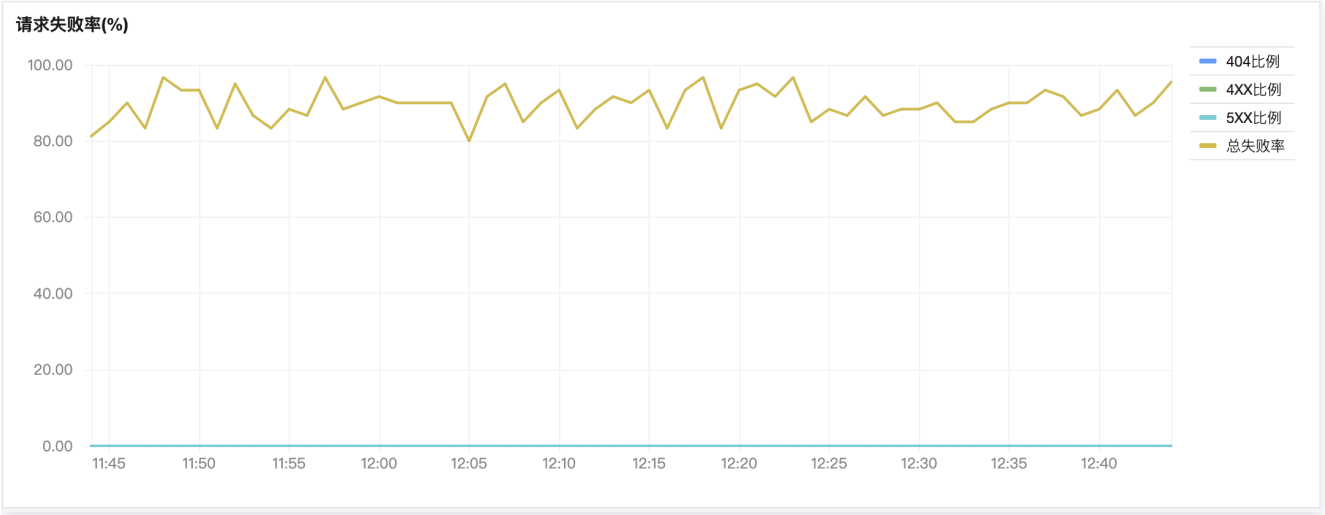
计算每分钟各协议类型的 PV：

```
* | select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 minute) as time, protocol_type, count(*)
as pv group by  time, protocol_type order by time desc limit 10000
```



计算每分钟的请求失败率(%)：

```
* | select date_trunc('minute', __TIMESTAMP__) as time, round(sum(case when status = 404 then 1.00 else
0.00 end)/ cast(count(*) as double)*100,3) as "404比例", round(sum(case when status >= 500 then 1.00 else
0.00 end)/cast(count(*) as double)*100,3) as "5XX比例", round(sum(case when status >= 400 and status < 500
then 1.00 else 0.00 end)/cast(count(*) as double)*100,3) as "4XX比例", round(sum(case when status >= 400
then 1.00 else 0.00 end)/cast(count(*) as double)*100,3) as "总失败率" group by time order by time limit
10000
```


请求失败率(%)
100.00
80.00
60.00
40.00
20.00
0.00
11:45
11:50
11:55
12:00
12:05
12:10
12:15
12:20
12:25
12:30
12:35
12:40
404比例
4XX比例
5XX比例
总失败率

柱状图

最近更新时间：2025-07-03 15:06:02

柱状图描述的是分类数据，直观表现每一个分类项的大小对比关系，适用于统计近一天各错误码类型出现的次数等分类统计场景。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**柱状图**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
图例	设置图表的图例内容，可以控制图例的样式与位置。同时也支持在图例中添加对比数据。
悬浮提示	设置鼠标悬浮触发的气泡提示的内容样式。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。
交互事件	自定义交互事件功能，支持单击图表内容触发交互打开检索分析页面、仪表盘页面、第三方 URL 等。详情请参见 交互事件 。

统计分析字段

柱图必须选择**维度字段**和**指标字段**，当碰到数据不匹配的情况时，需手动选择统计分析字段。

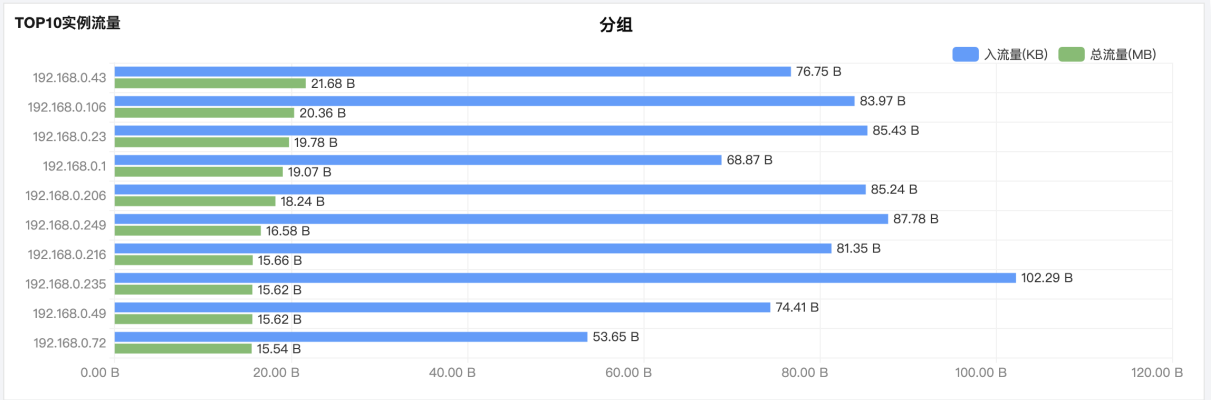
配置项	说明
维度字段	分类字段，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

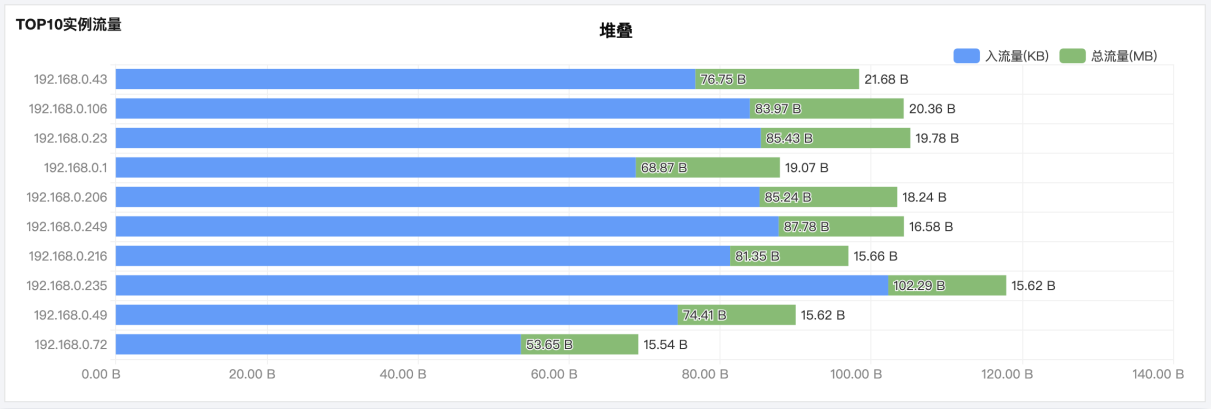
柱状图配置

配置项	说明
柱状图样式	• 条柱方向：控制条柱的方向，水平方向即为条形图，垂直方向为柱形图。 • 显示数值：控制是否显示每个条柱的数值标签。 • 堆叠方式：支持条柱分组（不堆叠）或堆叠显示。 • 排序方式：控制条柱的排序方式，支持按照指标升序与降序排列。在有多个指标时需要指定一个用于排序的指标，默认不排序。
坐标轴	• Y 轴标签最大显示长度。 • Y 轴尺度：控制 Y 轴的尺度，支持线性尺度与对数尺度。 • Y轴最大值/Y轴最小值：控制坐标轴显示的最大值与最小值，大于最大值或小于最小值的坐标区域不显示。

版权所有：腾讯云计算（北京）有限责任公司

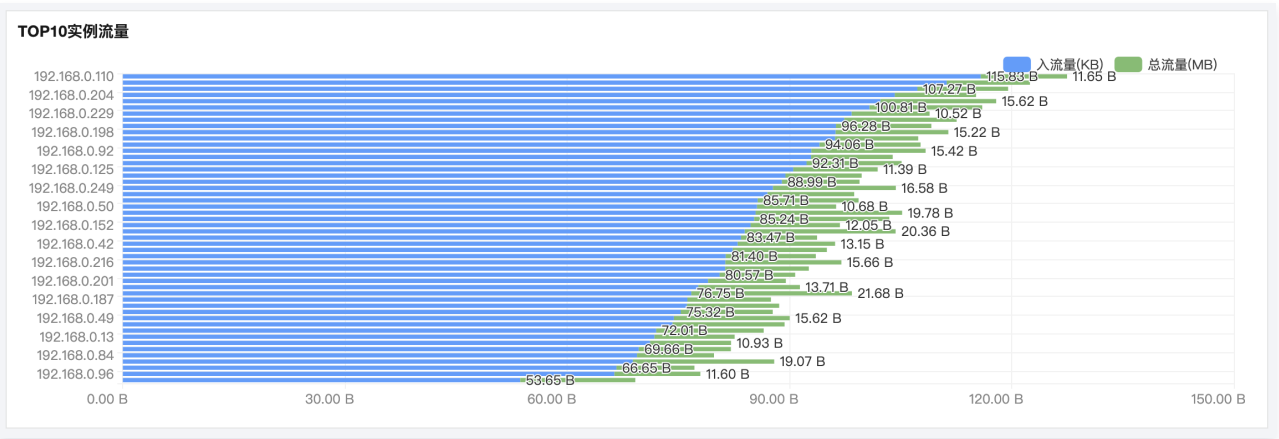
条柱模式示例：





图表操作

局部放大



在使用图表操作中，当统计结果过多，会造成上图所示问题：条柱密集、标签发生重叠隐藏，会影响分析的情况。此时，鼠标悬浮在图表上，滑动鼠标滚轮，可以放大或缩小显示范围，从而聚焦局部内容，显示完整的信息。

语句示例

Nginx 日志显示 HTTP 状态码占比：

```
* | select try_cast(status as varchar) as status, count(*) as log_count group by status
```

饼图

最近更新时间：2025-07-03 15:06:02

饼图描述的是不同分类的占比情况，通过扇区大小来衡量各分类项的占比情况。错误码占比情况分析等占比统计场景适用。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**饼图**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
图例	设置图表的图例内容，可以控制图例的样式与位置。同时也支持在图例中添加对比数据。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。
交互事件	自定义交互事件功能，支持单击图表内容触发交互打开检索分析页面、仪表盘页面、第三方 URL 等。详情请参见 交互事件 。

统计分析字段

饼图至少选择一个**指标字段**，当碰到数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
维度字段	分类字段，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

饼图配置

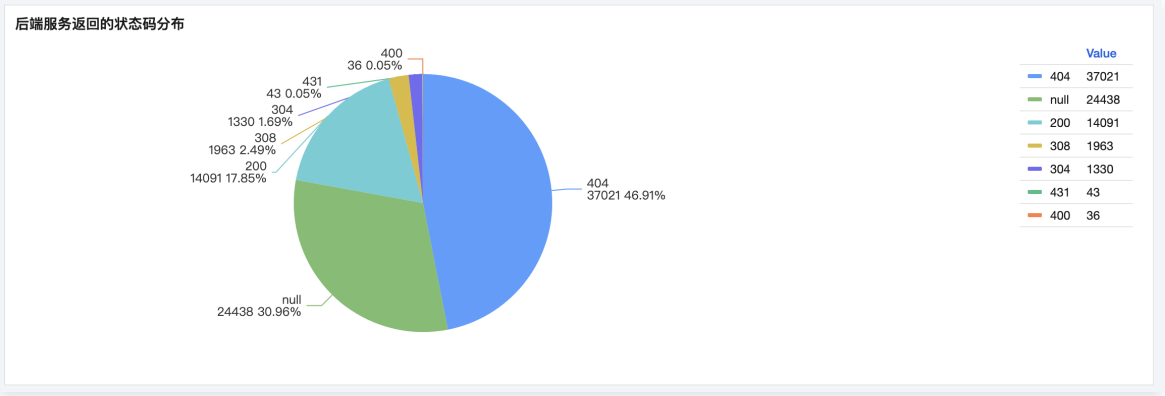
配置项	说明
饼图样式	• 饼图类型：控制饼图的样式，包括饼图、环形图。 • 指标聚合：高级功能，选择指标聚合方式，聚合相同指标的结果后再绘图（例如，指标返回了指标的范围值，包含了多个时序点的指标结果，先聚合所有时序点的总和再绘图）。默认不聚合，可设置指标最大的显示数量，超出部分将隐藏。 • 排序方式：控制扇区的排序方式，支持升序或降序，默认不排序。 • 扇区合并：合并 TOPN 以外的扇区为一个其他分区。当扇区分类过多时，可用此功能聚焦 TOPN 对象，默认不合并。 • 标签：显示饼图标签，可选择标签内容为：名称、数值、百分比中的一项或多项。

语句示例

Nginx 日志显示 HTTP 状态码占比：

```
* | select try_cast(status as varchar) as status, count(*) as log_count group by status
```

版权所有：腾讯云计算（北京）有限责任公司


后端服务返回的状态码分布
400
36 0.05%
431
43 0.05%
304
1330 1.69%
308
1963 2.49%
200
14091 17.85%
404
37021 46.91%
null
24438 30.96%
Value
404 37021
null 24438
200 14091
308 1963
304 1330
431 43
400 36

单值图

最近更新时间：2025-07-03 15:06:02

单值图描述的是单个指标，一般选择具备有业务价值的关键性指标。统计天、周、月 PV、UV 等单指标场景适用。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**单值图**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。

统计分析字段

单值图必需指标字段，当碰到数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
维度字段	分类字段，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择第一个数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

趋势对比

配置项	说明
趋势对比	开启趋势对比后，可以选择和 X 小时、天、月、年以前的同周期数据进行对比。趋势对比可以选择绝对值与百分比两类。

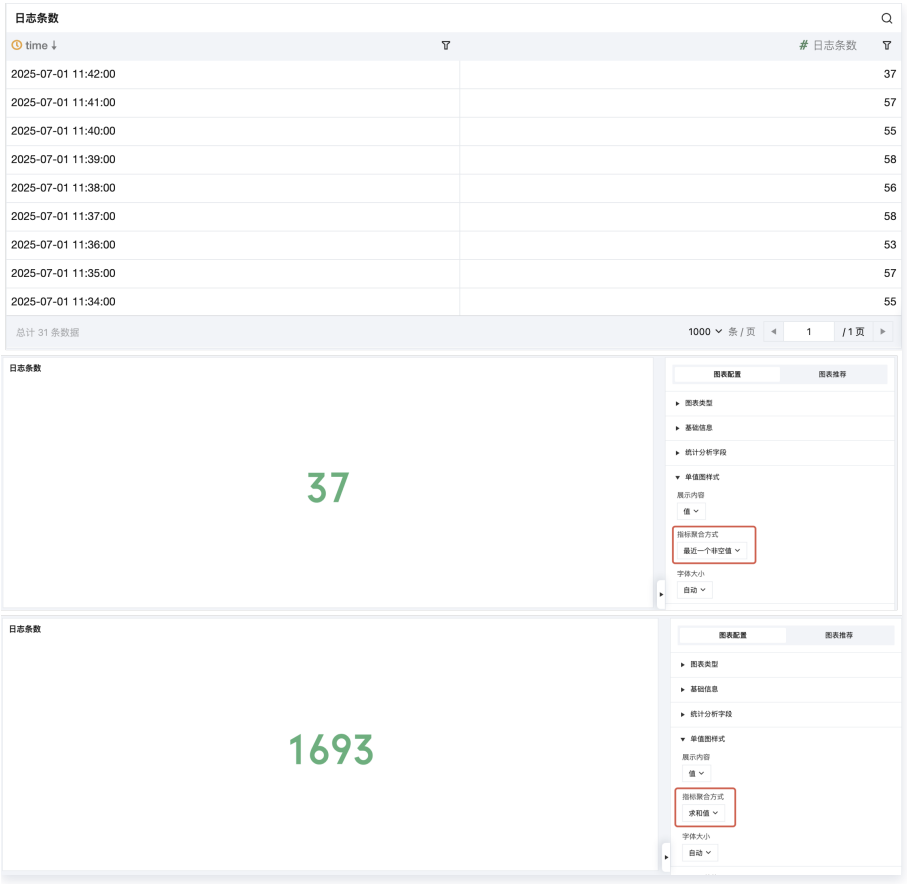


单值图配置

配置项	说明
单值图	• 展示内容：控制是否显示单值图的指标名称。 • 指标聚合方式：当统计结果中指标有多个结果时，单值图需要把结果聚合为一个数值显示或选择其中一个显示。默认使用最近一个非空值。 • 字体大小：配置单值图字体时为固定大小，配置为自动时字体大小根据图表尺寸自动适应。

统计方式示例：

返回多条数据时，默认选择最近一个非空值，即排在末尾的值37显示。若修改为求和值，则会把所有数据结果求和后显示。



阈值配置

配置项	说明
阈值配置	阈值点：设置阈值点，可以添加多个阈值区间。单击阈值对应色彩可打开色盘自定义颜色。



语句示例

统计 Error 日志条数

```
level:error* | select count(*) as "ERROR日志数"
```

计量仪

最近更新时间：2025-07-03 15:06:02

计量仪描述的也是单个指标，但与单值图不同的是，计量仪一般搭配阈值使用，用来衡量该指标的状态。系统健康度监控等有分级标准的场景适用。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘** > **仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**计量仪**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。

统计分析字段

计量仪图至少选择一个**指标字段**，当碰到数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
维度字段	分类字段，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

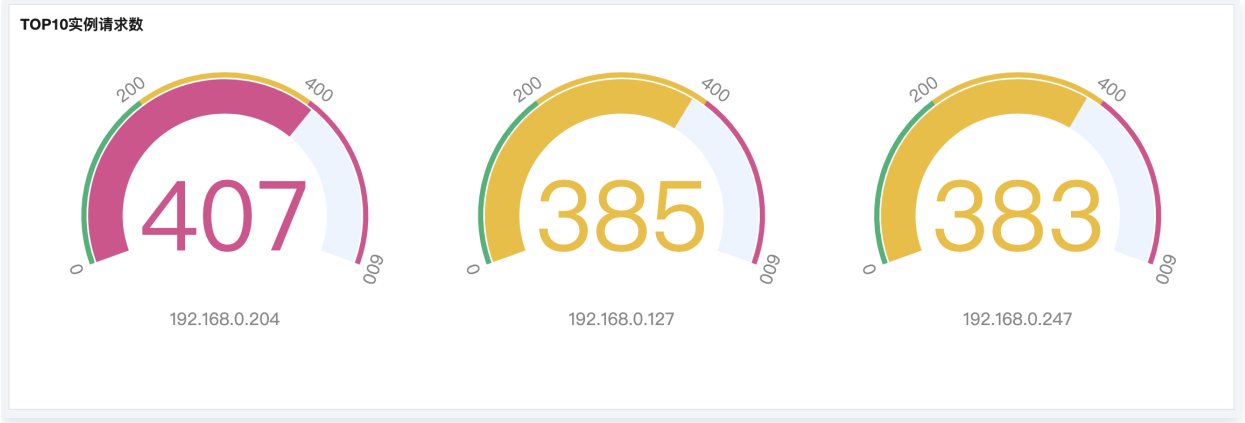
计量仪配置

配置项	说明
计量仪样式	• 显示阈值刻度：在图表上显示配置的阈值刻度。 • 指标聚合：高级功能，选择指标聚合方式，聚合相同指标的结果后再绘图（例如，指标返回了指标的范围值，包含了多个时序点的指标结果，先聚合所有时序点的总和再绘图）。默认不聚合，可设置指标最大的显示数量，超出部分将隐藏。

阈值配置

配置项	说明
阈值配置	• 阈值点：设置阈值点，可以添加多个阈值区间。单击阈值对应色彩可打开色盘自定义颜色。 • 最大值/最小值：控制计量仪扇形区的最大刻度与最小刻度，区间外的数据不会在图表上显示。

阈值配置示例：



语句示例

统计请求失败率

```
* | select round(count_if(try_cast(status as bigint) >= 400)*100.0/count(*),2) as "请求失败率(%)"
```

地图

最近更新时间：2025-07-03 15:06:02

地图通过图形的位置来表现数据的地理位置，通常来展示数据在不同地理区域上的分布情况。攻击 IP 地理分布等地理位置统计场景适用。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**地图**。
3. 按照需求配置图表。

通用配置

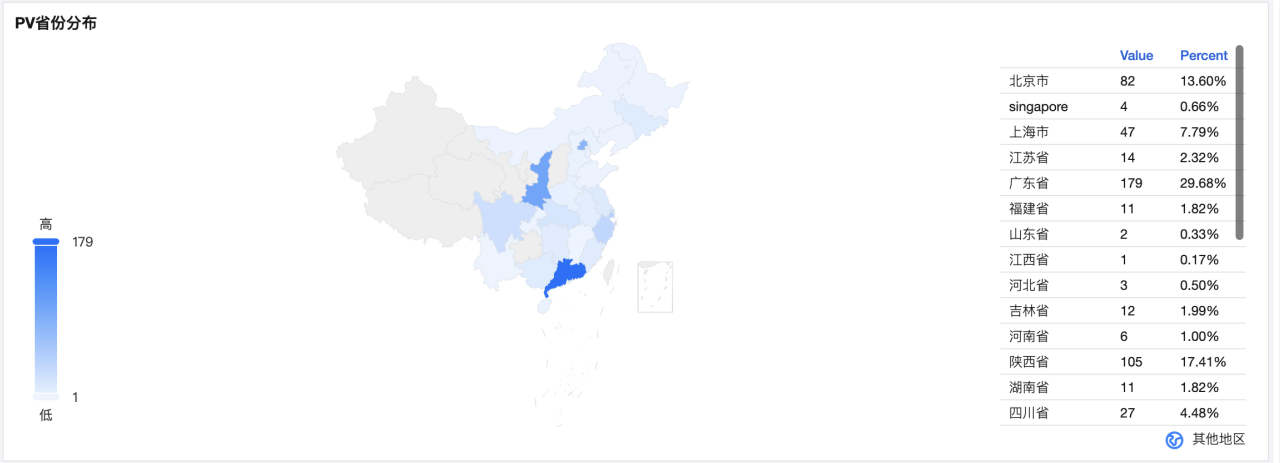
配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
地图样式	地图类型：设置地图的类型，包含中国地图、世界地图、中国省份地图。默认为自动，根据数据包含的区域信息自动适配。
图例	设置图表的图例内容，可以控制图例的样式与位置。同时也支持在图例中添加对比数据。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。

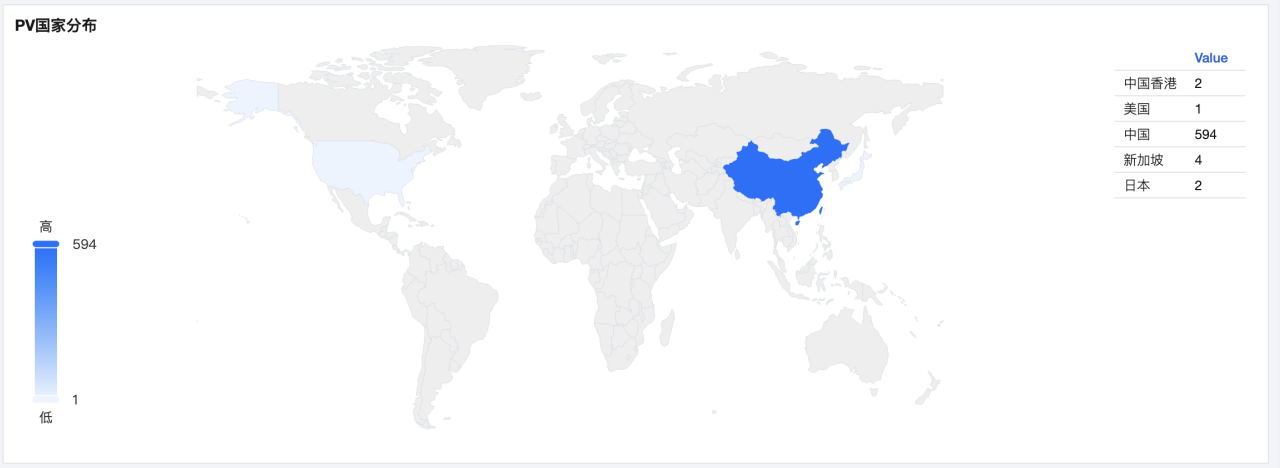
统计分析字段

地图必需维度字段和指标字段，当碰到数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
维度字段	地理信息字段（目前支持国家/省份中文名称、英文名称，国家代码，中国省份代码），默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

地图示例：

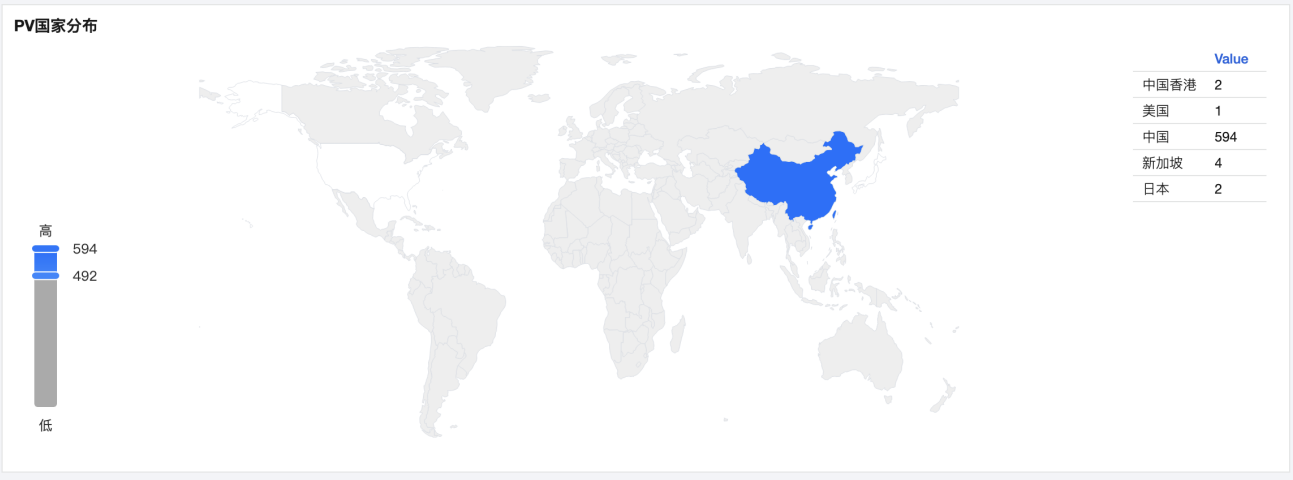




图表操作

根据数值过滤区域

拖拽地图左下角的数值范围，可只显示区间内的区域。



语句示例

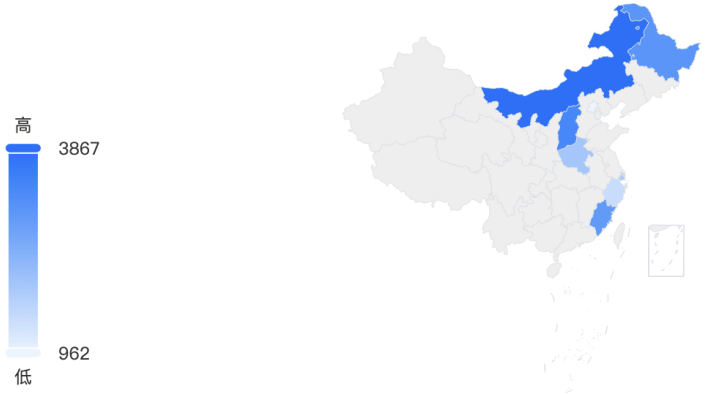
统计访问用户来源的地理分布（中国）

```
* | select ip_to_province(client_ip) as province ,  count(*) as "访问次数" group by province order by "访问次数" desc limit 1000
```

统计访问用户来源的地理分布（世界）

```
* | select ip_to_country(client_ip) as country ,  count(*) as "访问次数" group by country order by "访问次数" desc limit 1000
```

各地域访问次数



	Value
内蒙古自治区	3867
山西省	3326
黑龙江省	2998
福建省	2892
河南省	1944
上海市	1808
浙江省	1452
北京市	962

桑基图

最近更新时间：2025-07-03 15:06:02

桑基图是一种特定类型的流图，用于描述一组值到另一组值的流向。防火墙目的与源 IP 流量统计等场景适用。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘** > **仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**桑基图**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
桑基图样式	桑基图方向：设置桑基图的显示方向，可选择水平或垂直。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。

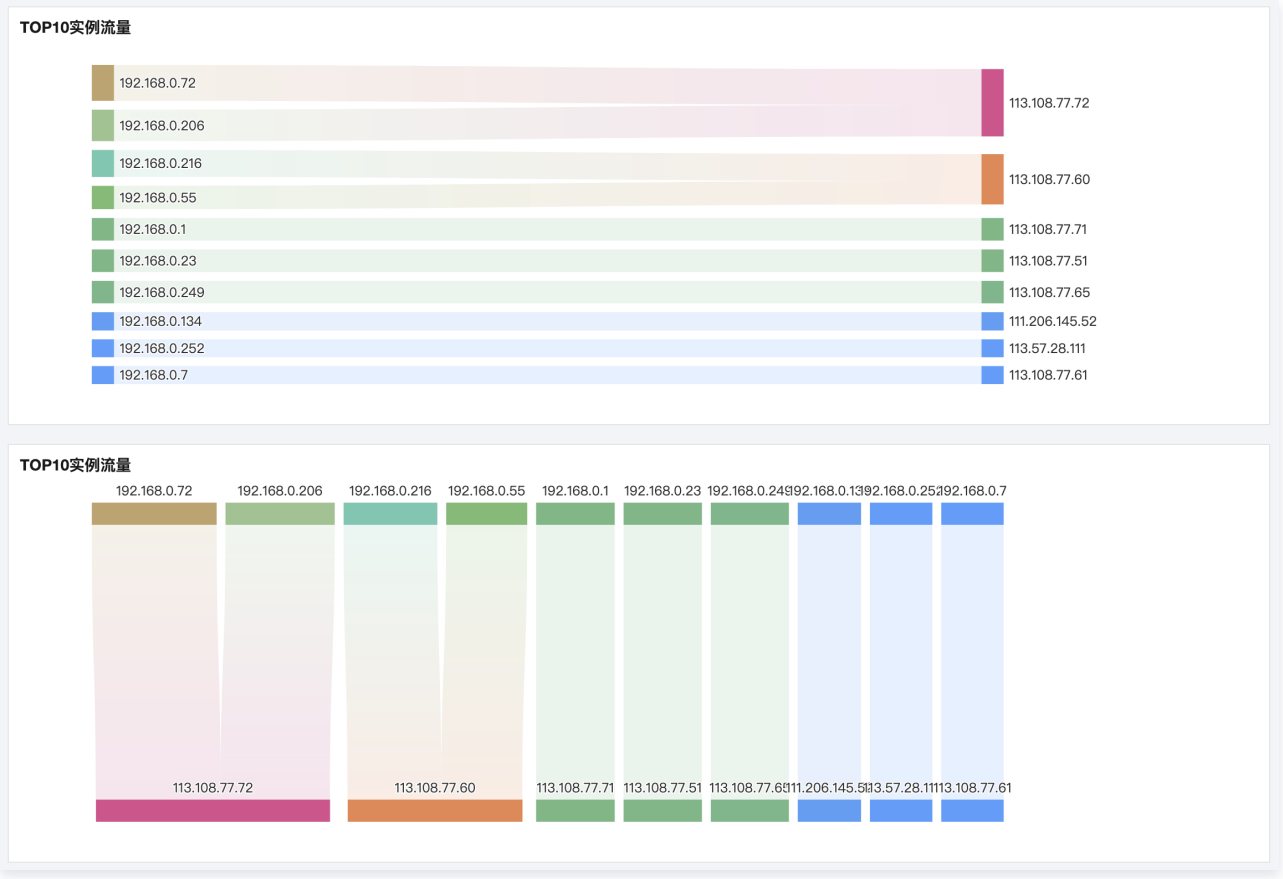
统计分析字段

桑基图必需起点字段、终点字段和指标字段，当碰到数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
起点字段	起点字段，默认选择第一个数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
终点字段	终点字段，默认选择第二个数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

版权所有：腾讯云计算（北京）有限责任公司

桑基图方向示例：



语句示例

统计 TOP 访问用户与目标地址间的流量分布：

```
* | select sum(bytes_sent) as "流量", remote_addr as "来源IP", http_host as "目标地址" group by "来源IP", "目标地址" order by "流量" desc limit 20
```

词云图

最近更新时间：2025-07-03 15:06:02

词云是文本数据的视觉表示，用于展示文本数据的出现频率。在高频操作人员统计等审计场景适用。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**词云**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
词云样式	• 最大显示词条数：控制最大显示多少条数据，取 TOP N，最大取100条。 • 字体大小：控制词云中词条的字体大小范围。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。

统计分析字段

词云图必需词条字段和指标字段，当碰到数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
词条字段	作为词条展示的字段，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

词云图示例：



语句示例

统计访问用户来源的 TOP 省份：

```
* | select ip_to_province(client_ip) as province ,  count(*) as "访问次数" group by province order by "访问次数" desc limit 30
```

漏斗图

最近更新时间：2025-07-03 15:06:02

漏斗图适用于单流向单路径的业务流程，对各环节进行统计并用梯形面积表示某个环节业务量与上一个环节之间的差异。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**漏斗图**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
图例	设置图表的图例内容，可以控制图例的样式与位置。同时也支持配置需要作为图例显示的数据。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。
交互事件	自定义交互事件功能，支持单击图表内容触发交互打开检索分析页面、仪表盘页面、第三方 URL 等。详情请参见 交互事件 。

统计分析字段

漏斗图必须选择**指标字段**，当碰到数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
维度字段	分类字段，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

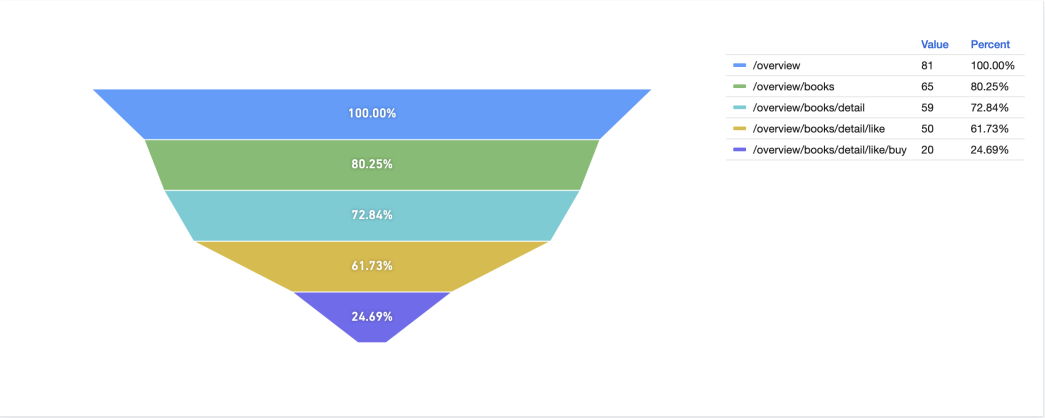
漏斗图配置

配置项	说明
漏斗图样式	• 排序方式：漏斗图数值的排序。默认按照查询结果依次排列。 • 显示内容：设置漏斗图每一层的标签显示形式，可以选择**名称**、**值**或**转化率**。 • 标签：配置漏斗图标签的显示内容。 • 标签位置：配置标签在图表里的位置。 • 转化率：设置转化率的计算方式，支持按占第一层的百分比计算或按占上一层的百分比计算。 • 指标聚合：高级配置，聚合相同指标的结果后再绘图（例如，指标返回了指标的范围值，包含了多个时序点的指标结果，先聚合所所有时序点的总和再绘图）。默认不聚合，可设置指标最大的显示数量，超出部分将隐藏。 • 漏斗图最大显示层数：漏斗图的最大显示层数，超出的部分将会隐藏。

语句示例

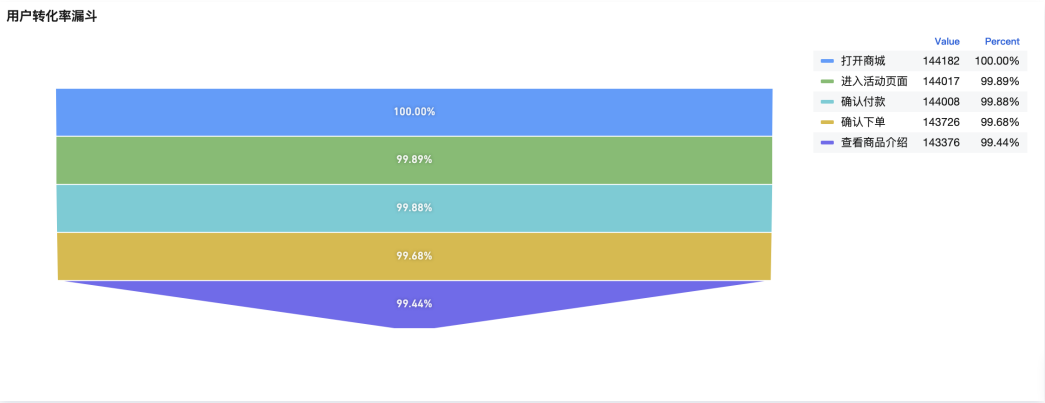
统计用户访问路径各节点的转化率：

```
* | select url, count(*) as pv group by url limit 5
```



统计用户转化率：

```
event:(Advertising OR EnterTheMall OR BrowserTheList OR Order OR Payment) |
 select count(distinct ip) as "用户数", CASE
  WHEN event = 'Advertising' THEN '进入活动页面'
  WHEN event = 'EnterTheMall' THEN '打开商城'
  WHEN event = 'BrowserTheList' THEN '查看商品介绍'
  WHEN event = 'Order' THEN '确认下单'
  WHEN event = 'Payment' THEN '确认付款'
  ELSE 'other' END as event group by event limit 1000
```



统计注册用户的七日留存率：

```
* | select
count(distinct (CASE WHEN count < 1 THEN ip END)) as "第1天留存率",
count(distinct (CASE WHEN count = 1 THEN ip END)) as "第2天留存率",
count(distinct (CASE WHEN count = 2 THEN ip END)) as "第3天留存率",
count(distinct (CASE WHEN count = 3 THEN ip END)) as "第4天留存率",
count(distinct (CASE WHEN count = 4 THEN ip END)) as "第5天留存率",
count(distinct (CASE WHEN count = 5 THEN ip END)) as "第6天留存率",
count(distinct (CASE WHEN count = 6 THEN ip END)) as "第7天留存率"
from
(
select t1-t0 as count,table1.ip from
( select extract( day_of_year from cast(__TIMESTAMP__ as timestamp)) as t0 ,ip where event='SighUp'  group
by t0 ,ip limit 10000 )
as table1
left join
( select extract( day_of_year from cast(__TIMESTAMP__ as timestamp)) as t1 ,ip where event='EnterTheMall'
group by t1 ,ip limit 10000 )
as table2
on table1.ip =  table2.ip
```

```
)
```



版权所有：腾讯云计算（北京）有限责任公司

日志

最近更新时间：2025-07-03 15:06:02

日志类型图表支持将原始日志保存到仪表盘查看，可以在仪表盘页面查看分析结果后，快捷的看到关联的日志内容，无需跳转到检索分析页面。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**日志**。
3. 按照需求配置图表。

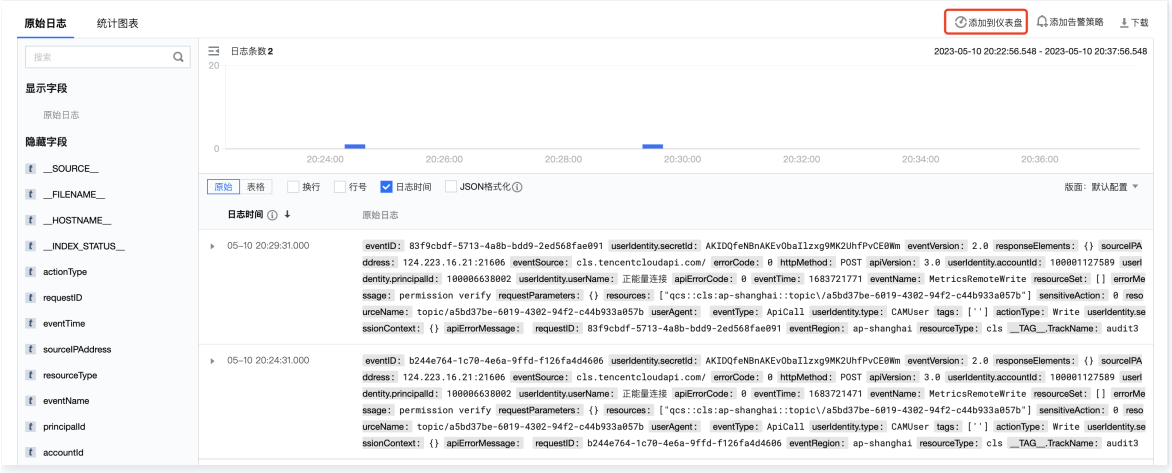
图表配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
日志	• 布局样式：选择原始样式，日志以文本的方式展示。选择表格样式，日志则以表格结构化展示字段与字段值。 • 显示字段：选择需要展示的字段，为空时默认展示所有字段。 • 换行：开启后，原始样式的日志文本将按字段换行，每个字段单独一行。 • 行号：显示日志行号的开关，默认开启。 • 日志时间：显示日志时间的开关，默认开启。

图表操作

添加日志类型图表到仪表盘

方法一：打开检索分析 > 原始日志，单击右上角的添加到仪表盘，将当前浏览的日志添加到仪表盘。



方法二：在仪表盘列表栏选择对应仪表盘，单击添加图表，在仪表盘图表编辑页面选择日志图表类型，并输入检索语句。



加载更多日志

日志图表默认显示前20条日志，滚动到最底部时会自动加载更多日志。



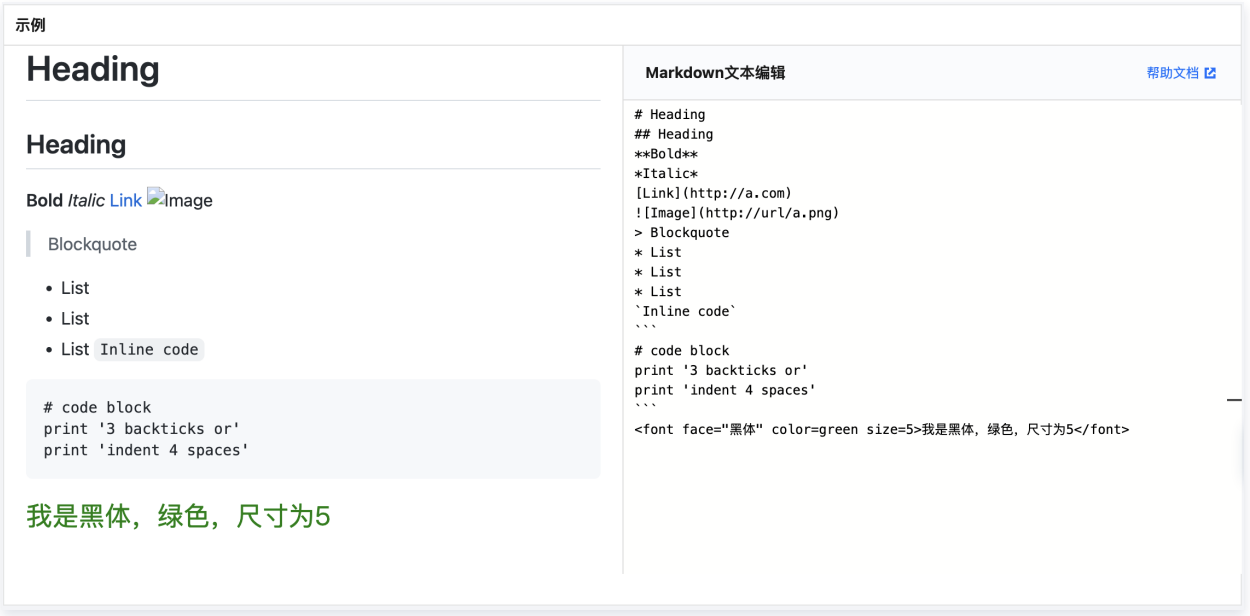
文本

最近更新时间：2025-07-03 15:06:02

文本类型图表支持 MarkDown 语法，可以在仪表盘页面插入文本、图片链接、超链接等。本图表仅支持在仪表盘使用。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**文本**。
3. 按照 MarkDown 语法编辑文本，示例如下：



语法	说明
`# Heading`	一级标题
`## Heading`	二级标题
`**Bold**`	字体加粗
`*Italic*`	字体倾斜
`[Link](http://a.com)`	超链接
`![Image](http://url/a.png)`	图片链接
`> Blockquote`	引用
`* List` `* List` `* List`	列表

`Horizontal rule:`
 	
`---`
	分割线

`` `Inline code` ``	行内代码块

``` # code block print '3 backticks or' print 'indent 4 spaces' ```	代码块
`<font face="黑体" color=green size=5>我是黑体，绿色，尺寸为5</font>`	带样式的文字

热力图

最近更新时间：2025-07-03 15:06:02

热力图支持通过对色块着色来显示数据的统计图表。对于统计的指标，较大的值由较深的颜色表示，较小的值由较浅的颜色表示。热力图适用于查看总体的情况、发现异常值、显示多个变量之间的差异，以及检测它们之间是否存在任何相关性的场景。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**热力图**。
3. 按照需求配置图表。

图表配置

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
坐标轴	• 显示 X 轴：显示/隐藏 X 轴 • Y 轴位置：配置 Y 轴的显示位置
图例	显示/隐藏图例配置。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。
交互事件	支持单击图表中的数值触发交互事件，如跳转到设定的 URL 地址、打开检索分析页面、打开仪表盘页面、添加过滤条件等。详情请参见 交互事件 。

统计分析字段

热力图必需指标字段，当碰到数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
X 轴字段	选择热力图 X 轴对应的字段，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
Y 轴字段	选择热力图 Y 轴对应的字段，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段
指标字段	数值字段，热力图色块对应的字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

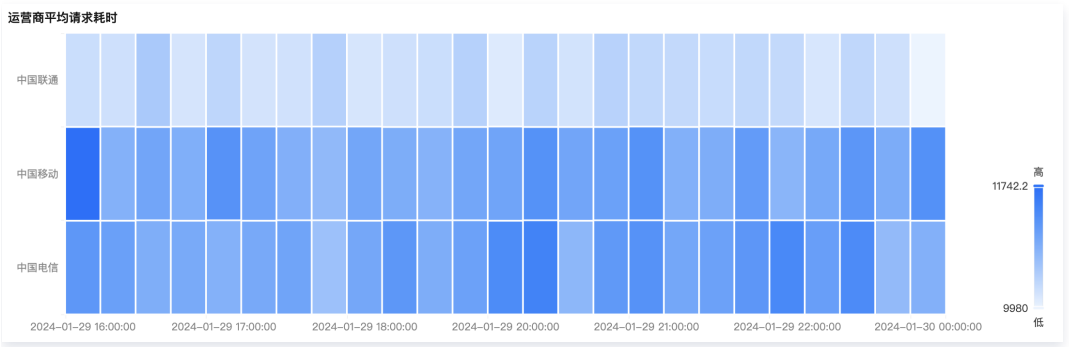
热力图配置

配置项	说明
热力图样式	显示数值：设置热力图色块数值标签的显示和隐藏。默认隐藏数值。

语句示例

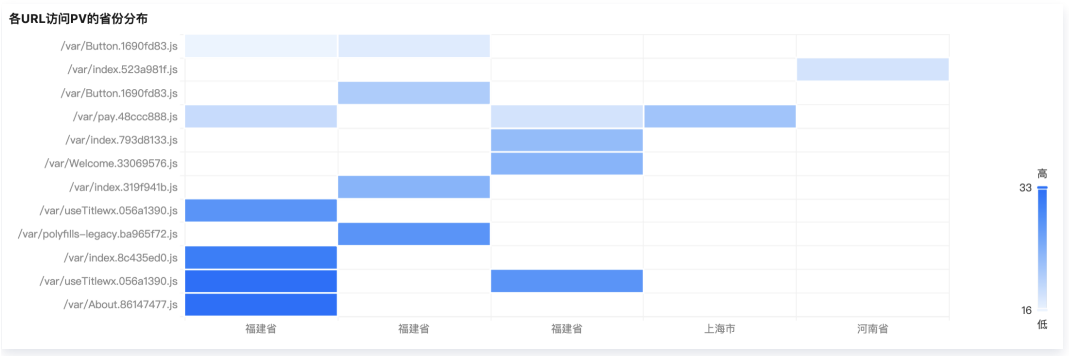
统计各运营商在时间轴上的平均请求延迟：

```
* | select histogram(__TIMESTAMP__,interval 1 hour) as time,isp as "运营商",round(avg( "request_time" ),1)
as "平均请求耗时" group by time,"运营商" order by time limit 10000
```



统计各 URL 访问 PV 的省份分布：

```
* | select  url,prov, count(*) as count group by url,prov order by count desc limit 1000
```



多语句绘图示例

最近更新时间：2025-07-03 15:06:02

概览

仪表盘统计图表支持渲染来自多个主题多个语句的数据，支持跨主题的数据关联和计算。满足多语句多主题数据对比展示，跨主题数据关联分析的场景。

图表类型	多语句绘图	默认展示逻辑
表格	支持	将多个语句的结果合并展示
时序图	支持	将多个语句的结果合并展示
饼图	支持	将多个语句的结果合并展示
计量仪	支持	将多个语句的结果合并展示
单值图	支持	从多个语句的结果中选择一个指标并展示
漏斗图	支持	将多个语句的结果合并展示
柱状图	支持，依赖语句连接（JOIN）	展示语句连接（JOIN）后的结果，如果没有配置语句连接（JOIN），则仅展示第一个语句的结果
桑基图	支持，依赖语句连接（JOIN）	展示语句连接（JOIN）后的结果，如果没有配置语句连接（JOIN），则仅展示第一个语句的结果
地图	支持，依赖语句连接（JOIN）	展示语句连接（JOIN）后的结果，如果没有配置语句连接（JOIN），则仅展示第一个语句的结果
词云	支持，依赖语句连接（JOIN）	展示语句连接（JOIN）后的结果，如果没有配置语句连接（JOIN），则仅展示第一个语句的结果
雷达图	支持，依赖语句连接（JOIN）	展示语句连接（JOIN）后的结果，如果没有配置语句连接（JOIN），则仅展示第一个语句的结果
热力图	支持，依赖语句连接（JOIN）	展示语句连接（JOIN）后的结果，如果没有配置语句连接（JOIN），则仅展示第一个语句的结果
拓扑图	支持，依赖语句连接（JOIN）	展示语句连接（JOIN）后的结果，如果没有配置语句连接（JOIN），则仅展示第一个语句的结果
日志	暂不支持	/

示例

当前有 A、B 两种业务服务。两个业务的访问日志分别保存在 A、B 两个日志主题。存在统计分析需求，详见以下内容。

统计并对比 A、B 两个业务的 PV 趋势

步骤1：在仪表盘 添加图表，选择时序图图表类型，添加 A、B 两个日志主题的查询语句，分别输入 PV 趋势的 SQL 查询语句。

- 业务 A 的语句：

```
* | select histogram(__TIMESTAMP__,interval 1 minute) as time, count(*) as "PV" group by time order by
time limit 10000
```

- 业务 B 的语句：

```
* | select histogram(__TIMESTAMP__,interval 1 minute) as time, count(*) as "PV" group by time order by
time limit 10000
```

版权所有：腾讯云计算（北京）有限责任公司



统计 A、B 两个业务的 PV，在总 PV 中的分布

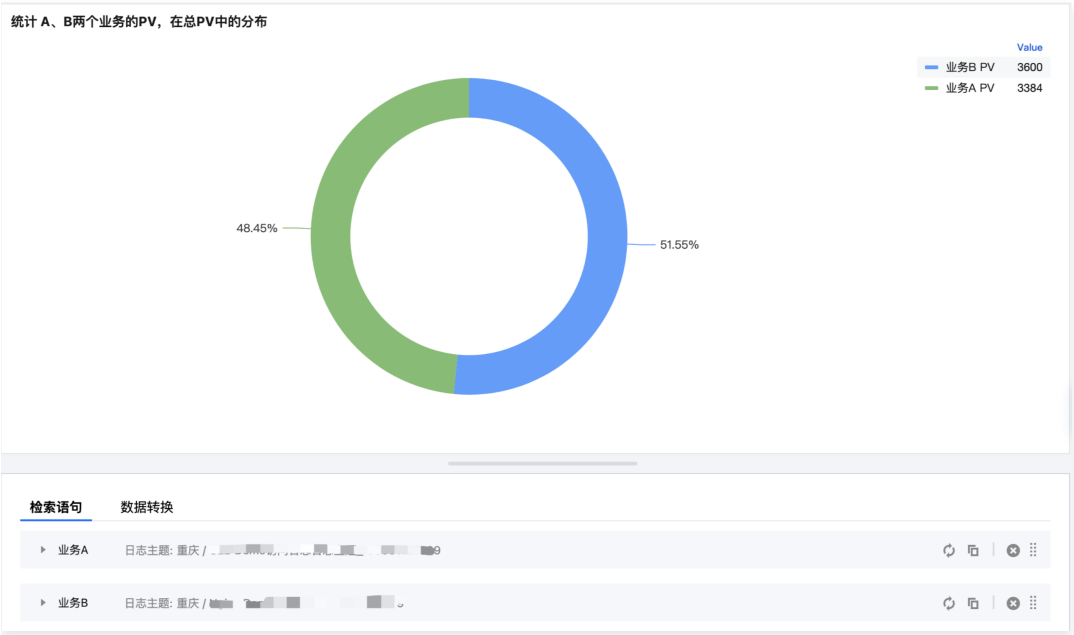
步骤1：在仪表盘 添加图表，选择饼图图表类型，添加 A、B 两个日志主题的查询语句，分别输入 PV 的查询语句。

- 业务 A 的语句：

```
* | select count(*) as "PV"
```

- 业务 B 的语句：

```
* | select count(*) as "PV"
```



统计业务 A 和业务 B 的 PV 总和

步骤1：在仪表盘 添加图表，选择单值图图表类型，添加 A、B 两个日志主题的查询语句，分别输入 PV 的查询语句，得到两个业务的 PV。

- 业务 A 的语句：

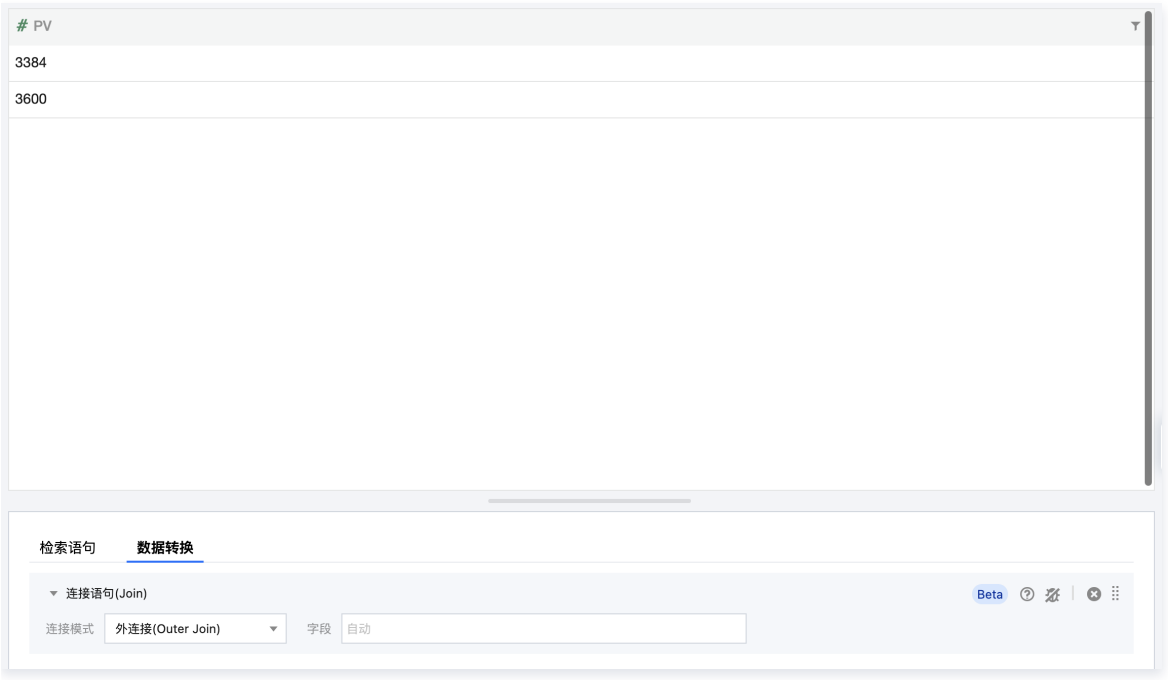
```
* | select count(*) as "PV"
```

- 业务 B 的语句：

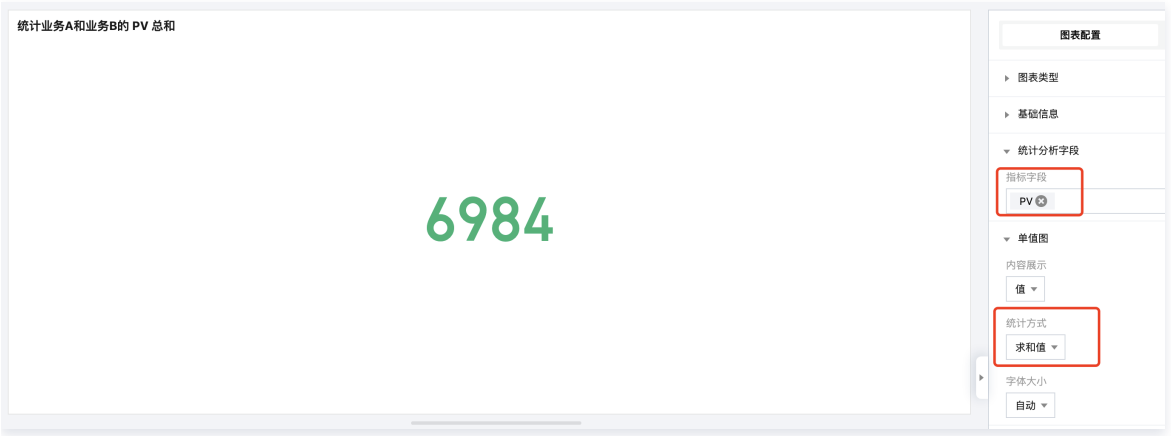
```
* | select count(*) as "PV"
```

步骤2：添加数据转换--连接语句（JOIN），选择外连接，合并两个语句的业务 PV 在一张表里。

> **注意：**
> 此处基于同名字段做连接，需保证两个指标同名。



步骤3：单值图选择统计指标 PV，统计方式为对 PV 求和，得到两个业务的 PV 总和。



按照 server_name 统计所有业务的请求错误数分布

只有 A 业务主题有 server_name 字段，B 业务没有该字段，此外 A、B 业务均有服务端的 IP 字段可以关联 server_name。

步骤1：在仪表盘 添加图表，选择表格图表类型，添加 A、B 两个日志主题的查询语句。

- 业务 A 的语句，查询 server_name、服务端 IP 的请求错误数分布。

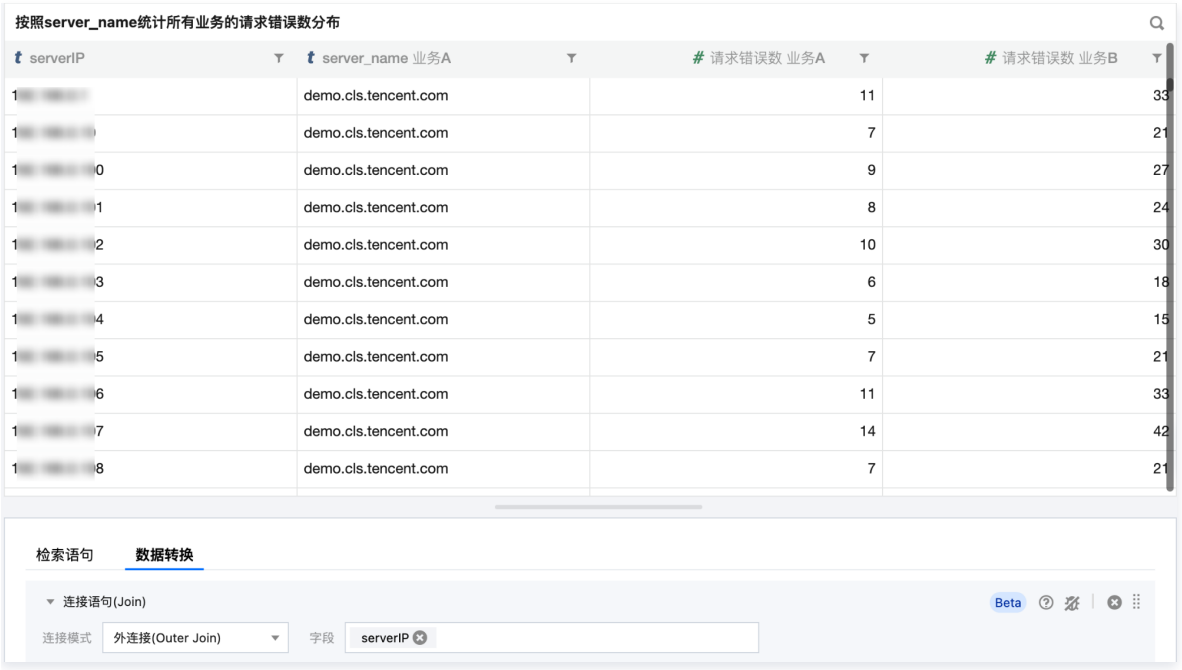
```
status>=400 | select server_name,server_addr as "serverIP",count(*) as "请求错误数"  group by
server_name,server_addr limit 10000
```

- 业务 B 的语句，查询服务端 IP 的请求错误数分布。

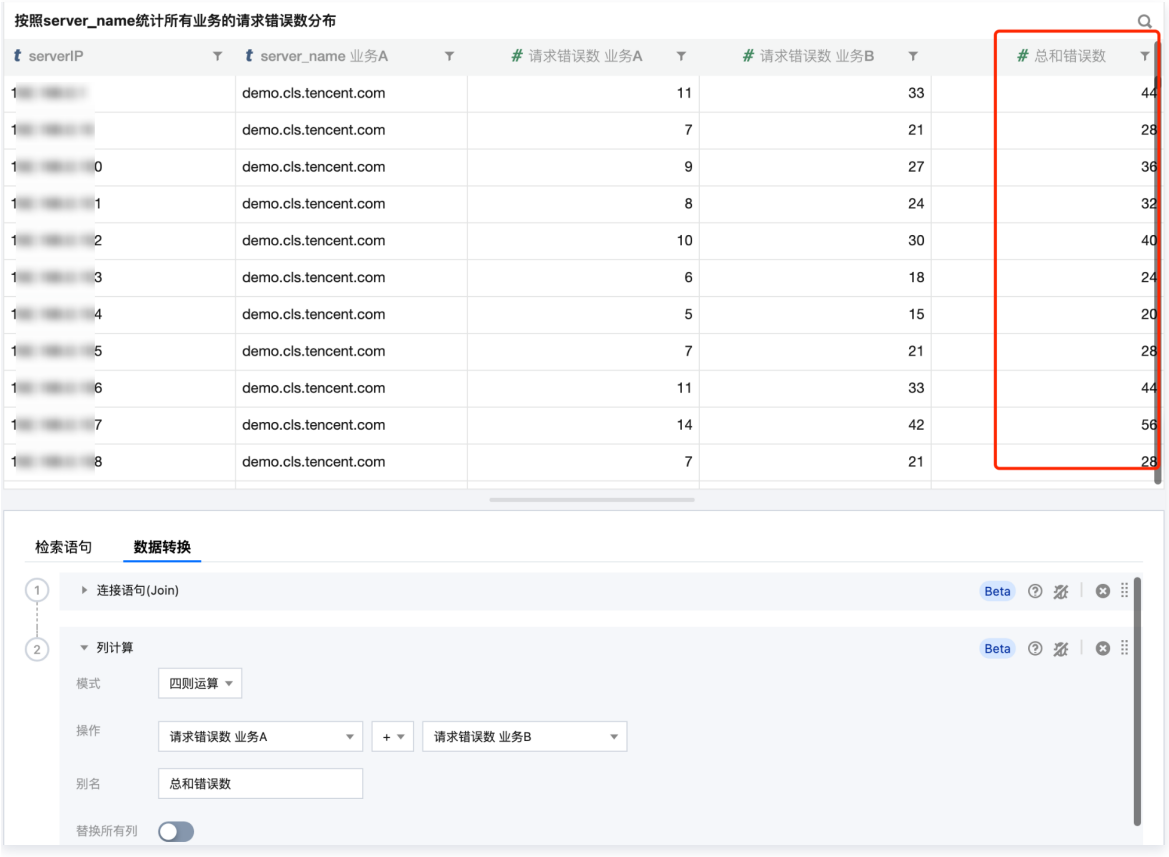
```
status>=400 | select sys_address as "serverIP",count(*) as "请求错误数"  group by  sys_address limit 10000
```

步骤2：添加数据转换--连接语句（JOIN），选择外连接，合并两个语句在一张表里。

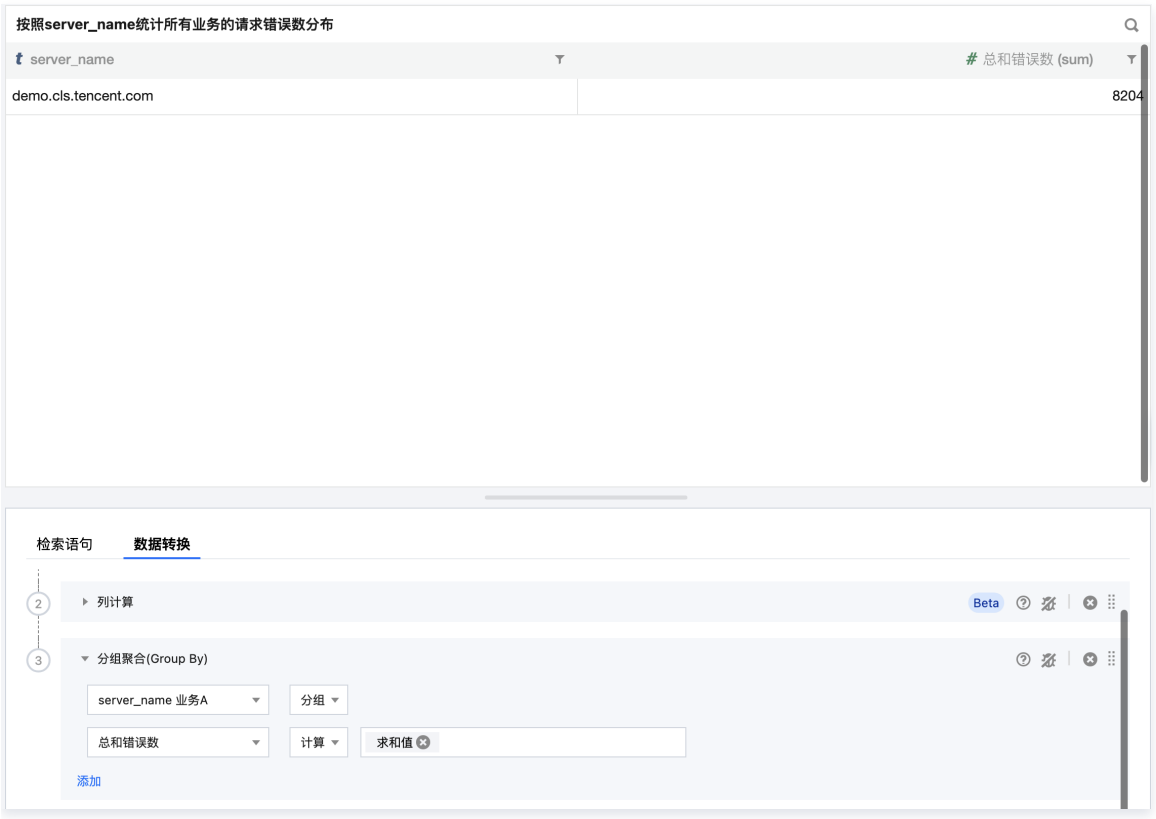
> **注意：**
> 此处基于同名字段 serverIP 做连接。



步骤3：添加数据转换--列计算，计算每个 serverIP 上业务 A、业务 B 的请求错误总和。



步骤4：添加数据转换-分组聚合，重新按照 server_name 分组，计算所有业务的错误数在 server_name 上的分布。



雷达图

最近更新时间：2025-04-29 19:18:42

雷达图是一种在多个变量间进行可视化和比较的图表。它通过从一个中心点出发的多条轴线，代表不同的数据类别或指标，然后在每条轴线上用点来表示数据值，最后将这些点连接起来形成一个多边形，从而展示数据之间的关系和趋势。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**雷达图**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
图例	设置图表的图例内容，可以控制图例的样式与位置。同时也支持在图例中添加对比数据。
悬浮提示	设置鼠标悬浮触发的气泡提示的内容样式。
标准配置	设置图表内所有指标类型的字段单位。详情请参见 单位配置 。
交互事件	支持单击图表中的数值触发交互事件，如跳转到设定的 URL 地址、打开检索分析页面、打开仪表盘页面、添加过滤条件等。详情请参见 交互事件 。

统计分析字段

雷达图必需选择**维度字段**和**指标字段**，当出现数据不匹配的情况时，需手动选择统计分析字段。

配置项	说明
维度字段	分类字段，每条轴对应一个字段值，默认选择数据类型为 string 的字段。当没有 string 类型字段时，需手动选择指定字段。
指标字段	数值字段，默认选择数据类型为 number 的字段。当没有 number 类型字段时，需手动选择指定字段。

雷达图配置

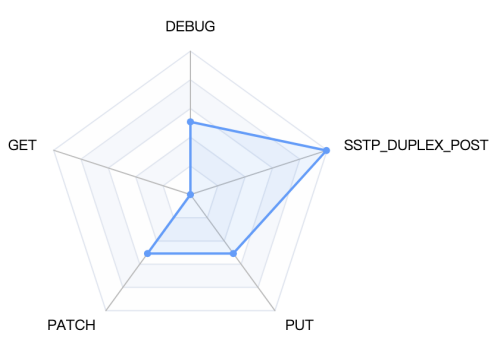
配置项	说明
雷达图样式	• 雷达图类型：设置雷达图的展示类型 • 显示指标数值：控制指标数值标签的显示/隐藏 • 显示维度标签：控制维度数值标签的显示/隐藏 • 最大维度显示数量：控制显示维度的数量 • 线宽：控制线条的粗细。 • 不透明度：控制填充区域的不透明度。 • 显示点：显示数据点，无数据则不显示 • 点的大小：控制线条上数据点的大小，为0时不显示。

语句示例

统计各请求类型的请求延迟对比：

```
* | select request_method, round(avg(request_time)*1000, 2) as "平均访问延时" group by request_method  limit
100
```

请求类型延时对比



拓扑图

最近更新时间：2025-05-06 10:05:12

拓扑图是一种直观展示节点间关系的图表类型，在日志分析场景中可用于分析服务间的调用关系、系统架构情况以及数据流的方向等。

操作步骤

1. 登录 日志服务控制台，选择**仪表盘 > 仪表盘列表**，选择您想查看的仪表盘。
2. 单击 添加图表，新建一个图表，图表类型选择**拓扑图**。
3. 按照需求配置图表。

通用配置

配置项	说明
基础信息	图表名称：设置图表的显示名称，可为空。
节点配置	• 节点 ID：节点展示的节点 ID 字段，从源节点 ID 或目标节点 ID 中选择。 • 节点显示名称：节点展示的节点名称，为空时显示节点 ID。 • 节点内展示的字段：显示在节点上的字段信息，仅支持选择一个字段。例如节点的某个重要监控指标，或重要属性。 • 节点提示框展示的字段：单击节点触发的提示窗内展示的信息，支持选择多个字段内容。 • 显示节点图标：在节点上显示图标，图表类型通过字段值映射。图标会覆盖节点内展示的字段信息。 • 节点阈值：选择节点的某个指标作为阈值，满足阈值条件时，节点变更为对应色彩，可用于节点健康状态监控等场景。
连线配置	• 连线上显示的内容：显示在连线上的字段信息，仅支持选择一个字段。例如两个服务间的请求平均耗时等。 • 连线提示框展示的字段：单击连线触发的提示窗内展示的信息，支持选择多个字段内容。 • 连线线宽：连线线宽映射的指标，例如两个节点间的流量大小，线宽越宽则流量越大。 • 连线阈值：选择连线的某个指标作为阈值，满足阈值条件时，连线变更为对应色彩，可用于服务间请求耗时监控等场景。
布局配置	• 布局类型：选择默认的布局类型。 • 显示节点名称：控制是否显示节点名称。 • 显示缩略图：控制是否显示缩略图。
标准配置	• 单位：设置图表内所有指标类型的字段单位。详情请参见 单位配置 。 • 精度：设置指标类型的字段的小数保留位数。
交互事件	支持单击图表中的数值触发交互事件，如跳转到设定的 URL 地址、打开检索分析页面、打开仪表盘页面、添加过滤条件等。详情请参见 交互事件 。

统计分析字段

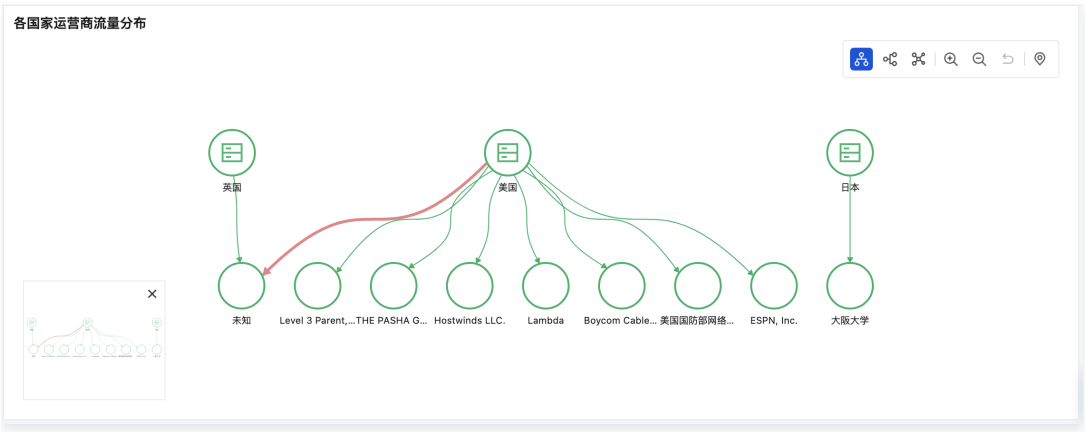
拓扑图必须选择**源节点 ID** 字段和**目标节点 ID** 字段，需手动选择统计分析字段。

配置项	说明
源节点 ID	选择拓扑图源节点 ID 的字段，一对源目节点构成一组拓扑关系。
目标节点 ID	选择拓扑图目标节点 ID 的字段，一对源目节点构成一组拓扑关系。

语句示例

统计各国家运营商流量分布。

```
log-status : OK | select ip_to_country(srcaddr) as src_addr_country,ip_to_provider(srcaddr) as
src_addr_provider, sum(bytes) as total_bytes group by src_addr_provider,src_addr_country order by
total_bytes desc limit 10
```


各国家运营商流量分布
英国
美国
日本
未知
Level 3 Parent,...
THE PASHA G...
Hostwinds LLC.
Lambda
Boycom Cable...
美国国防部网络...
ESPN, Inc.
大阪大学

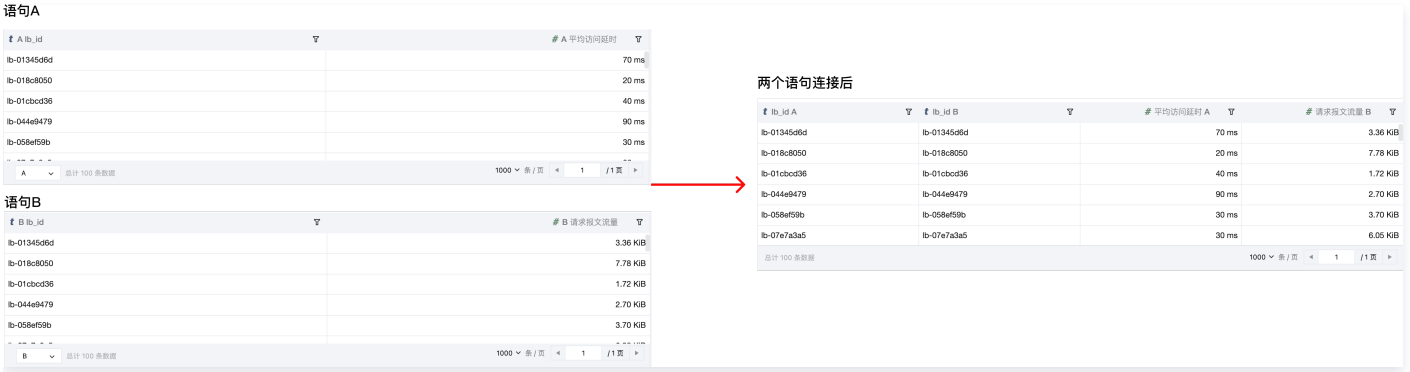
数据转换

最近更新时间：2025-05-06 10:57:22

数据转换支持对统计分析的结果进行二次处理，包括字段类型修改、显示/隐藏字段、多个语句结果关联、列计算等。使用多个语句绘图时，可以通过数据转换完成跨主题的数据聚合。

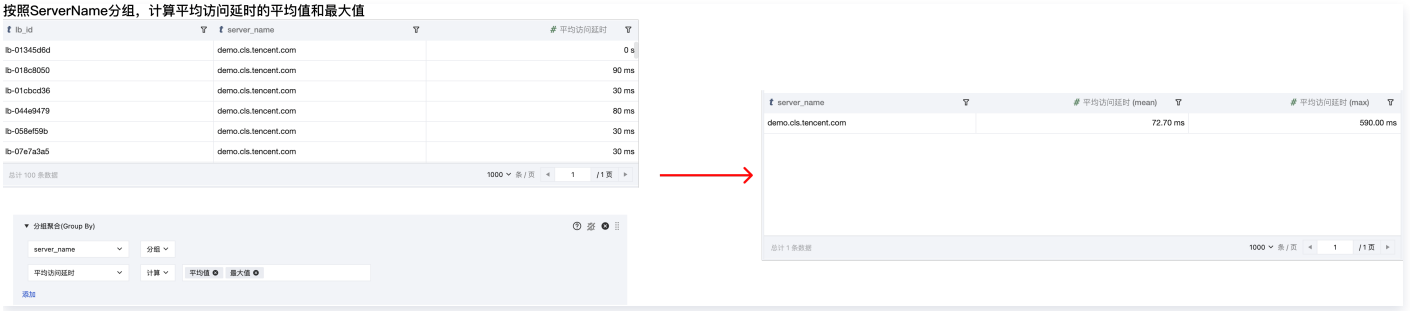
连接语句（Join）

支持将多个语句多个主题的结果合并为一个结果。例如，根据 lb_id 字段连接两个查询结果，合并请求报文流量、平均访问延时到一个结果中。



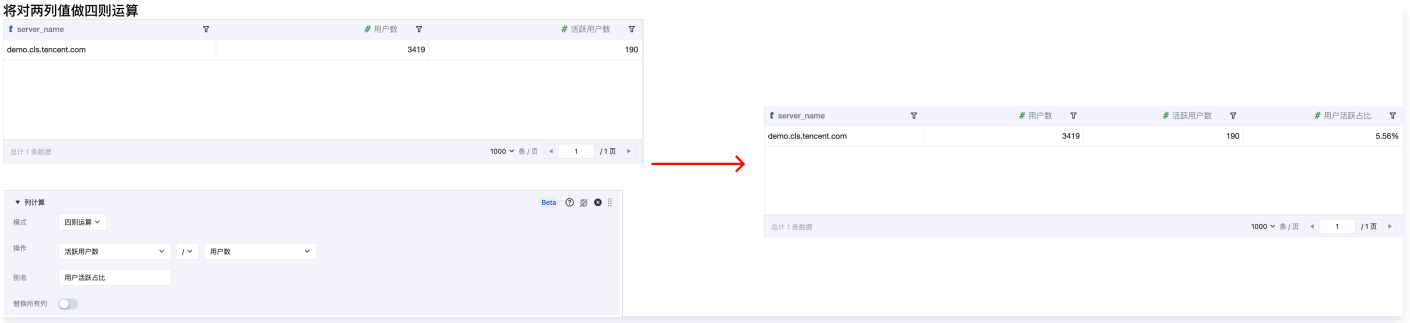
分组聚合（Group By）

根据指定的字段对查询分析结果进行分组合并，并对合并后的组做聚合计算。例如，根据 server_name 字段分组，忽略 lb_id 字段，并对各组平均访问延时求平均值与最大值。



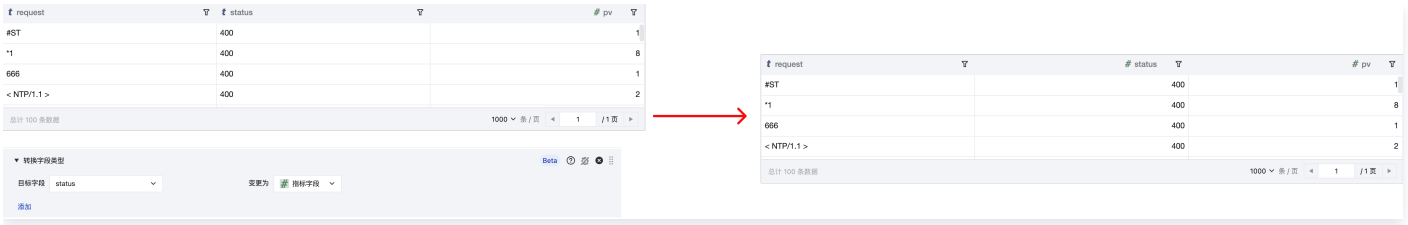
列计算

通过列计算，实现两个主题语句合并后的四则运算。例如根据 server_name 合并 A、B 两个主题的语句，得到总用户数和活跃用户数。然后通过列计算，使用活跃用户数/总用户数 求出用户活跃率。



转换字段类型

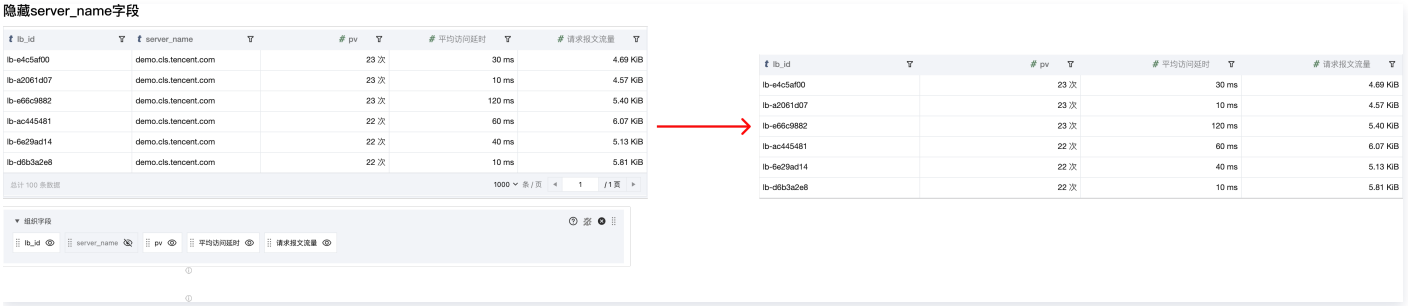
修改字段类型，例如将 status 字段由 number 类型转换为 string 类型。



组织字段

支持字段显示/隐藏，字段排序。例如隐藏 server_name 字段。

隐藏server_name字段



单位配置

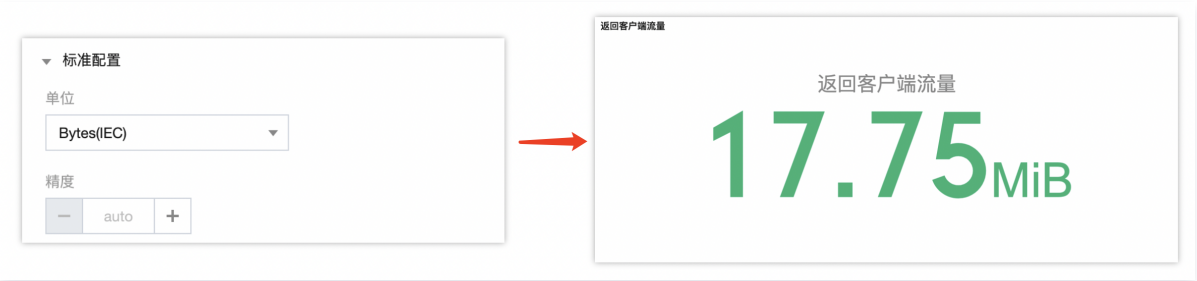
最近更新时间：2024-05-29 11:46:11

使用 SQL 聚合的结果默认是没有单位的数值。通过单位配置，可以在图表中显示单位。单位支持自动换算，选择一个原始单位后，当数值满足换算进率时，自动换算为更高一级别的单位。单位支持精度配置。

选择一个单位并自动换算

在日志服务（Cloud Log Service，CLS）图表分析中，SQL 聚合结果默认是没有单位的原始数值，需要用户指定一个基本单位，如例中的数值单位 byte。用户选择一个初始单位后，当数值满足更高级单位的进率，就会自动换算。如例中所示，单位从 bytes 自动换算 MiB。单位配置里的精度影响小数位数，默认 Auto 保留2位小数，通过修改可以控制保留的小数位数。





手动输入单位

在使用中，如果数据具有特殊单位，而单位配置选项里没有时。用户可以通过手动输入，自定义添加一个单位（带 custom 前缀）。

> **注意：**
> 自定义的单位是固定不变的，不会自动换算。因此，若不希望触发单位自动换算，统一使用原始单位，可以通过手动输入添加一个固定单位。



图表时间配置

最近更新时间：2024-10-18 10:20:11

操作场景

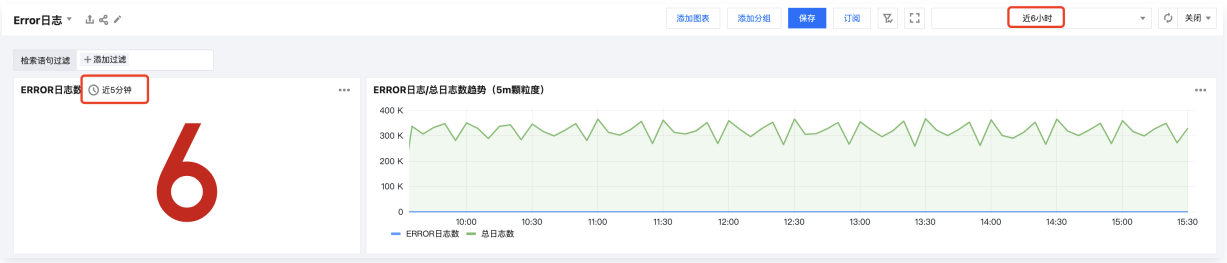
仪表盘图表支持独立的图表时间配置。与图表时间相对的是仪表盘时间。仪表盘图表创建时，默认使用仪表盘时间，其数据范围受仪表盘时间控制。修改仪表盘时间，所有关联图表的时间范围都会发生改变。而关闭使用仪表盘时间后，图表时间独立控制，不再随全局时间的变化而变化，仪表盘的图表间可以有不同的时间范围，支持更丰富的对比场景。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中单击**查看仪表盘**，进入仪表盘页面。
3. 单击编辑图表或添加图表，选择自定义图表，进入图表编辑页面。
4. 图表默认使用仪表盘全局时间，关闭**使用仪表盘时间**的开关，将支持独立的图表时间。



5. 查看图表，图表时间与仪表盘时间不同。



过滤与变量

最近更新时间：2025-07-03 15:06:02

类型说明

类型	说明	生效范围
下拉选项过滤	通过指定字段的字段值对仪表盘内所有图表的数据进行过滤。过滤字段在日志主题索引配置内开启统计时，可自动获取字段的值作为列表项。	仪表盘内所有日志主题的图表
检索语句过滤	通过输入检索语句对仪表盘内所有图表的数据进行过滤，相当于在各图表查询语句中增加了过滤条件。检索语句过滤支持 范围过滤、NOT 过滤、全文检索等更丰富的过滤方式。	仪表盘内所有日志主题的图表
数据源变量	数据源变量支持批量切换仪表盘内图表的数据源，适用于一个仪表盘应用到多个日志主题、仪表盘内数据进行蓝绿对比等场景。	仪表盘内使用该变量的图表
自定义变量	自定义变量支持用户自己设置静态输入或动态查询的变量值，替换到检索语句、标题、文本图表中，实现快捷的语句批量修改。	仪表盘内使用该变量的图表
系统预置变量	系统预置变量，无需配置即可直接在语句中引用，自动替换语句中对应的字段。在需要自适应时间颗粒度、自动获取当前时间范围的场景适用。	仪表盘内使用该变量的图表

过滤

下拉选项过滤配置

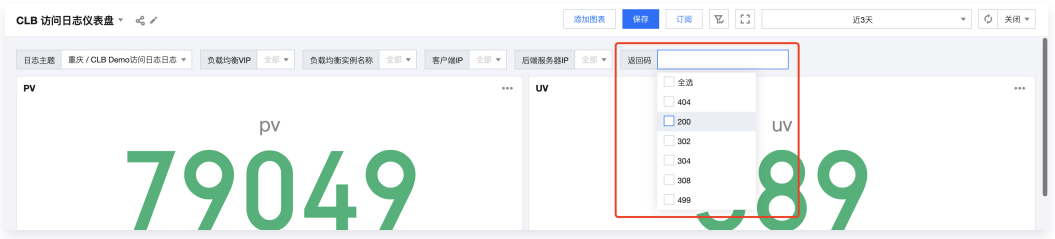
1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**仪表盘** > **仪表盘列表**，进入仪表盘管理页面。
3. 单击需要操作的仪表盘 ID/名称，进入该仪表盘详情页面。
4. 单击顶部的过滤与变量图标，选择**新增过滤与变量**，打开设置页面。



5. 在弹出的窗口中，设置下拉选项过滤配置，单击**确定**。

表单元素	说明
类型	不同的类别对应不同的配置项与应用场景，此处选择**下拉选项过滤**。
过滤器别名	界面上过滤器的显示名称，非必填项目，空置时自动使用过滤字段为显示名称。
日志主题	过滤字段来源的日志主题。
过滤字段	过滤的对象字段。
动态选项	开启后，自动获取过滤字段的值作为过滤选项。
上级级联过滤	可选择当前过滤字段级联的上一级过滤字段，配置后只会显示符合上一级过滤条件的过滤项。常用如：国家、省份等具有级联关系的字段。
静态选项	手动添加过滤项，静态过滤项将常驻显示，支持配置别名，非必填项目。
默认过滤	用于配置查看仪表盘时的默认过滤项，非必填项目。
支持多选	开启后可以选择多个过滤项作为过滤条件。

6. 返回仪表盘详情页面，单击**过滤器**，选择过滤项，仪表盘数据将刷新为过滤后内容。



检索语句过滤配置

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**仪表盘 > 仪表盘列表**，进入仪表盘管理页面。
3. 单击需要操作的仪表盘 ID/名称，进入该仪表盘详情页面。
4. 单击顶部的过滤与变量图标，选择**新增过滤与变量**，打开设置页面。

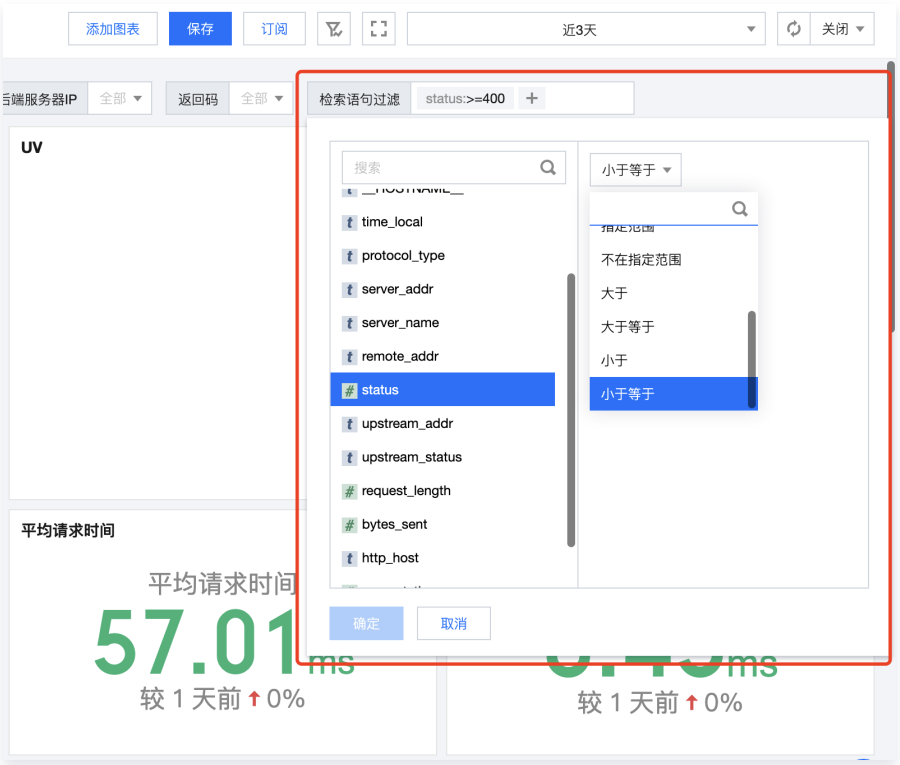


5. 在弹出的窗口中，设置检索语句过滤信息，单击**确定**。

表单元素	说明
类型	不同的类别对应不同的配置项与应用场景，此处选择**检索语句过滤**。
过滤器名称	过滤器的名称，名称唯一，不能重复。
过滤器别名	界面上过滤器的显示名称，非必填项目。
日志主题	过滤字段来源的日志主题。
模式	检索语句的输入模式，支持**交互模式**与**语句模式**两种。
默认过滤	用于配置查看仪表盘时的默认过滤项，非必填项目。

6. 返回仪表盘详情页面，单击**过滤器**，选择过滤项，仪表盘数据将刷新为过滤后内容。
   检索语句过滤支持：范围过滤、NOT 过滤、全文检索等更丰富的过滤方式。

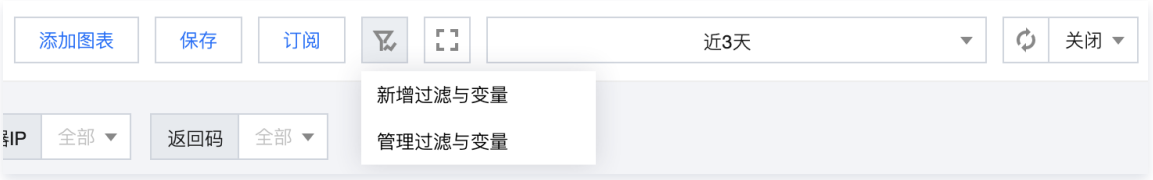
版权所有：腾讯云计算（北京）有限责任公司



变量

数据源变量配置

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**仪表盘** > **仪表盘列表**，进入仪表盘管理页面。
3. 单击需要操作的仪表盘 ID/名称，进入该仪表盘详情页面。
4. 单击顶部的过滤与变量图标，选择**新增过滤与变量**，打开设置页面。



5. 在弹出的窗口中，设置模板变量信息，单击**确定**。

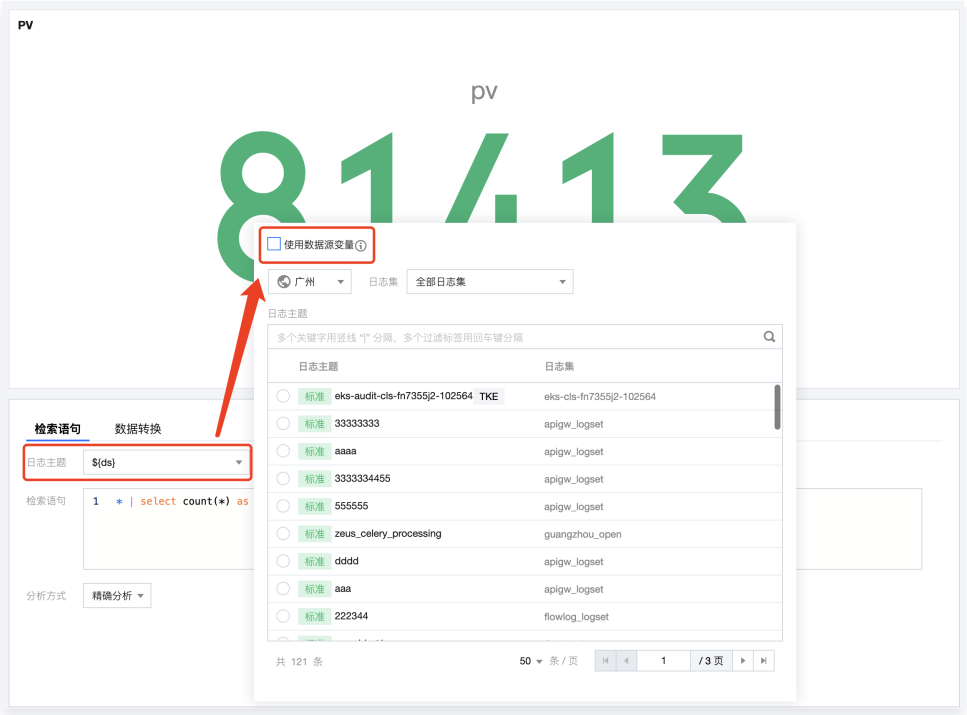
表单元素	说明
变量类型	变量的类别，不同的类别对应不同的配置项与应用场景，此处选择**数据源变量**。
变量名称	查询检索语句中变量的命名，仅支持字母、数字和_。
变量别名	仪表盘上变量的显示名称，非必填项目，为空时自动使用变量名称为显示名称。
主题类型	数据源选项的的主题类型，可选择**日志主题**或**指标主题**。
数据源范围	变量值的可选范围，默认为**所有日志主题**。可选择**自定义过滤**，设置过滤条件，只查看符合条件的日志主题。
默认日志主题	默认使用的日志主题。

6. 返回仪表盘详情页面，单击图表右上角的 ··· > **编辑**。

> **说明：**

> 如果您的仪表盘中没有图表，请 添加图表 。

7. 在编辑图表**检索语句**内，单击**日志主题**，勾选**使用数据源变量**，选择刚新建的模板变量。



8. 返回仪表盘详情页面，单击上方的数据源变量下拉框，将日志主题切换为其他日志主题，使用该变量的图表将自动切换数据源。



自定义变量配置

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**仪表盘** > **仪表盘列表**，进入仪表盘管理页面。
3. 单击需要操作的仪表盘 ID/名称，进入该仪表盘详情页面。
4. 单击顶部的过滤与变量图标，选择**新增过滤与变量**，打开设置页面。



5. 在弹出的窗口中，设置**自定义变量**信息，单击**确定**。



表单元素	说明
类型	变量的类别，不同的类别对应不同的配置项与应用场景，此处选择**自定义变量**。
变量名称	查询检索语句中变量的命名，仅支持字母与数字。通过${变量名称}的格式引用变量。
变量别名	查仪表盘上变量的显示名称，非必填项目，为空时自动使用变量名称为显示名称。
静态变量值	手动添加静态变量值，静态变量值将常驻显示，支持配置别名。
动态变量值	开启后，可以选择一个日志主题并输入检索分析语句，取检索分析的结果作为可选变量值。
默认值	默认使用变量值，为必填项。

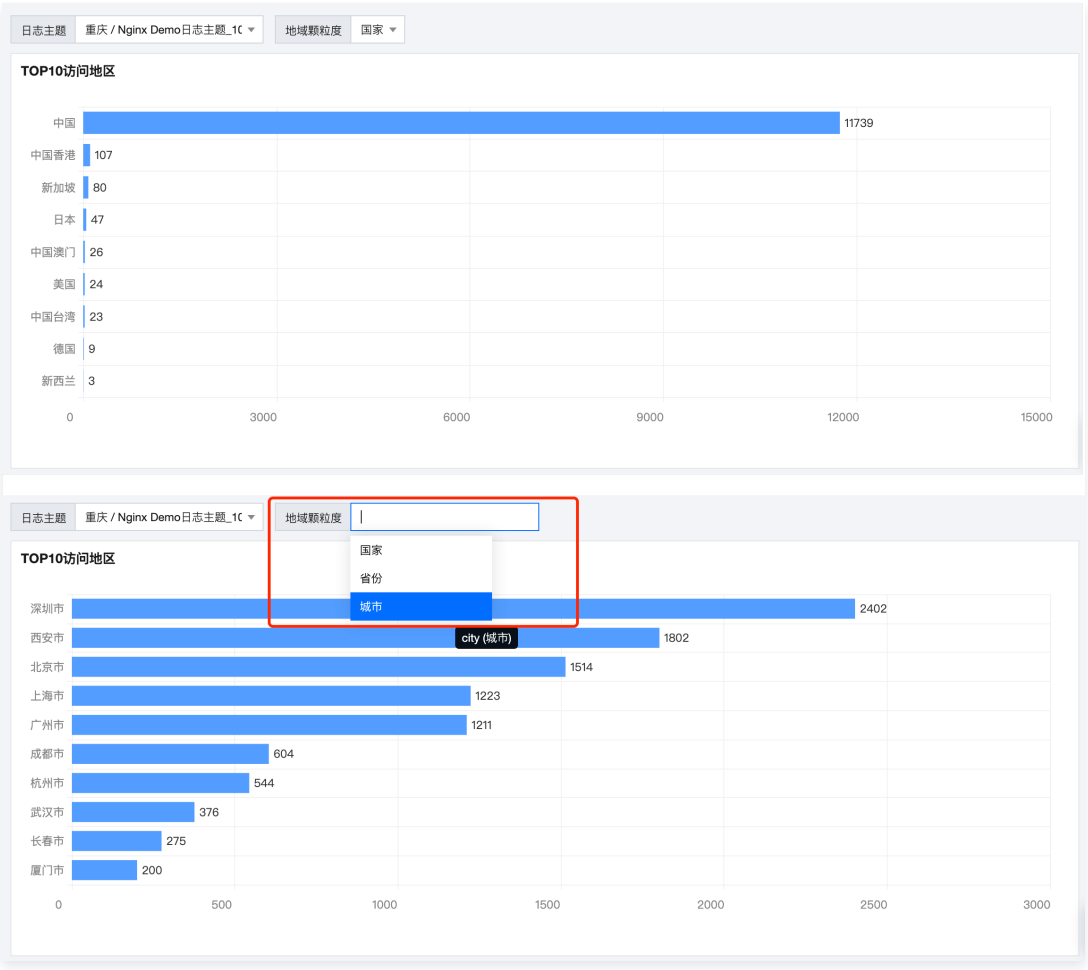
6. 返回仪表盘详情页面，单击图表右上角的 ••• > **编辑**。

> **说明：**
> 如果您的仪表盘中没有图表，请 添加图表 。

7. 在编辑图表**检索语句**内，插入刚才创建的自定义变量**${region}**，替换原始的文本。



8. 单击**应用到仪表盘**。
9. 返回仪表盘详情页面，单击上方的**地域颗粒度**变量下拉框，修改颗粒度，发现插入该变量的图表发生对应的变更。



系统预置变量

系统预置变量无需配置即可在仪表盘中直接使用。

时间颗粒度变量

时间颗粒度变量应用到语句中时将会自动根据所选时间范围适配不同的时间颗粒度，并替换到语句中。用法如下：

```
* | select histogram(__TIMESTAMP__,interval ${__interval}) as time, count(*) as "日志条数" group by time order by time
limit 10000
```

变量	说明	变量替换后的结果格式
${__interval}	变量自动根据时间范围适配不同的时间颗粒度，不指定时间颗粒度单位。	• 10s、15m、2h等（默认格式） • 10 second、15 minute、2 hour等（使用 histogram 函数时）

示例：

- 日志主题：使用 histogram 函数做时序分析。

```
* | select histogram(__TIMESTAMP__,interval ${__interval}) as time, count() as count group by time
order by time limit 10000
```

- 日志主题：使用 time_series 函数做时序分析。

```
* | select time_series(__TIMESTAMP__, '${__interval}', '%Y-%m-%dT%H:%i:%s+08:00', '0')  as time,
count(*) as count group by time order by time limit 10000
```

- 指标主题：统计指标每秒平均变化率。

```
rate(MetricName{label=~".*"}[${__interval}])
```

时间长度变量

时间范围变量，应用到语句中时将会自动根据所选时间范围计算所选时间的长度，并替换到语句中。用法如下：

```
sum_over_time(MetricStorage{TopicsetId=~".*"}[${__range}])
```

变量	说明	变量替换后的结果格式
${__range}	将会自动根据所选时间范围计算所选时间的长度。	10s、150s、300s等
${__range_s}	将会自动根据所选时间范围计算所选时间的长度，指定时间长度的单位为s，结果不携带单位。	10、150、300等
${__range_ms}	将会自动根据所选时间范围计算所选时间的长度，指定时间长度的单位为ms，结果不携带单位。	10000、150000、300000等

示例：

- 指标主题：统计所选时间范围的 PV。

```
sum_over_time(ReqCount{label=~".*"}[${__range}])
```

- 指标主题：统计所选时间范围的 QPS。

```
sum_over_time(ReqCount{label=~".*"}[${__range}])/${__range_s}
```

常见问题

配置并使用数据源变量后，为什么没有生效，或只有部分图表生效?

数据源变量并不会直接针对仪表盘内所有图表生效，只有在图表编辑页面中使用了该变量的图表才会生效。

使用案例

在仪表盘内分析不同应用接口的性能指标（下拉选项过滤）

需求场景

日志主题 A 为某应用的 nginx 访问日志，需要通过仪表盘查看该**应用整体**及**指定某个接口**的吞吐量、错误请求数和响应时间。样例日志如下：

```
body_bytes_sent:1344
client_ip:127.0.0.1
host:www.example.com
http_method:POST
http_referer:www.example.com
http_user_agent:Mozilla/5.0
proxy_upstream_name:proxy_upstream_name_4
remote_user:example
req_id:5EC4EE87A478DA3436A79550
request_length:13506
```

```
request_time:1
http_status:201
time:27/Oct/2021:03:25:24
upstream_addr:219.147.70.216
upstream_response_length:406
upstream_response_time:18
upstream_status:200
interface:proxy/upstream/example/1
```

解决方案

1. 创建仪表盘。
2. 针对应用性能指标，分别创建三个图表（时序图）。其对应的查询语句分别如下：
   - 吞吐量：

     ```
     * | select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 minute) as analytic_time, count(*)
     as pv group by analytic_time order by analytic_time limit 1000
     ```

   - 错误请求数：

     ```
     http_status:>=400 | select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 minute) as
     analytic_time, count(*) as pv_lost group by analytic_time order by analytic_time limit 1000
     ```

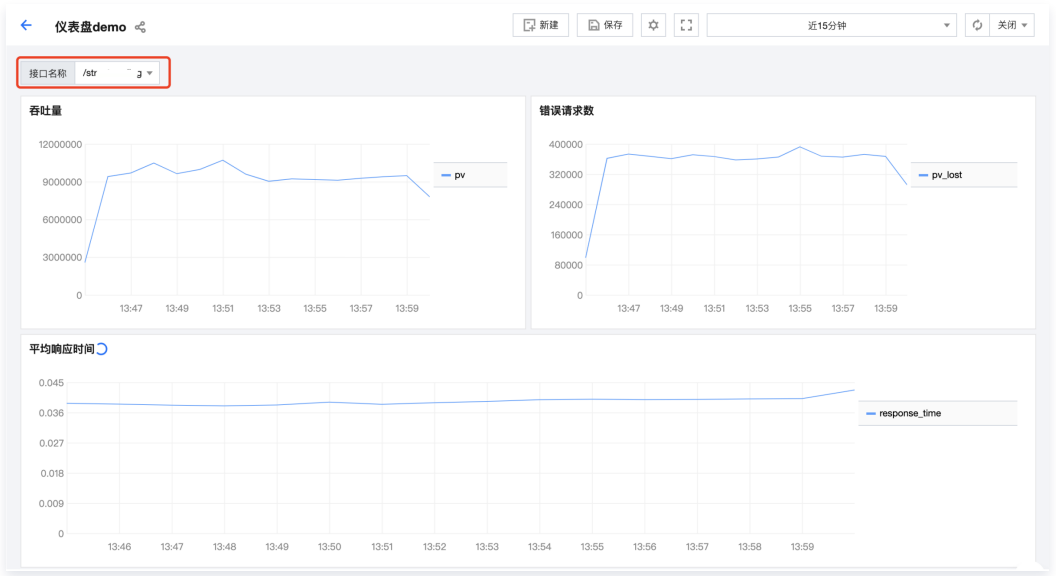
   - 平均响应时间：

     ```
     * | select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 minute) as analytic_time,
     avg(request_time) as response_time group by analytic_time order by analytic_time limit 1000
     ```

3. 添加下拉选项过滤。
   - 类型：下拉选项过滤。
   - 显示名称：接口名称。
   - 日志主题：日志主题 A。
   - 字段选择：interface。
4. 返回仪表盘详情页面，即可在页面顶部看到该变量。
   - **接口名称**没有指定值时，表示不进行数据过滤，仪表盘内各个图表展示的是全部数据，即应用的整体性能指标。



   - **接口名称**指定具体值时，仪表盘内所有图表以该接口作为过滤条件进行数据过滤，展示该接口的性能指标。



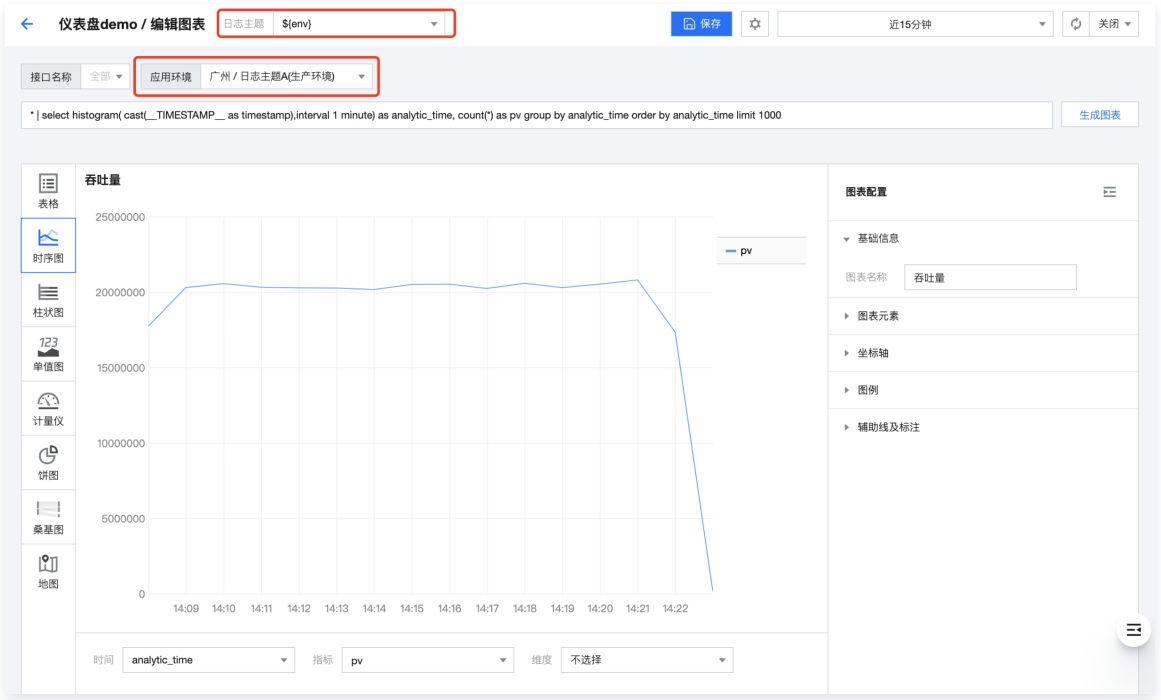
在仪表盘内分别查看生产环境及测试环境的性能指标（数据源变量）

需求场景

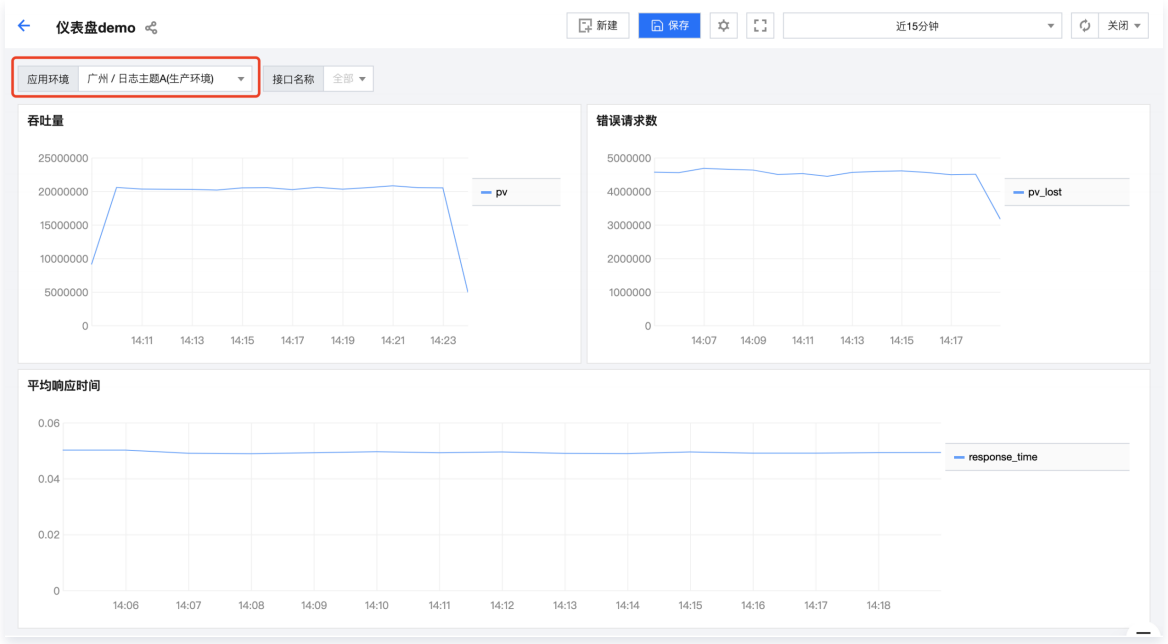
某应用具备生产环境及测试环境，分别将日志采集到“日志主题 A（生产环境）”和“日志主题 B（测试环境）”中。因此，在应用的开发、测试及运维过程中，需要同时关注两个环境的性能指标。

解决方案

1. 创建仪表盘。
2. 添加模板变量。
   - 变量类型：数据源。
   - 变量名称：env。
   - 显示名称：应用环境。
   - 数据源范围：所有日志主题。
   - 默认日志主题：日志主题 A（生产环境）。
3. 添加图表。
   在**日志主题**下拉菜单中，勾选“使用数据源变量”，选择上一步中创建的 `${env}` 变量。图表将使用该变量的值作为当前图表的数据源，即日志主题 A（生产环境）。



4. 重复执行**步骤3**，添加其他图表。
5. 返回仪表盘详情页面，并在该页面顶部单击数据源变量**应用环境**，在该变量的下拉菜单中切换日志主题。同时，使用该变量的图表也将切换日志主题。



交互事件

最近更新时间：2025-07-03 15:06:02

图表交互事件功能，支持单击图表中的数值触发交互事件，如跳转到设定的 URL 地址、打开检索分析页面、打开仪表盘页面、添加过滤条件等。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，进入**检索分析**或**仪表盘**，创建统计分析图表。
3. 在图表配置里选择**交互事件**，添加相关交互。可引用变量如下：

变量引用范围	变量	意义
当前图表	${__meta.topicId}	当前图表数据的日志主题
当前图表	${__meta.startTime}	当前图表数据的开始时间
当前图表	${__meta.endTime}	当前图表数据的结束时间
当前图表	${__meta.region}	当前图表日志主题的所在地域
当前行	${__row.fieldname}	单击行里 fieldname 字段的值（fieldname 为被引用字段的字段名）
当前列	${__field.name}	单击列里 field 字段的字段名
当前值	${__value.raw}	当前值，以原始格式引用
当前值	${__value.text}	当前值，以字符串格式引用
当前值	${__value.numeric}	当前值，以数值格式引用

操作示例

打开外部 URL

场景：分析 CLB 访问日志仪表盘，找出失败率最高的 CLB 实例，跳转到对应 CLB 实例了解实例详情并进行配置。
在 TOP10实例请求失败率(%)图表中，配置**交互事件** > **打开外部 URL**，跳转的 URL 中的 ${__row.lb_id} 是变量，配置如下：




```
https://console.cloud.tencent.com/clb/detail?rid=1&id=${__row.lb_id}
```

配置后，点击对应 lb_id 跳转到实例的详情页面。

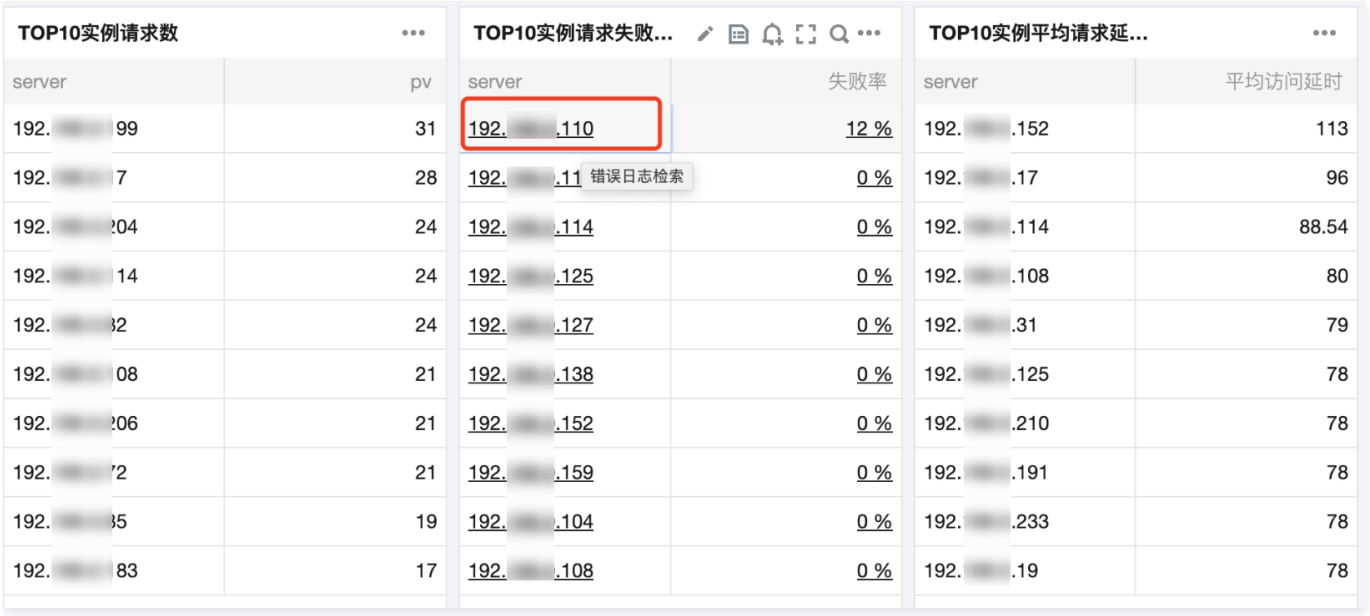
检索其他日志主题

场景：现已经统计了服务端 IP 的错误率的 TOP 情况，当出现错误率较高的 IP 时，需要跳转到检索分析页面，查看该 IP 的 status 状态码为4XX 的日志。
在图表中配置**交互事件** > **检索其他日志主题**，选择沿用当前日志主题，配置检索语句如下图：



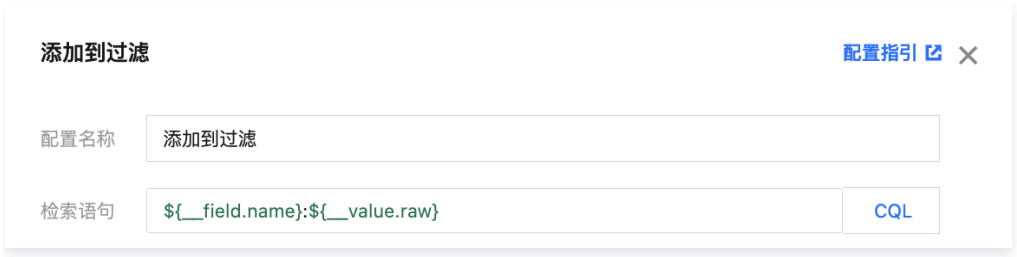

```
server_addr:${__value.raw} AND status:[400 TO 499]
```

单击目标 IP 地址后，自动打开检索页面检索，并显示检索结果。



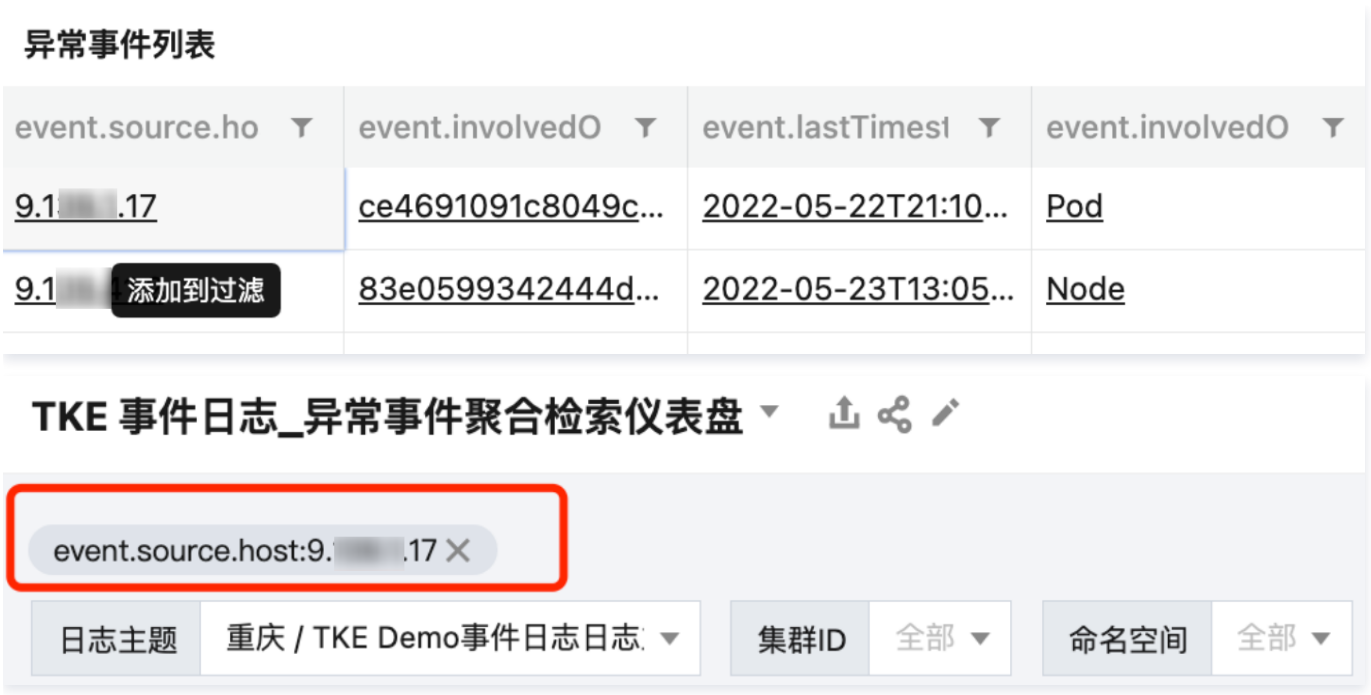
添加到过滤

场景：通过表格查看日志内容，希望任意字段都可以通过点击添加到过滤，便于在浏览时做下钻分析。配置添加到过滤如下：




```
${__field.name}:${__value.raw}
```

配置后可点击任意字段值，都会将字段值添加到过滤。



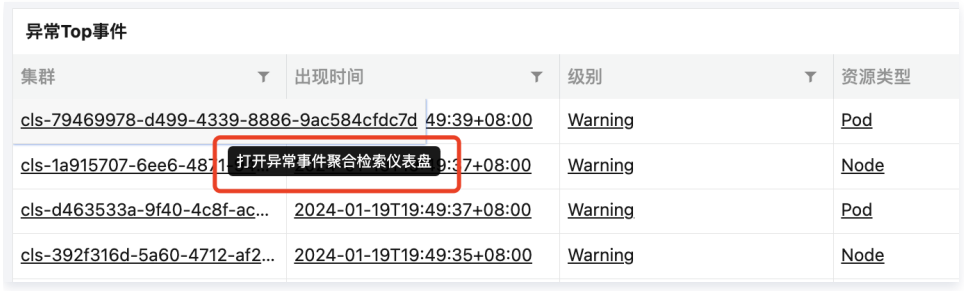
打开仪表盘

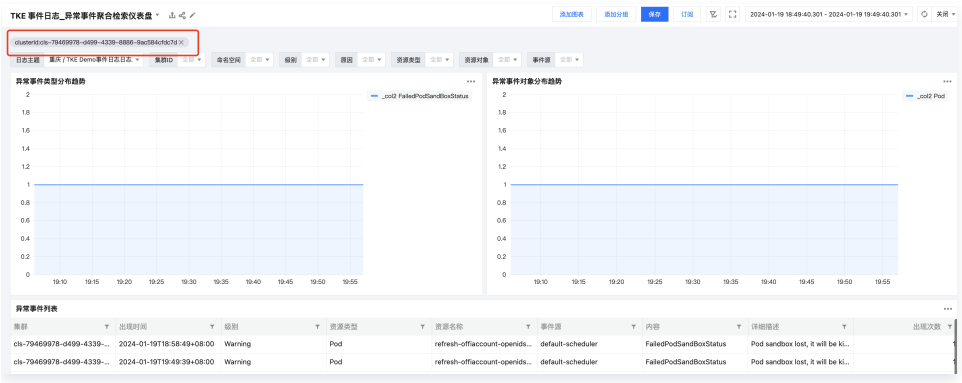
场景：查看“总览仪表盘”，定位出错误数最多的集群。然后携带该集群ID作为过滤条件，打开“异常事件聚合检索分析仪表盘”，查看详情。配置如下：

```
clusterId:${__row.集群}
```



配置后，点击对应集群字段。跳转打开异常事件聚合检索分析仪表盘，且已过滤出该集群的指标。





版权所有：腾讯云计算（北京）有限责任公司

添加分组

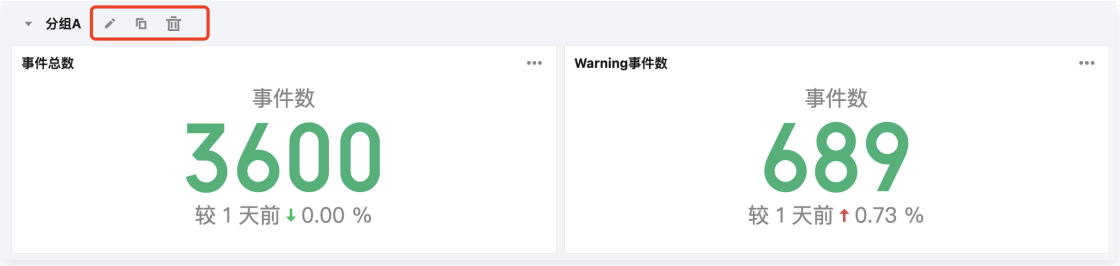
最近更新时间：2024-10-16 16:03:12

操作场景

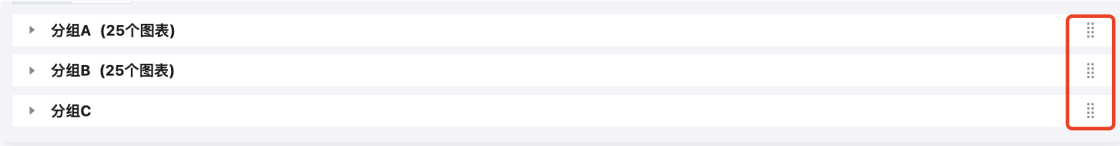
仪表盘支持图表分组，通过分组可以对仪表盘图表内容进行归类和收折。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中单击**仪表盘 > 查看仪表盘**，进入仪表盘页面。
3. 单击顶部**添加分组**，输入分组名称，创建分组。
4. 拖动图表添加到分组中。
5. 分组支持编辑、复制、删除、拖拽操作。复制/删除时，可选择是否同时批量复制/删除组内的所有图表。



收折状态下，分组支持拖拽。



免密分享仪表盘

最近更新时间：2025-07-03 15:06:02

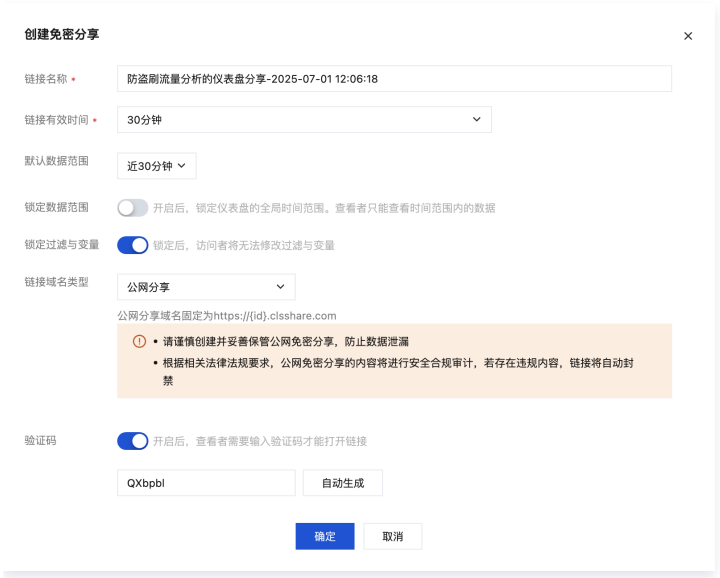
使用场景

免密分享支持将仪表盘或图表分享给其他人临时查看，无需登录鉴权，提高协作效率。

操作步骤

创建免密分享

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**仪表盘** > **查看仪表盘**，进入仪表盘页面。
3. 免密分享入口：
   - **仪表盘免密分享**：单击顶部的**分享**按钮 ，找到**免密分享**，然后单击**新建**。
   - **单个图表免密分享**：单击图表右上角**更多操作**按钮 ，在菜单中选择**创建免密分享**。
4. 创建仪表盘分享。配置项如下：



配置项	说明
链接名称	免密分享链接的名称。
链接有效时间	链接的有效时间，从链接创建成功后开始计算，内网分享最长支持1个月，公网分享最长支持24小时。
默认数据范围	分享内容的默认时间范围，默认使用当前仪表盘的时间范围。
锁定数据范围	开关，开启后锁定仪表盘的全局时间范围。查看者只能查看时间范围内的数据，默认关闭。
锁定过滤与变量	开关，开启后查看者将无法修改过滤与变量，默认开启。
链接域名类型	支持公网分享与内网分享两种类型： • 公网分享：免密链接可以从公网访问，建议同时开启验证码以保证安全性，默认启用验证码。 • 内网分享：免密链接支持内网访问，依赖 DataSight 内网域名，需先创建 DataSight 内网域名，详情请参见 使用 DataSight 独立控制台 。
验证码	开关，开启后查看者需要输入验证码才能打开链接

5. 复制生成的免密分享链接，发送给其他人。

管理免密分享

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**查看仪表盘**，进入仪表盘页面。
3. 单击顶部的**分享**按钮  ，查看免密分享列表。
4. 对列表中的分享链接进行管理，包括：
   - 复制分享链接。
   - 重置分享链接的有效期：分享链接过期后，可以通过重置有效期，使其重新生效。
   - 删除分享链接。

预置仪表盘

最近更新时间：2024-10-24 20:59:22

操作场景

日志服务（Cloud Log Service，CLS）支持接入多种云产品标准日志，详情见 云产品日志接入 。针对这些云产品日志，CLS 提供开箱即用的日志分析仪表盘，用户接入日志后即可快速分析。此外，可以通过 CLS 提供的免费 云产品 Demo 日志 快速体验预置仪表盘。本文为您介绍如何快速体验预置仪表盘。

预置仪表盘列表

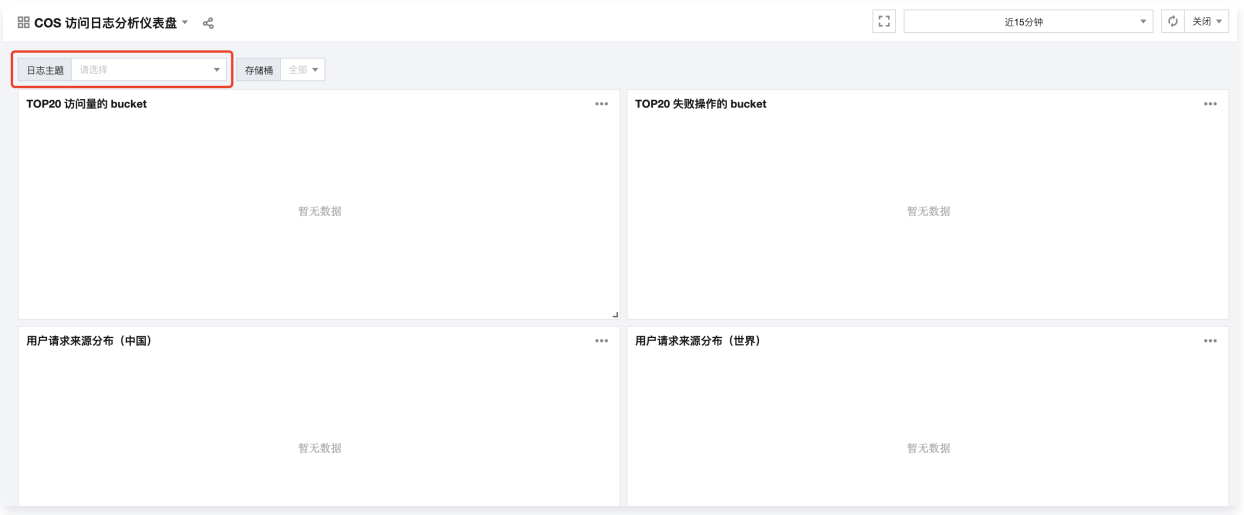
云产品	预置仪表盘
负载均衡（Cloud Load Balancer，CLB）	CLB 访问日志仪表盘
Nginx访问日志	• Nginx 访问大盘 • Nginx 监控大盘
内容分发网络（Content Delivery Network，CDN）	• CDN 访问日志-质量监控分析仪表盘 • CDN 访问日志-用户行为分析仪表盘
对象存储（Cloud Object Storage，COS）	• COS 访问日志分析仪表盘
网络流日志（Flow Logs，FL）	• ENI 流日志-高级分析仪表盘 • CCN 流日志-高级分析仪表盘
容器服务（Tencent Kubernetes Engine，TKE）	• TKE 审计日志-总览仪表盘 • TKE 审计日志-节点操作概览仪表盘 • TKE 审计日志-K8S 对象操作概览仪表盘 • TKE 事件日志-总览仪表盘 • TKE 事件日志-异常事件聚合检索仪表盘
操作审计（CloudAudit）	CloudAudit 审计日志_事件分析仪表盘
API 网关（API Gateway）	API Gateway 访问日志_接口质量分析仪表盘
Web 应用防火墙（Web Application Firewall，WAF）	WAF 访问日志_访问流量分析仪表盘

操作步骤

查看预置仪表盘

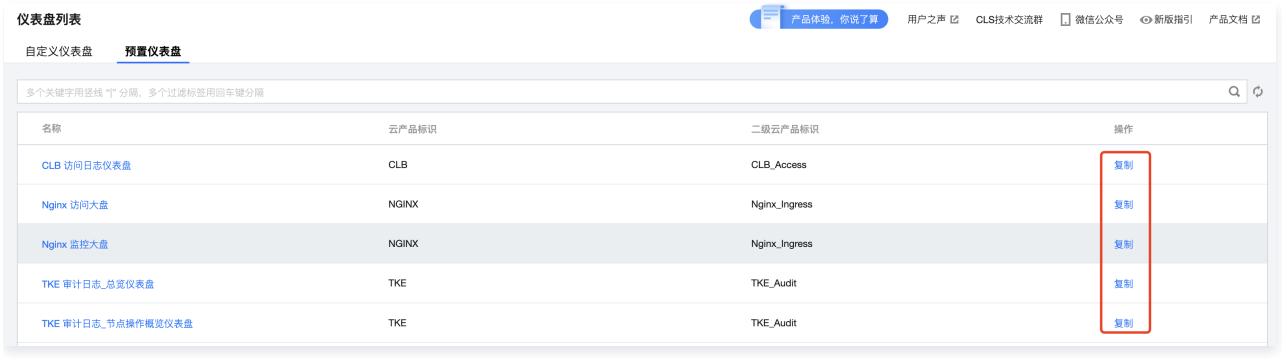
1. 登录 日志服务控制台 。
2. 在左侧导航栏中单击**仪表盘 > 仪表盘列表**，进入仪表盘列表页面。
3. 在仪表盘列表页面，选择**预置仪表盘**，选择并打开目标预置仪表盘。

4. 选择对应云产品的日志主题。



基于预置仪表盘做编辑

如果当前的预置仪表盘不满足需求，需要基于预置仪表盘添加更丰富的统计图表，可单击**复制**，复制预置仪表盘到自定义仪表盘列表，进行自定义编辑。



订阅仪表盘

最近更新时间：2024-08-15 14:35:22

使用场景

日志服务支持订阅仪表盘，可以将仪表盘以图片的形式导出。用邮件等方式定期发送给指定的对象。对于需要面向团队发送日、周、月报表的仪表盘，可使用该功能。

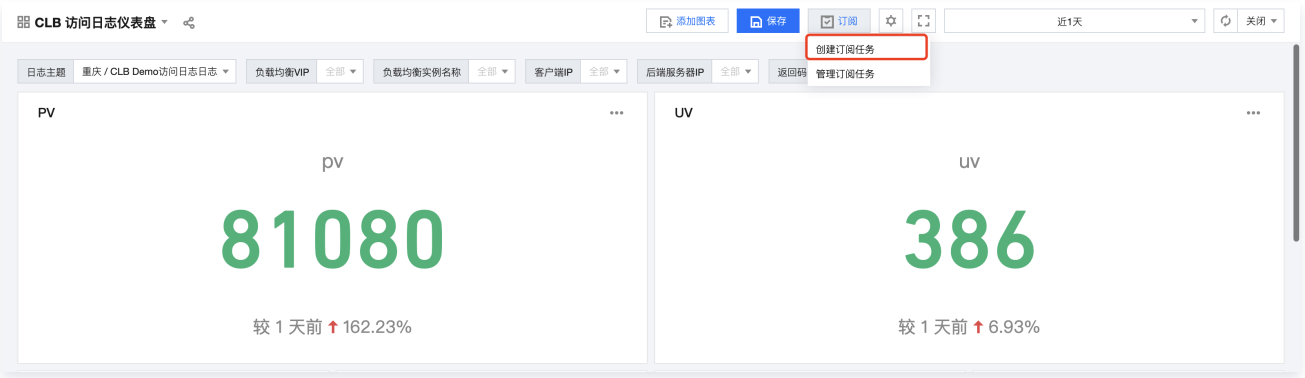
使用限制

每个账户最多支持100个订阅任务。

操作步骤

创建订阅任务

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**查看仪表盘**，进入仪表盘页面。
3. 单击顶部的**订阅**，选择**创建订阅任务**，进入订阅配置页面。



4. 填写订阅**任务名称**，选择订阅仪表盘的**时间范围**。
5. 设置订阅仪表盘的**过滤与变量**。
   - 选择默认配置，则订阅仪表盘的模板变量使用查看仪表盘时的默认配置。当查看仪表盘的默认变量变更时，订阅仪表盘也会发生变更。
   - 选择自定义配置，则自定义设置变量，订阅仪表盘渲染时使用自定义的变量值。
6. 单击**预览内容**，预览订阅仪表盘的效果。
7. 选择订阅周期，支持以每日、每周、每月为周期。每周、每月周期支持多选。
8. 配置订阅方式：

渠道类型	说明
腾讯云用户	选择腾讯云用户作为订阅仪表盘接收对象，支持邮件接收。未配置邮箱地址的用户无法接收邮件。
自定义邮件	填写自定义邮箱地址，可填写多个。
企业微信	通过企业微信机器人 Webhook 地址，推送订阅仪表盘。详情请查看 企业微信群机器人设置
飞书	通过飞书机器人 Webhook 地址，推送订阅仪表盘。详情请查看 飞书群组机器人设置 。其中安全设置推荐使用“自定义关键词”并引用仪表盘名称里的固定词汇，不要使用变量。
钉钉	通过钉钉机器人 Webhook 地址，推送订阅仪表盘。详情请查看 钉钉自定义机器人设置 。其中安全设置推荐使用“自定义关键词”并引用仪表盘名称里的固定词汇，不要使用变量。

9. 仪表盘链接：订阅发送的内容里包含该仪表盘的访问链接，当启用 DataSight 或配置了自定义域名时，可选择使用 DataSight 或自定义域名为访问链接。
10. 单击**立即发送一次**，日志服务将向配置的所有接收对象立即发送一次订阅邮件。

管理订阅任务

1. 登录 日志服务控制台 。

2. 在左侧导航栏中，单击**查看仪表盘**，进入仪表盘页面。
3. 单击顶部的**订阅**，选择**管理订阅任务**，进入订阅任务管理页面。

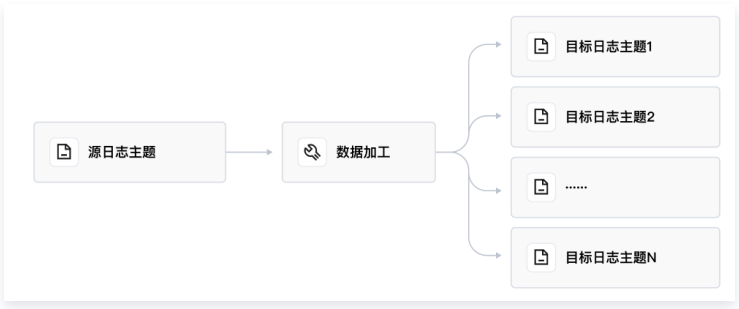
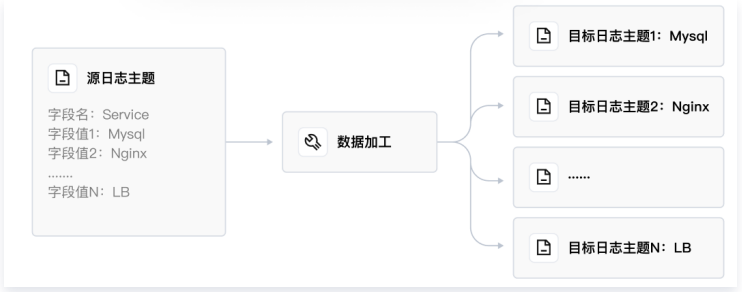


支持对该仪表盘的订阅任务进行新建、编辑、删除、查看上次执行时间的操作。

数据处理
数据加工
数据加工概述

最近更新时间：2024-12-19 16:03:12

数据加工提供对日志数据的过滤、清洗、脱敏、富化、分发等能力。
按照数据加工在数据链路中的位置、source(数据源)和sink(结果保存)不同，当前可支持以下几种数据处理的场景：

场景	说明
	**日志采集-加工-日志主题：** 日志采集至 CLS，先经过数据加工(过滤、结构化)，再写入日志主题。如图，数据加工在数据链路中处于日志主题之前，称之为前置数据加工。 • 在前置数据加工中做 日志过滤，可有效降低日志写流量、索引流量、索引存储量、日志存储量； • 在前置数据加工中做 日志结构化，开启键值索引后，可使用 SQL 对日志进行分析、配置仪表盘和告警。
	**日志主题-加工-固定日志主题：** 将源日志主题中的数据，经过加工之后，存至一个日志主题，或者 分发日志 至多个日志主题。
	**日志主题-加工-动态日志主题：** 按照源日志主题的字段值，动态地创建日志主题，并将相关日志分发至对应的日志主题中。例如源日志主题中有一个字段 Service，有"Mysql"、"Nginx"、"LB"等值，CLS 可以自动创建名字为 Mysql、Nginx、LB 等的日志主题，并将相关日志写入以上主题中。

基本概念

数据加工任务相关的名词和概念，请参见 基本概念 。

功能特性

- **提取结构化的数据**。方便后续 BI 分析、生成监控图表（Dashboard）等。如果您的原始日志不是结构化数据，则无法进行 SQL 计算，也意味着无法对日志进行 OLAP 分析、使用 CLS 仪表盘（基于 SQL 结果绘制图表）等。所以建议您使用数据加工来将非结构化的数据转为结构化的数据。如果您的日志是有规律的，也可以在日志采集时提取结构化数据，请参见 完全正则格式（单行）或者 分隔符格式 。与采集侧相比，数据加工提供更为复杂的结构化处理逻辑。
- **日志过滤**。节约后续使用成本。丢弃不需要的日志数据，节约云上的存储成本和流量成本。例如后续您可能将日志投递到腾讯云 COS、Ckafka，可以有效节约投递的流量。
- **敏感数据脱敏**。例如：将身份证、手机号码等信息脱敏。
- **日志分发**。例如：按照日志级别：ERROR、WARNING、INFO 将日志分类，然后分发到不同的日志主题。

产品优势

- 简单易用，对数据分析师、运维工程师尤为友好。提供开箱即用的函数无需购置、配置、运维 Flink 集群，只需要使用我们封装的 DSL 函数就可以实现海量日志的流处理，包括清洗过滤、脱敏、结构化、分发等，详情请参见 函数总览 。
- 高吞吐实时日志数据流处理。加工效率高（毫秒处理延时）、吞吐高，可达10-20MB/S/分区（源日志主题的分区）。

客户案例

- **数据清洗过滤：**客户 A，丢弃无效日志，仅保留指定字段，补齐部分缺失的字段和字段值。当日志中没有 product_name、sales_manager 字段时，视为无效日志，丢弃该条日志。否则保留日志，并且只保留 price、sales_amount、discount 这三个字段。其它字段 drop 掉，如果该条日志缺失 discount 字段，那么新增这个字段，并给它赋一个默认值，例如“70%”。
- **数据转换：**客户 B，原始日志中的字段值是 IP 地址，客户需要根据 IP，新增国家、城市的字段和值。例如2X0.18X.51.X5，新增字段国家：中国，城市：北京。将 UNIX 时间戳转换为北京时间，例如1675826327，转化为2023/2/8 11:18:47。
- **日志分类投递：**客户 C，原始日志是多层级 JSON，JSON 中还包括了 Array 数组，客户 C 使用数据加工将多层级 JSON 的指定节点的数组提取出来作为字段值，例如从 Array[0]中提取 Auth 字段的值，然后根据 Auth 字段的值，对日志数据进行分发。当值为“SASL”时，投递到目标主题 A；当值为“Kerberos”时，投递到目标主题 B；当值为“SSL”时，投递到目标主题 C。
- **日志结构化：**客户 D，原始日志 `"2021-12-02 14:33:35.022 [1] INFO  org.apache.Load - Response:status: 200, resp msg: OK"` 通过数据加工完成结构化，结果为 `log_time:2021-12-02 14:33:35.022, loglevel:info, status:200` 。
  更多详细信息，请参见 数据加工案例 。

费用说明

数据加工会产生相关的费用，详情请参见 计费概述 。如果您的业务只需要使用加工后的日志，推荐您将源日志主题的保存时间配置为3-7天，不开启源日志主题的索引，可有效节约费用。

规格与限制

数据加工的规格与限制，请参见 规格与限制 。

创建加工任务

最近更新时间：2025-06-24 15:47:31

本文将为您介绍如何创建加工任务。

前提条件

- 已开通日志服务，创建源日志主题、并成功采集到日志数据。
- 已 创建目标日志主题。建议目标日志主题为空主题，便于加工好的数据写入。
- 确保当前操作账号拥有配置数据加工任务的权限，请参见 CLS 访问策略模板。

> **注意：**
> 数据加工只能处理实时日志流，不能处理历史日志。

操作步骤

1. 登录 日志服务控制台，在左侧导航栏中选择**数据加工**。
2. 单击**新建**。配置基本信息：

前置数据加工

前置加工的使用场景：
日志采集至 CLS，先进行数据加工（过滤、结构化），再写入日志主题。
举例：日志主题名为"test"，使用 LogListener 采集了数据，现需将 loglevel="Error"的日志写入"test"，其余的日志不上报，那么可以创建一个前置数据加工任务"my_transform"来完成。



前置数据加工



前置数据加工的价值：

- 在前置加工中做日志过滤，可有效降低日志写流量、索引流量、索引存储量、日志存储量；
- 在前置加工中做日志结构化，开启键值索引后，可使用 SQL 对日志进行分析、配置仪表盘和告警。

> **说明：**
> - 每个日志主题仅能配置一个前置加工任务。
> - 前置加工暂不支持分发日志到多个固定日志主题，或动态日志主题。因此您**无法**在前置加工任务中使用 log_output 和 log_auto_output 函数。
> - 若您新增/修改了采集配置，会导致采集的数据发生变化，因此需要同时调整前置加工的加工语句。
> - 前置加工任务和非前置的加工任务不支持互转。

配置项：

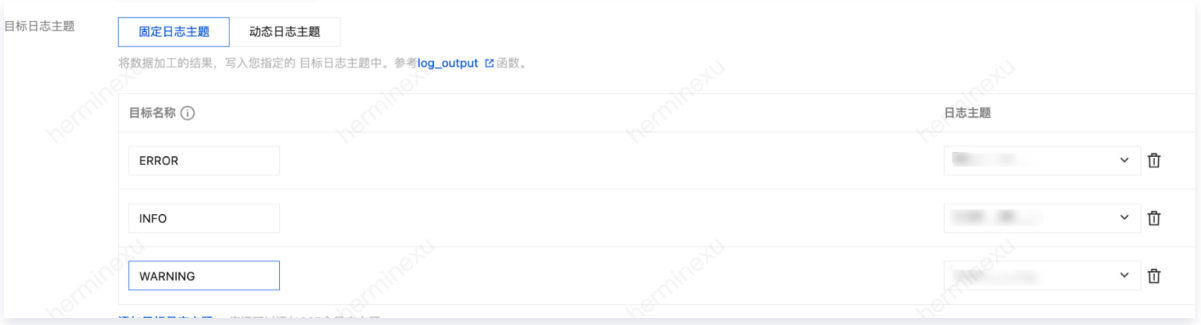
配置项	说明
任务名称	数据加工任务的名称，示例：my_transform。
启用状态	任务启动/停止，默认启动。
前置数据加工	打开开关。 前置数据加工的**功能入口：** • 入口1：在新建数据加工任务时，打开**前置数据加工**的开关。 • 入口2：您也可以在 采集配置 页面底部，单击**数据加工**，进入前置数据加工编辑页面。
日志主题	指定前置加工结果写入的日志主题。
外部数据源	添加外部数据源，可用于维度表 join 场景。当前仅支持腾讯云 MySQL，请参考 res_rds_mysql 函数。 • 地域：云 MySQL 所在地域 • 云数据库 MySQL实例：请在下拉菜单中勾选 • 用户名：输入您的数据库用户名 • 密码：输入您的数据库密码 • 别名：您的 MySQL 的别名，您将在 res_rds_mysql 中使用该别名作为参数。 **注意：** 当前仅上海地域可用。如需使用，请 联系我们 。
数据加工服务日志	数据加工任务运行日志，保存在 cls_service_log 服务日志主题（免费），数据加工任务监控图表中的告警功能依赖该日志主题， 默认开启。
上传加工失败日志	开启后，加工失败的日志，将写入目标主题；关闭该选项后，会丢弃加工失败的日志。
加工失败日志字段名	如您选择将加工失败的日志写入目标日志主题，失败日志将保存在该字段中，字段名默认为 ETLParseFailure。
高级配置	添加环境变量：添加数据加工任务运行时的环境变量。 例如添加一对变量，名称 ENV_MYSQL_INTERVAL，值 300，然后您可在 res_rds_mysql 函数中，使用 refresh_interval=ENV_MYSQL_INTERVAL，任务将解析为 refresh_interval=300。 **注意：** 当前仅上海地域可用。如需使用，请 联系我们 。

非前置数据加工

非前置数据加工主要用于日志分发的场景：

- **分发至固定日志主题的使用场景。**

  用于分发的目标日志主题确定的场景。示例：将源日志主题中的 loglevel= warning 日志输出到名称为 WARNING 的日志主题，将 loglevel=error 日志输出到名称为 ERROR 的日志主题，将 loglevel=info 的日志输出到名称为 INFO 的日志主题。请参见 log_output 函数。



- **分发至动态日志主题的使用场景。**

用于分发目标日志主题较多或者无法确定的场景。示例：您的日志中有一个名为"AppName "的 Key (字段)， 这个字段由于业务的原因，会随着时间不断的新增 Value，而您需要按照"AppName"来分发日志，适合使用此选项。请参见 log_auto_output 函数。



配置项：

分发至固定日志主题

配置项	说明
任务名称	数据加工任务的名称，例如 my_transform。
启用状态	任务启动/停止，默认启动。
前置数据加工	关闭开关。
源日志主题	数据加工任务的数据源。
外部数据源	添加外部数据源，可用于维度表join场景，当前仅支持腾讯云 MySQL，请参见 res_rds_mysql 函数。 • 地域：云 MySQL 所在地域。 • 云数据库 MySQL 实例：请在下拉菜单中勾选。 • 用户名：输入您的数据库用户名。 • 密码：输入您的数据库密码。 • 别名：您的 MySQL 的别名，您将在 res_rds_mysql 中使用该别名作为参数。 ⚠ 注意： 当前仅上海地域可用。如需使用，请 联系我们 。
加工时间范围	指定数据加工处理的日志范围。 ⚠ 注意： 仅处理日志主题生命周期内的数据。
目标日志主题	选择**固定日志主题**： • 日志主题：数据加工结果的输出目标，可配置为1个或多个。 • 目标名称：例如在源日志主题中，将 loglevel= warning 日志输出到日志主题 A，将 loglevel=error 日志输出到日志主题 B，将 loglevel=info 的日志输出到日志主题 C。您可将日志主题 A、B、C 的目标名称分别配置为 warning、error、info。
数据加工服务日志	数据加工任务运行日志，保存在 cls_service_log 服务日志主题（免费），数据加工任务监控图表中的告警功能依赖该日志主题， 默认开启。
上传加工失败日志	开启后，加工失败的日志，将写入目标主题，关闭该选项后，会丢弃加工失败的日志。
加工失败日志字段名	如您选择将加工失败的日志写入目标日志主题，失败日志将保存在该字段中，字段名默认为 ETLParseFailure。
高级配置	添加环境变量：添加数据加工任务运行时的环境变量。 例如添加一对变量， 名称 ENV_MYSQL_INTERVAL，值 300，然后您可在 res_rds_mysql 函数中，使用 refresh_interval=ENV_MYSQL_INTERVAL，任务将解析为 refresh_interval=300。 ⚠ 注意：

	当前仅上海地域可用。如需使用，请 联系我们 。

分发至动态日志主题

配置项	说明
任务名称	数据加工任务的名称，例如 my_transform。
启用状态	任务启动/停止，默认启动。
前置数据加工	关闭开关。
源日志主题	数据加工任务的数据源。
外部数据源	添加外部数据源，可用于维度表 join 场景。当前仅支持腾讯云 MySQL，请参见 res_rds_mysql 函数。 • 地域：云 MySQL 所在地域。 • 云数据库 MySQL 实例：请在下拉菜单中勾选。 • 用户名：输入您的数据库用户名。 • 密码：输入您的数据库密码。 • 别名：您的 MySQL 的别名，您将在 res_rds_mysql 中使用该别名作为参数。 ⚠ 注意： 当前仅上海地域可用。如需使用，请 联系我们 。
加工时间范围	指定数据加工处理的日志范围。 ⚠ 注意： 仅处理日志主题生命周期内的数据。
目标日志主题	选择**动态日志主题**。无需配置目标日志主题，会按照指定的字段值，自动生成。

超限处理	当您的数据加工任务生成的主题个数超出了 产品规格 ，您可以选择：
	• 创建兜底日志集、日志主题并将日志写入兜底主题（创建任务时创建）。
	  ○ 兜底日志集：auto_undertake_logset，单地域单账号下一个。
	  ○ 兜底日志主题：auto_undertake_topic_$(数据加工任务名称)。例如用户创建了两个数据加工任务 etl_A 和 etl_B，那么会产生两个兜底主题：auto_undertake_topic_etl_A、auto_undertake_topic_etl_B。
	• 丢弃日志数据：不创建兜底主题，直接丢弃日志。

数据加工服务日志	数据加工任务运行日志，保存在 cls_service_log 服务日志主题（免费），数据加工任务监控图表中的告警功能依赖该日志主题，默认开启。
上传加工失败日志	开启后，加工失败的日志，将写入目标主题，关闭该选项后，会丢弃加工失败的日志。
加工失败日志字段名	如您选择将加工失败的日志写入目标日志主题，失败日志将保存在该字段中，字段名默认为 ETLParseFailure。
高级配置	添加环境变量：添加数据加工任务运行时的环境变量 例如添加一对变量， 名称 ENV_MYSQL_INTERVAL，值 300，然后您可在 res_rds_mysql 函数中，使用 refresh_interval=ENV_MYSQL_INTERVAL，任务将解析为 refresh_interval=300。 ⚠ 注意： 当前仅上海地域可用。如需使用，请 联系我们 。

3. 配置完成后单击**下一步**。
4. 编辑加工语句。数据加工语句可使用语句模式和交互模式进行编辑。
   - **语句模式**下，您可使用所有加工函数，请参见 数据加工实战案例 或者 函数总览 。
   - **交互模式**下，仅支持部分函数，请参见 可视化函数 。目标日志主题配置为**动态日志主题**时，不支持交互模式。
5. 调试加工语句：您可以使用原始日志或测试数据对 DSL 语句进行调试。
   - **原始日志**：系统会自动加载原始日志数据，默认100条。
   - **测试数据**：系统随机加载的原始数据，可能无法满足您的调试需求，那么您可在**原始数据**页签，单击**加入测试数据**，将原始日志加入到测试数据中，然后在**测试数据**页签，修改这些数据，使其满足您的调试需求。



完成 DSL 加工语句的编写后，单击页面左上角**执行预览**或者**断点调试**（交互模式下为可视化函数右上角的▶），运行和调试 DSL 函数。运行结果会在右侧展示，您可以根据运行结果，调整 DSL 语句，直到满足您的需求。

6. 单击**确定**，提交数据加工任务。

查看加工详情

最近更新时间：2025-06-24 15:47:31

操作场景

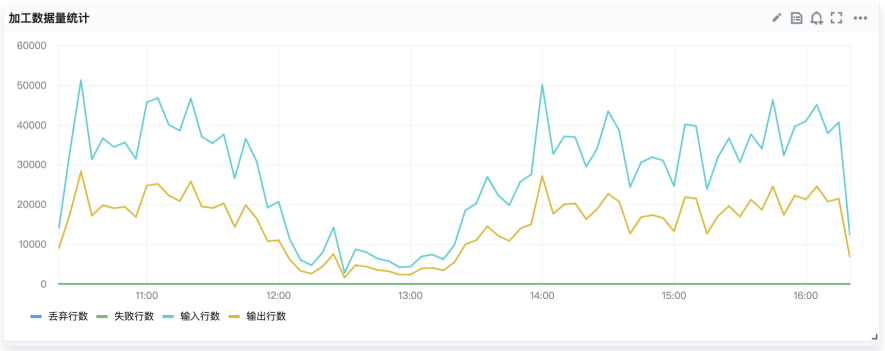
数据加工详情展示了该任务的基本信息和执行详情。

操作步骤

查看任务执行详情

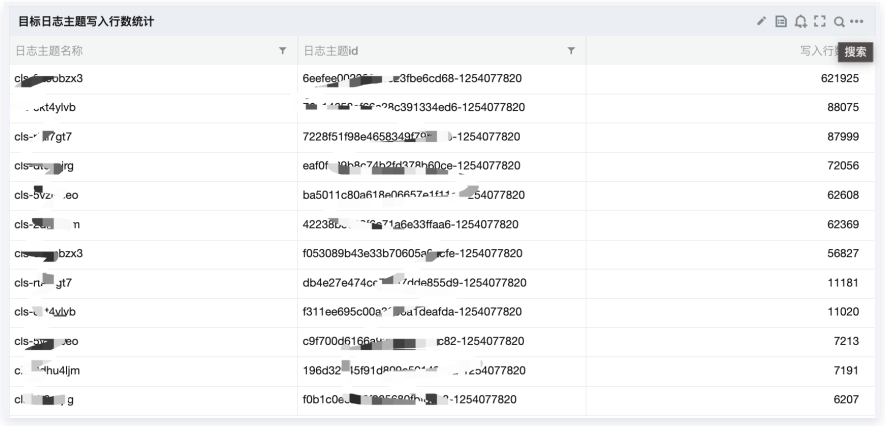
1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**数据加工**，进入数据加工管理页面。
3. 单击需要查看详情的加工任务 ID/名称，进入该加工任务的详情页面。
4. 选择**执行详情**页签，可以按照时间段查看数据加工任务的执行详情。
   - **加工数据量统计**

     包括加工的日志输入行数、输出行数、失败行数、丢弃行数。丢弃行数指的是您使用 log_keep/log_drop 函数处理的日志行数。



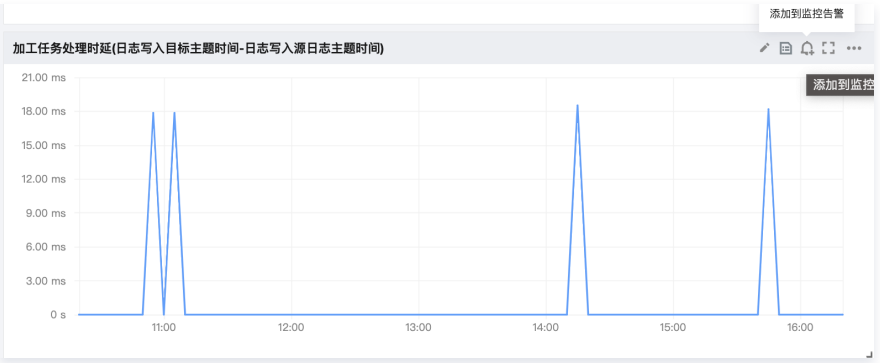
   - **目标日志主题写入行数统计**

     您可在右上角点击搜索按钮，输入关键字搜索具体目标日志主题进行查看。您可在右上角点击，**添加失败行数到监控告警**，详细信息可参见 配置告警策略 。



   - **加工任务处理时延**

     您可在右上角点击，**添加处理时延到监控告警**，详细信息可参见 配置告警策略 。



- **加工失败分析**

  加工失败分析通过展示报错函数、语句、原始日志等信息，对加工失败的数据进行分析。
  例如，正常情况下，原始日志是 `{"content": "{\"sub_field1\":1,\"sub_field2\":\"2\"}"}` ，那么 ext_json("content")是可以正确运行的，当 content 为"" 空字符时，会解析失败。

字段	示例
原始日志	{"content": ""}
错误信息	-
报错原因	parse failed
报错语句所在行数	1
报错函数	ext_json()

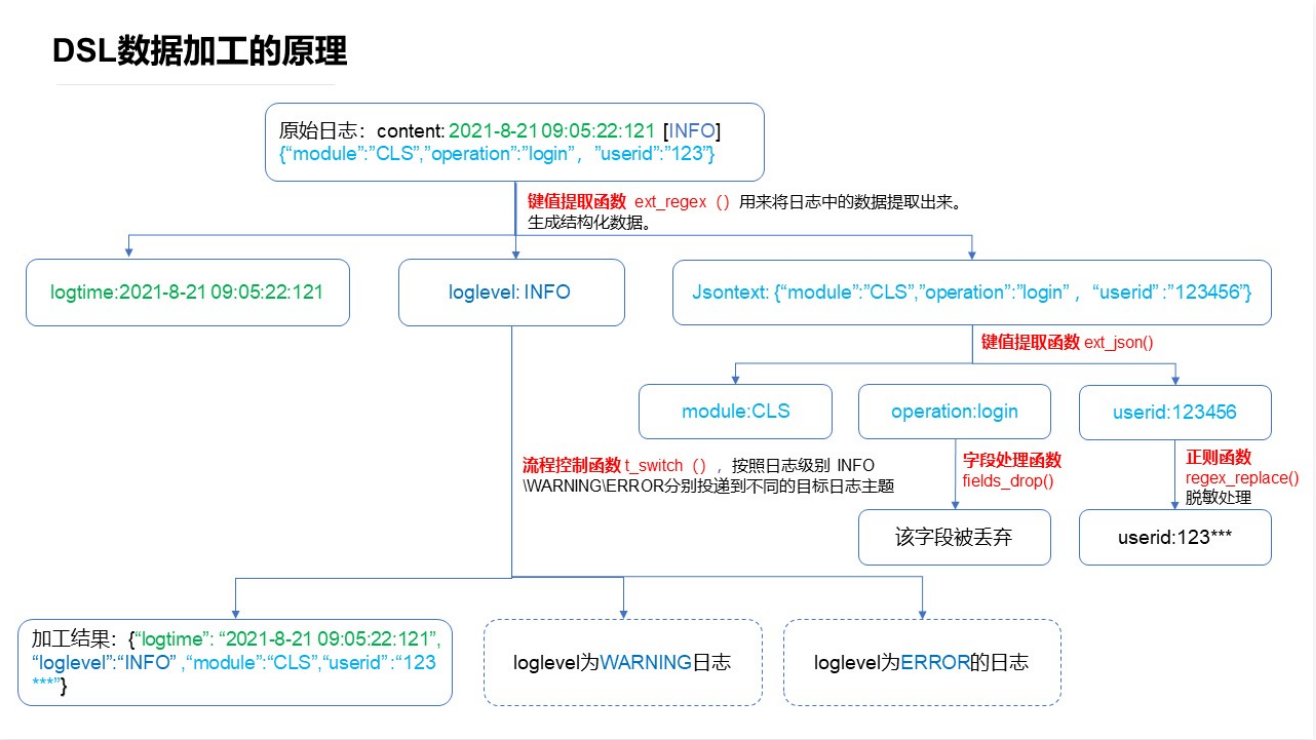
查看任务基本信息

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**数据加工**，进入数据加工管理页面。
3. 单击需要查看详情的加工任务 ID/名称，进入该加工任务的详情页面。
4. 选择**基本信息**页签，可以查看到该加工任务的基本配置、源日志主题、目标日志主题以及加工语法。
5. 如果您的目标日志主题类型为动态日志主题，可查看动态日志主题及其相关的日志集的**创建记录**。

数据加工函数
函数总览

最近更新时间：2025-06-24 15:47:32

日志服务（Cloud Log Service，CLS）数据加工函数可以自由组合，来完成日志的清洗、结构化、过滤、分发、脱敏等场景。
如下图是一条含有 JSON 的日志，经过数据加工处理之后，变成结构化数据、然后对某个字段值进行脱敏、最后分发的流程。



函数概览

键值提取函数类

从日志文本中提取字段/字段值。

函数名称	函数功能	函数语法描述	返回值类型
ext_sep	基于分隔符提取字段值内容	ext_sep("源字段名", "目标字段1,目标字段2,目标字段...", sep="分隔符", quote="不参与分割的部分", restrict=False, mode="overwrite")	返回提取后的日志(LOG)
ext_sepstr	基于指定字符（串）提取字段值内容	ext_sepstr("源字段名","目标字段1,目标字段2,目标字段...", sep="abc", restrict=False, mode="overwrite")	返回提取后的日志(LOG)
ext_json	提取 JSON 字符串格式的字段值	返回提取后的日志(LOG) ext_json("源字段名",prefix="",suffix="",format="full",exclude_node="不平铺的JSON节点")	返回提取后的日志(LOG)
ext_json_jmes	使用 Jmes 表达式提取字段值	ext_json_jmes(“源字段名”, jmes= "提取JSON的公式", output="目标字段", ignore_null=True, mode="overwrite")	返回提取后的日志(LOG)
ext_kv	基于两级分割符提取字段值	ext_kv("源字段名", pair_sep=r"\s", kv_sep="=", prefix="", suffix="", mode="fill-auto")	返回提取后的日志(LOG)
ext_regex	基于正则表达式提取字段值	ext_regex(“源字段名”, regex="正则表达式", output=“目标字段1，目标字段2，目标字段.....”, mode="overwrite")	返回提取后的日志(LOG)

ext_first_notnull	返回参数中第一个非 null 且非空字符的结果值	ext_first_notnull(值1, 值2, ...)	返回参数中第一个非 null 结果值
ext_grok	基于 grok 语法提取匹配结果值	ext_grok(字段值, grok="", extend="")	返回提取后的日志(LOG)

富化函数类

在已有字段的基础上，根据规则新增字段。

函数名称	函数功能	函数语法描述	返回值类型
enrich_table	使用 csv 结构数据对日志中的字段进行匹配。当值相同时，可以将 csv 中的其他字段和值，添加到源日志中	enrich_table(“csv 源数据”, “csv 富化字段”, output=“目标字段1，目标字段2，目标字段....”, mode="overwrite")	返回映射后的日志(LOG)
enrich_dict	使用 Dict 结构对日志中的字段值进行匹配。当指定的字段的值和 Dict 中的 Key 相同时，将此 Key 对应的 Value 赋值给日志中的另一字段	enrich_dict(“JSON 字典”, "源字段名", output=目标字段, mode="overwrite")	返回映射后的日志(LOG)

流程控制函数类

由于条件判断。

函数名称	函数功能	函数语法描述	返回值类型
compose	组合操作函数，类似于分支代码块的组合能力，可以组合多个操作函数，并按顺序执行，可以结合分支、输出函数使用	compose("函数1","函数2", ...)	返回日志(LOG)
t_if	对符合条件的日志，进行相应的函数处理，否则不进行任何处理	t_if("条件", 函数)	返回日志(LOG)
t_if_not	对不符合条件的日志，进行相应的函数处理，否则不进行任何处理	t_if_not("条件",函数)	返回日志(LOG)
t_if_else	基于条件判断，分别进行不同的函数处理	t_if_else("条件", 函数1, 函数2)	返回日志(LOG)
t_switch	基于多分支条件，分别进行不同的函数处理，如果存在不符合所有条件的数据，将被丢弃	t_switch("条件1", 函数1, "条件2", 函数2, ...)	返回日志(LOG)

行处理函数类

用于日志的分发、丢弃、拆分。

函数名称	函数功能	函数语法描述	返回值类型
log_output	输出到指定的目标主题。可以配合分支条件使用，也可以单独使用	log_output(日志主题别名)，该参数在新建数据加工任务时，目标日志主题别名处配置	无返回，对目前的数据流进行输出
log_auto_output	将日志输出到动态的目标主题。例如您需要根据日志字段"pd"的值，动态创建多个目标日志主题，并将对应的日志分发至目标日志主题。	log_auto_output(topic_name="", logset_name="", index_options="", period=3，storage_type="",hot_period=0)	无返回，对目前的数据流进行输出
log_split	使用分隔符结合 Jmes 表达式，对特定字段进行拆分，拆分结果分裂为多行日志	log_split(字段名, sep=",", quote="\", jmes="", output="")	返回日志(LOG)
log_drop	丢弃符合条件的日志	log_drop(条件1)	返回日志(LOG)
log_keep	保留符合条件的日志	log_keep(条件1)	返回日志(LOG)
log_split_jsonarray_jmes	将日志根据 Jmes 语法将 JSON 数组拆分和展开	log_split_jsonarray_jmes("field", jmes="items", prefix="")	返回日志(LOG)

字段处理函数

用于字段的增删改查、重命名。

函数名称	函数功能	函数语法描述	返回值类型
extract_tag	为动态生成的主题打标签。从日志字段中提取标签值，并将这些标签值作为动态生成主题的标签	extract_tag(标签名1，标签值1，标签名2，标签值2....)	返回日志(LOG)
fields_drop	根据字段名进行匹配，删除匹配到的字段	fields_drop(字段名1, 字段名2, ..., regex=False,nest=False)	返回日志(LOG)
fields_keep	根据字段名进行匹配，保留匹配到的字段	fields_keep(字段名1, 字段名2, ..., regex=False)	返回日志(LOG)
fields_pack	根据正则表达式来匹配字段名，并将匹配到的字段打包到新的字段，新字段值使用 JSON 格式进行组织	fields_pack(目标字段名, include=".*", exclude="", drop_packed=False)	返回日志(LOG)
fields_set	用来设置字段值，或者增加新字段	fields_set(字段名1, 字段值1, 字段名2, 字段值2, mode="overwrite")	返回日志(LOG)
fields_rename	字段重命名	fields_rename(字段名1, 新字段名1, 字段名2, 新字段名2, regex=False)	返回日志(LOG)
has_field	字段存在时，返回 True，否则返回 False	has_field(字段名)	返回条件值(BOOL)
not_has_field	字段不存在时，返回 True，否则返回 False	not_has_field(字段名)	返回条件值(BOOL)
v	获取字段值，返回对应字符串	v(字段名)	返回值字符串类型(STRING)

字典和列表函数

函数名称	函数功能	函数语法描述	返回值类型
array_get	获取列表(数组)的值	array_get(数组,下标位置)	返回值字符串类型(STRING)
json_edit	增加/删除/修改字典中的键值对	json_edit("result", path="", key="",value="333",index=1,mode="edit")	返回 dict
json_select	通过 Jmes 表达式，提取 JSON 字段值，并返回 Jmes 提取结果的 JSON 字符串	json_select(v(字段名), jmes="")	返回值字符串类型(STRING)
xml_to_json	解析 XML 值并转换为 JSON 字符串，输入值必须为 XML 字符串结构，否则会导致转换异常	xml_to_json(字段值)	返回值 dict
json_to_xml	解析 JSON 字符串值并转换为 XML 字符串	json_to_xml(字段值)	返回值字符串类型(STRING)
if_json	判断是否为 JSON 字符串	if_json(字段值)	返回条件值(BOOL)

正则处理函数类

使用正则公式对文本中的字符进行匹配、替换。

函数名称	函数功能	函数语法描述	返回值类型
sensitive_detection	检测敏感信息，例如身份证、银行卡等敏感信息	sensitive_detection(scope="", ratio=1, discover_items="", replace_items="")	返回值字符串类型(STRING)

regex_match	基于正则对数据进行匹配，返回是否匹配成功，可以选择全匹配还是部分匹配	regex_match(字段值, regex="", full=True)	返回条件值(BOOL)
regex_select	基于正则对数据进行匹配，返回相应的部分匹配结果，可以指定匹配结果的第几个表达式，以及第几个分组（部分匹配+指定捕获组序号），如果最终没有匹配结果，则返回空字符串	regex_select(字段值, regex="", index=1, group=1)	返回值字符串类型(STRING)
regex_split	基于正则对数据进行分割，返回 JSON Array 字符串（部分匹配）	regex_split(字段值, regex=\"\", limit=100)	返回值字符串类型(STRING)
regex_replace	基于正则匹配并替换（部分匹配）	regex_replace(字段值, regex="", replace="", count=0)	返回值字符串类型(STRING)
regex_findall	基于正则进行匹配，并将匹配结果添加到 JSON 数组中，并返回 Array 字符串（部分匹配）	regex_findall(字段值, regex="")	返回值字符串类型(STRING)

日期处理类

函数名称	函数功能	函数语法描述	返回值类型
dt_str	将时间类的字段值（特定格式的日期字符串或者时间戳），转换为指定时区、格式的目标日期字符串	dt_str(值, format="格式化字符串", zone="")	返回值字符串类型(STRING)
dt_to_timestamp	将时间类的字段值（特定格式的日期字符串），同时指定字段对应的时区，转换为 UTC 时间戳	dt_to_timestamp(值, zone="")	返回值字符串类型(STRING)
dt_from_timestamp	将时间类的时间戳字段，指定目标时区后，转换为时间字符串	dt_from_timestamp(值, zone="")	返回值字符串类型(STRING)
dt_now	获取加工计算时的本地时间	dt_now(format="格式化字符串", zone="")	返回值字符串类型(STRING)
custom_cls_log_time	自定义日志时间，会按照您的加工规则生成新的日志时间，支持 秒、毫秒、微秒、纳秒。	custom_cls_log_time(time)	UTC 时间戳类型

字符串处理类

函数名称	函数功能	函数语法描述	返回值类型
str_exist	判断是否存在子串	str_exist(值1, 值2, ignore_upper=False)	返回计算的结果(BOOL)
str_count	在值中指定范围内查找子串，返回子串出现的次数	str_count(值, sub="", start=0, end=-1)	返回子串次数(INT)
str_len	返回字符串长度	str_len(值)	返回字符串长度(INT)
str_uppercase	返回大写字符串	str_uppercase(值)	返回值字符串类型(STRING)
str_lowercase	返回小写字符串	str_lowercase(值)	返回值字符串类型(STRING)
str_join	使用拼接字符串，拼接多值	str_join(拼接字符串1, 值1, 值2, ...)	返回值字符串类型(STRING)
str_replace	替换字符串，返回替换结果字符串	str_replace(值, old="", new="", count=0)	返回值字符串类型(STRING)
str_format	格式化字符串，返回格式化结果	str_format(格式化字符串, 值1, 值2, ...)	返回值字符串类型(STRING)
str_strip	移除字符串的开头和结尾处的指定字符	str_strip(值, chars="\t\r\n")	返回值字符串类型(STRING)

版权所有：腾讯云计算（北京）有限责任公司

str_lstrip	移除字符串开头处的指定字符。	str_strip(值, chars="\t\r\n")	返回值字符串类型(STRING)
str_rstrip	移除字符串结尾处的指定字符。	str_strip(值, chars="\t\r\n")	返回值字符串类型(STRING)
str_find	在值中查找子串，并返回子串出现的位置	str_find(值, sub="", start=0, end=-1)	返回指定第一次出现在值中的子字符串的位置(INT)
str_start_with	判断字符串是否以指定字符串开头	str_start_with(值, sub="", start=0, end=-1)	返回是否匹配的结果(BOOL)
str_end_with	判断字符串是否以指定字符串结尾	str_end_with(值, sub="", start=0, end=-1)	返回是否匹配的结果(BOOL)

类型转换函数类

函数名称	函数功能	函数语法描述	返回值类型
ct_int	对值进行整型转换，可指定原值的进制，转为十进制数值	ct_int(值1, base=10)	返回求值结果
ct_float	将值转换为浮点型数值	ct_float(值)	返回求值结果
ct_str	将值转换为字符串	ct_str(值)	返回求值结果
ct_bool	将值转换为布尔值	ct_bool(值)	返回求值结果

逻辑表达式函数类

函数名称	函数功能	函数语法描述	返回值类型
op_if	根据条件判断，返回相应的值	op_if(条件1, 值1, 值2)	条件为 True 时，返回值1，否则返回值2
op_and	对值进行 and 运算，均为 True 时，返回 True，否则返回 False	op_and(值1, 值2, ...)	返回计算的结果(BOOL)
op_or	对值进行 or 运算，若存在参数值为 False 则返回 False，否则返回 True	op_or(值1, 值2, ...)	返回计算的结果(BOOL)
op_not	对值进行 not 运算	op_not(值)	返回计算的结果(BOOL)
op_eq	对值进行比较，相等则返回 True	op_eq(值1, 值2)	返回比较的结果(BOOL)
op_ge	对值进行比较，值1大于或等于值2时返回 True	op_ge(值1, 值2)	返回比较的结果(BOOL)
op_gt	对值进行比较，值1大于值2时返回 True	op_gt(值1, 值2)	返回比较的结果(BOOL)
op_le	对值进行比较，值1小于或等于值2时返回 True	op_le(值1, 值2)	返回比较的结果(BOOL)
op_lt	对值进行比较，值1小于值2时返回 True	op_lt(值1, 值2)	返回比较的结果(BOOL)
op_add	对值进行求和运算	op_add(值1, 值2)	返回求值结果
op_sub	对值进行求差运算	op_sub(值1, 值2)	返回求值结果
op_mul	对值进行乘积运算	op_mul(值1, 值2)	返回求值结果
op_div	对值进行除法运算	op_div(值1, 值2)	返回求值结果
op_sum	对多值累加求和	op_sum(值1, 值2, ...)	返回求值结果
op_mod	对值进行模计算	op_mod(值1, 值2)	返回求值结果
op_null	对值进行是否为 null 判断，是则返回 True，否则返回 False	op_null(值)	返回计算的结果(BOOL)

版权所有：腾讯云计算（北京）有限责任公司

op_notnull	对值进行是否为非 null 判断，是则返回 True，否则返回 False	op_notnull(值)	返回计算的结果(BOOL)
op_str_eq	对字符串值进行比较，相等则返回 True	op_str_eq(值1, 值2, ignore_upper=False)	返回计算的结果(BOOL)
random	生成随机数（两个参数之间的一个随机数），左闭右闭	random(值1, 值2)	返回求值结果

编解码函数类

函数名称	函数功能	函数语法描述	返回值类型
decode_url	将编码 URL 进行解码	decode_url(值)	返回值字符串类型(STRING)
md5_encoding	计算并返回 MD5 值	md5_encoding(值)	计算并返回 MD5 值
uuid	生成 universally unique identifier (UUID)，即唯一识别码	uuid()	计算并返回 UUID 值
str_encode	对字符串进行指定格式的编码	str_encode(值)	返回值字符串类型(STRING)
str_decode	对字符串进行指定格式的解码	str_decode(值)	返回值字符串类型(STRING)
base64_encode	将字符串进行 base64 编码	base64_encode(值, format="RFC3548")	返回base64字符串类型(STRING)
base64_decode	将字符串进行 base64 解码	base64_decode(值, format="RFC3548")	返回值字符串类型(STRING)

IP 解析函数类

函数名称	函数功能	函数语法描述	返回值类型
geo_parse	解析出函数的地理位置	geo_parse(字段值, keep=("country","province","city"), ip_sep=",")	返回 JSON 字符串
is_subnet_of	判断 IP 是否在该网段内，支持多个网段	is_subnet_of (网段列表, ip)	返回计算的结果(BOOL)

字段提取模式说明

相关函数

分类	函数	提取模式默认值
字段处理	fields_set	overwrite
键值提取	ext_sep	overwrite
	ext_sepstr	overwrite
	ext_regex	overwrite
	ext_kv	overwrite
	ext_json_jmes	overwrite
富化函数	enrich_table	overwrite
	enrich_dict	overwrite

提取模式说明

原始日志：

```
{
    "a": "",
    "b": 100
}
```

参数名	说明	示例	
		加工语句	加工结果

add	当目标字段不存在时，设置目标字段	fields_set("c", 200,mode="add")	{
			"a":"",
			"b":"100",
			"c":"200"
			}

add-auto	当新值非空，且目标字段不存在时，设置目标字段。	fields_set("c", "",mode="add-auto")	{
			"a":"",
			"b":"100"
			}

fill	当目标字段不存在或者值为空字符串时，设置目标字段。	fields_set("a", "123",mode="fill")	{
			"a":"123",
			"b":"100"
			}

fill-auto	当新值非空，且目标字段不存在或者值为空字符串时，设置目标字段。	fields_set("a", "123",mode="fill-auto")	{
			"a":"123",
			"b":"100"
			}

overwrite	总是设置目标字段。	fields_set("a", "123",mode="overwrite")	{
			"a":"123",
			"b":"100"
			}

overwrite-auto	当新值非空字符串，设置目标字段。	fields_set("b", "123",mode="overwrite-auto")	{
			"a":"null",
			"b":"123"
			}

版权所有：腾讯云计算（北京）有限责任公司

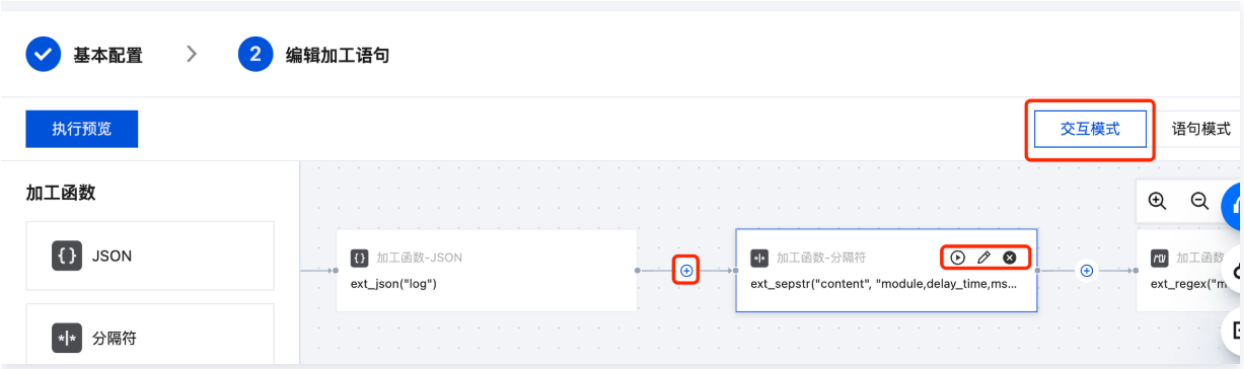
可视化函数

最近更新时间：2025-06-24 15:47:32

> **注意：**
> 交互模式仅支持部分函数。当目标日志主题为动态日志主题时，不支持交互模式.

管理可视化函数

您可在**编辑加工语句**页面切换至**交互模式**。



- **添加**可视化函数：单击⊕，选中您需要添加的可视化函数大类、可视化函数名称。
- **调试**可视化函数：单击可视化函数右上角的▶。
- **删除**可视化函数：单击可视化函数右上角的 X。
- **编辑**可视化函数：单击可视化函数右上角的 。

当前支持的可视化函数如下：

函数大类	可视化函数名称	使用场景
提取键值	• JSON：从 JSON 节点中提取字段和字段值 • 分隔符：根据分隔符提取字段值，需要用户填字段名 • 正则：根据正则公式提取字段值，需要用户填写字段名	日志结构化

日志处理	• 过滤日志：配置过滤日志的条件（多个条件为 OR 的关系）。例如字段 A 存在，或者字段 B 不存在，则过滤掉该条日志。
	• 分发日志：配置分发日志的条件
	  ○ 例如 status="error"且 message 中包含 "404"的，分发至 A 主题
	  ○ 例如 status="running"且 message 中包含 "200"的，分发至 B 主题
	• 保留日志：配置保留日志的条件
		删除/保留日志

字段处理	• 删除字段 • 重命名字段	删除/重命名字段

加工案例

日志结构化

原始日志

日志中有两个字段 log 和 message，可对其进一步结构化。

```
{
    "log": "{\"offset\":281,\"file\":{\"path\":\"/logs/gate.log\"}}",
    "message": "2024-10-11 15:32:10.003 DEBUG [gateway3036810e0c33b] ",
    "content":"cls_ETL|1.06s|fields_renamed"
```

```
}
```

可视化加工函数的配置



可视化函数	配置项目	例子中配置的值	是否必填	说明
加工函数-JSON	描述	-	否	填写您对该加工函数的描述
	字段	log	是	选中您要处理的原字段
	新字段前缀	-	否	为提取出来的新字段名称添加前缀
	新字段后缀	-	否	为提取出来的新字段名称添加后缀
	原字段自动删除	☑	否	加工完成后，删除原字段
加工函数-正则	描述	-	否	填写您对该加工函数的描述
	字段	message	是	选中您要处理的原字段
	正则表达式	(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}.\d{3}) ([A-Z]{5}|[A-Z]{4}) \ [(.+)\]	是	• \d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}.\d{3} 提取为新字段 dates • ([A-Z]{5}|[A-Z]{4})提取为新字段 level • [(.+)\]提取为新字段 traceid
	新字段名	dates、level、traceid	是	使用正则公式提取的新字段
	原字段自动删除	☑	否	加工完成后，删除原字段
加工函数-分隔符	描述	-	否	填写您对该加工函数的描述
	字段	content	是	选中您要处理的原字段
	分隔符	:	是	分隔日志为多个新字段的值
	新字段名	module、delay_time、msg	是	使用分隔符提取的新字段
	原字段自动删除	☑	否	加工完成后，删除原字段

加工结果

提取出来新的字段和字段值如下：

```
{
"dates":"2024-10-11 15:32:10.003",
"delay_time":"1.06s",
"level":"DEBUG",
"module":"cls_ETL",
"msg":"fields_renamed",
"offset":"281",
"path":"/logs/gate.log",
"traceid":"gateway3036810e0c33b"
}
```

日志分发

原始日志

根据 loglevel 的值分发日志。

```
{
  "__FILENAME__": "python.log",
  "__SOURCE__": "127.0.0.1",
  "log_level": "ERROR",
  "status": "404",
  "time": "2024-07-21 05:17:30.421"
}
```

可视化加工函数的配置



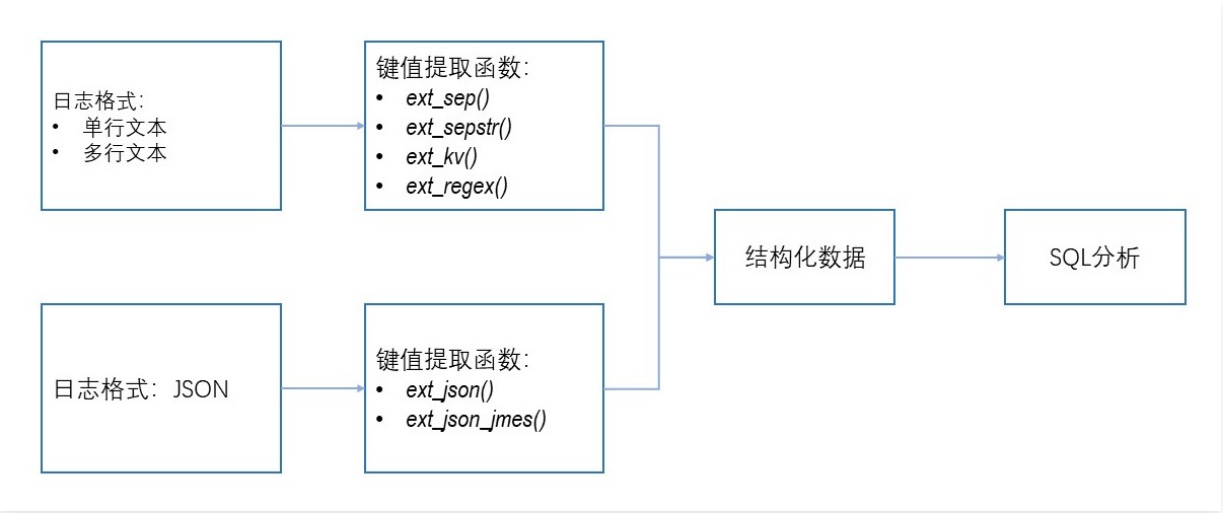
可视化函数	配置项目	例子中配置的值	是否必填	说明
加工函数-日志分发	描述	-	否	填写您对该加工函数的描述
	分组1	log_level="info"	是	配置您的分发条件，当日志满足条件时，将会发送到对应的日志主题。例子中是名为 info 的日志主题
	分组2	log_level="warning"	否	配置您的分发条件，当日志满足条件时，将会发送到对应的日志主题。例子中是名为 warning 的日志主题
	分组3	log_level=" error"且 status="400"	否	配置您的分发条件，当日志满足条件时，将会发送到对应的日志主题。例子中是名为 error 的日志主题

键值提取函数

最近更新时间：2025-03-21 10:29:12

简介

键值提取函数常见的使用场景如下图，处理为结构化数据之后，可以进一步用于 SQL 分析的场景。



ext_sep() 函数

函数定义

基于分隔符（单字符）提取字段值内容。

语法描述

```
ext_sep("源字段名", "目标字段1,目标字段2,目标字段...", sep="分隔符", quote="不参与分割的部分", restrict=False, mode="overwrite")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	待提取的字段名	string	是	-	用户日志中已存在的字段名
output	单个字段名或以英文半角逗号拼接的多个新字段名	string	是	-	-
sep	分隔符	string	否	,	任意单字符
quote	将值包括起来的字符	string	否	-	-
restrict	默认为 False。当提取的值个数与用户输入的目标字段数不一致时： • True：忽略，不进行任何提取处理 • False：尽量匹配前几个字段	bool	否	False	-
mode	新字段的写入模式。默认强制覆盖	string	否	overwrite	-

示例

- **示例1：使用逗号作为分隔符，提取日志中的值。**

  原始日志：

```
{"content": "hello Go,hello Java,hello python"}
```

加工规则：

```
//使用逗号作为分割符号，将content字段值分割为三段，分别对应f1,f2,f3三个字段。
ext_sep("content", "f1, f2, f3", sep=",", quote="", restrict=False, mode="overwrite")
//丢弃content字段
fields_drop("content")
```

加工结果：

```
{"f1":"hello Go","f2":"hello Java","f3":"hello python"}
```

- 示例2：使用 quote 将字符串整体处理。

  原始日志：

```
{"content": " Go,%hello ,Java%,python"}
```

  加工规则：

```
ext_sep("content", "f1, f2",  quote="%", restrict=False)
```

  加工结果：

```
//%hello ,Java%虽然也有逗号，但它作为一个整体，就不参与分隔符提取。
{"content":" Go,%hello ,Java%,python","f1":" Go","f2":"hello ,Java"}
```

- 示例3：当 `restrict=True,` 表示当分割的值的个数和目标字段不同时，函数不执行。

  原始日志：

```
{"content": "1,2,3"}
```

  加工规则：

```
ext_sep("content", "f1, f2", restrict=True)
```

  加工结果：

```
{"content":"1,2,3"}
```

ext_sepstr() 函数

函数定义

基于指定的字符（多字符）提取字段值内容。

语法描述

```
ext_sepstr("源字段名","目标字段1,目标字段2,目标字段...", sep="abc", restrict=False, mode="overwrite")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围

field	待提取的字段名	string	是	-	用户日志中已存在的字段名
output	单个字段名或以英文半角逗号拼接的多个新字段名	string	是	-	-
sep	分隔字符（串） **注意：** 当您的分隔符中包含特殊字符，例如 `. \ * + ? ^ $ | () [] {} - /` 时，需要加上转义符。	string	否	,	-
restrict	默认为 False。当提取的值个数与用户输入的目标字段数不一致时： • True：忽略，不进行任何提取处理 • False：尽量匹配前几个字段	bool	否	False	-
mode	新字段的写入模式。默认强制覆盖	string	否	overwrite	-

示例

原始日志：

```
{"message":"1##2##3"}
```

加工规则：

```
//使用“##”作为分割符，提取键值。
ext_sepstr("message", "f1,f2,f3,f4", sep="##")
```

加工结果：

```
//当目标字段的个数超出了分隔出的值，超出的字段返回"".
{"f1":"1","f2":"2","message":"1##2##3","f3":"3","f4":""}
```

ext_json() 函数

函数定义

从 JSON 中提取字段值。

语法描述

```
ext_json("源字段名",prefix="",suffix="",format="full",exclude_node="不平铺的JSON节点")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	待提取的字段名	string	是	-	-
prefix	新字段前缀	string	否	-	-
suffix	新字段后缀	string	否	-	-
format	• full：字段名格式为全路径（父节点 parent + 连接符 sep + prefix + key 当前节点 + suffix）	string	否	simple	-

	• simple：非全路径（prefix + key 当前节点 + suffix）				
sep	拼接符，用来拼接节点名使用	string	否	#	-
depth	最大展开层级深度，超过此层级的节点不再展开	number	否	100	1-500
expand_array	数组节点是否展开	bool	否	False	-
include_node	基于正则匹配节点名称的白名单	string	否	-	-
exclude_node	基于正则匹配节点名称的黑名单	string	否	-	-
include_path	基于正则匹配节点路径的白名单	string	否	-	-
exclude_path	基于正则匹配节点路径的黑名单	string	否	-	-
retain	保留某些特殊字符不做转义，例如：\n、\t	string	否	-	-
escape	是否需要对数据进行转义，默认值 True，有特殊字符时不能转义	bool	否	True	-

示例

- **示例1：提取全部节点的 KV 成为新字段，提取时不分层级，该例子是多层嵌套。**

  原始日志：

```
{
"data": "{  \"k1\": 100, \"k2\": {  \"k3\": 200,  \"k4\": { \"k5\": 300}}}"
}
```

  加工规则：

```
ext_json("data")
```

  加工结果：

```
{"data":"{  \"k1\": 100, \"k2\": {  \"k3\": 200,  \"k4\": { \"k5\":
300}}}","k1":"100","k3":"200","k5":"300"}
```

- **示例2：sub_field1不参与提取。**

  原始日志：

```
{"content": "{\"sub_field1\":1,\"sub_field2\":\"2\"}"}
```

  加工规则：

```
//exclude_node=subfield1意思就是这个节点不提取
ext_json("content", format="full", exclude_node="sub_field1")
```

  加工结果：

```
{"sub_field2":"2","content":"{\"sub_field1\":1,\"sub_field2\":\"2\"}"}
```

- **示例3：给子节点加上前缀 prefix。**

  原始日志：

```
{"content": "{\"sub_field1\":{\"sub_sub_field3\":1},\"sub_field2\":\"2\"}"}
```

加工规则1：

```
//sub_field2被提取出来时，自动加上前缀udf\_，成为udf\_\_sub\_field2
ext_json("content", prefix="udf_", format="simple")
```

加工结果1：

```
{"content":"{\"sub_field1\":
{\"sub_sub_field3\":1},\"sub_field2\":\"2\"}","udf_sub_field2":"2","udf_sub_sub_field3":"1"}
```

加工规则2：

```
//format=full表示提取的字段名自带层级关系，sub_field2被提取出来时，自动加上它的父节点名称，成为
#content#__sub_field2
ext_json("content", prefix="__", format="full")
```

加工结果2：

```
{"#content#__sub_field2":"2","#content#sub_field1#__sub_sub_field3":"1","content":"{\"sub_field1\":
{\"sub_sub_field3\":1},\"sub_field2\":\"2\"}"}
```

- **示例4：支持特殊字符。**
  原始日志1：

```
{"content": "{\"sub_field1\":1,\"sub_field2\":\"\\n2\"}"}
```

加工规则1：

```
ext_json("content",retain="\n")
```

加工结果1：

```
{"sub_field2":"\\n2","content":"{\"sub_field1\":1,\"sub_field2\":\"\\n2\"}","sub_field1":"1"}
```

原始日志2：

```
{"content": "{\"sub_field1\":1,\"sub_field2\":\"\\n2\\t\"}"}
```

加工规则2：

```
ext_json("content",retain="\n,\t")
```

加工结果2：

```
{"sub_field2":"\\n2\\t","content":"{\"sub_field1\":1,\"sub_field2\":\"\\n2\\t\"}","sub_field1":"1"}
```

- **示例5：自定义是否进行转义。**
  原始日志：

```
{"message":"{\"ip\":\"183.6.104.157\",\"params\":\"
[{\\\"tokenType\\\":\\\"RESERVED30\\\",\\\"otherTokenInfo\\\":
```

```
{\\\"unionId\\\":\\\"123\\\"},\\\"unionId\\\":\\\"adv\\\"}]\"}"}
```

加工规则：

```
ext_json("message", escape=False)
fields_drop("message")
```

加工结果：

```
{"ip":"183.6.104.157", "params":"[{\"tokenType\":\"RESERVED30\",\"otherTokenInfo\":
{\"unionId\":\"123\"},\"unionId\":\"adv\"}]"}
```

ext_json_jmes() 函数

函数定义

从 JSON 中提取字段值。

语法描述

```
ext_json_jmes("源字段名", jmes= "提取JSON的公式", output="目标字段", ignore_null=True, mode="overwrite")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	待提取的字段名	string	是	-	-
jmes	jmes 表达式，请参见 JMESPath **说明：** 如果 jmes 中包含了特殊字符，例如 `. \ * + ? ^ $ | () [] {} - /` 时，需要加上转义符。	string	是	-	-
output	输出字段名，仅支持单个字段	string	是	-	-
ignore_null	是否忽略节点值为 null 的节点，默认 True，忽略 null 字段值，否则当提取结果为 null 时，将输出空字符串	bool	否	True	-
mode	新字段的写入模式。默认强制覆盖	string	否	overwrite	-

示例

- **示例1：提取节点值（叶子结点）。**

  原始日志：

```
{"content": "{\"a\":{\"b\":{\"c\":{\"d\":\"value\"}}}}"}
```

  加工规则：

```
//jmes="a.b.c.d"的意思是取a.b.c.d的值。
ext_json_jmes("content", jmes="a.b.c.d", output="target")
```

  加工结果：

```
{"content":"{\"a\":{\"b\":{\"c\":{\"d\":\"value\"}}}}","target":"value"}
```

- 示例2：提取节点值（非叶子结点）。

  原始日志：

```
{"content": "{\"a\":{\"b\":{\"c\":{\"d\":\"value\"}}}}"}
```

  加工规则：

```
//jmes="a.b.c.d"的意思是取a.b.c的值。
ext_json_jmes("content", jmes="a.b.c", output="target")
```

  加工结果：

```
{"content":"{\"a\":{\"b\":{\"c\":{\"d\":\"value\"}}}}","target":"{\"d\":\"value\"}"}
```

ext_regex() 函数

函数定义

基于正则表达式提取字段值。

语法描述

```
ext_regex(“源字段名”, regex="正则表达式", output=“目标字段1，目标字段2，目标字段.....”, mode="overwrite")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	待提取的字段名	string	是	-	-
regex	regex 表达式，如果包含了特殊字符，用户输入时需要进行转义，否则提示语法错误	string	是	-	-
output	单个字段名或以英文半角逗号拼接的多个新字段名	string	否	-	-
mode	新字段的写入模式。默认强制覆盖	string	否	overwrite	-

示例

- 示例1：匹配数字

  原始日志：

```
{"content": "1234abcd5678"}
```

  加工规则：

```
ext_regex("content", regex="\d+", output="target1,target2")
```

  加工结果：

```
{"target2":"5678","content":"1234abcd5678","target1":"1234"}
```

- 示例2：有命名捕获，自动填充部分字段值。
  原始日志：

```
{"content": "1234abcd"}
```

加工规则：

```
ext_regex("content", regex="(?<target1>\d+)(.*)", output="target2")
```

加工结果：

```
{"target2":"abcd","content":"1234abcd","target1":"1234"}
```

ext_kv() 函数

函数定义

基于两级分割符提取字段值。

语法描述

```
ext_kv("源字段名", pair_sep=r"\s", kv_sep="=", prefix="", suffix="", mode="fill-auto")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	待提取的字段名	string	是	-	-
pair_sep	一级分隔符，分割多个键值对	string	是	-	-
kv_sep	二级分隔符，分割键和值	string	是	-	-
prefix	新字段前缀	string	否	-	-
suffix	新字段后缀	string	否	-	-
mode	新字段的写入模式。默认强制覆盖	string	否	-	-

示例

日志中有两级分割符“|”和“=”。
原始日志：

```
{"content": "a=1|b=2|c=3"}
```

加工规则：

```
ext_kv("content", pair_sep="|", kv_sep="=")
```

加工结果：

```
{"a":"1","b":"2","c":"3","content":"a=1|b=2|c=3"}
```

ext_first_notnull() 函数

函数定义

返回参数中第一个非 null 且非空字符的结果值。

语法描述

```
ext_first_notnull(值1, 值2, ...)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参列表	参与计算的参数或表达式	string	是	-	-

示例

原始日志：

```
{"data1": null, "data2": "", "data3": "first not null"}
```

加工规则：

```
fields_set("result", ext_first_notnull(v("data1"), v("data2"), v("data3")))
```

加工结果：

```
{"result":"first not null","data3":"first not null","data2":"","data1":"null"}
```

ext_grok 函数

函数定义

基于 Grok 提取匹配结果值，Grok 是一种采用组合多个预定义的正则表达式。Grok 公式验证 。

> **注意：**
> 当您的日志中有特殊字符如 . \ * + ? ^ $ | () [] {} - / 时，grok 语法中需要加转义，例如：2024 Nov 9 -trump，对应的 grok 语句：%{YEAR:year} %{MONTH:month} %{MONTHDAY:day} \-%{WORD:trump}，提取结果：{ "year": 2024, "month": "Nov", "day": 9,"President": "trump"}

语法描述

```
ext_grok(字段, grok="", extend="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	字段	string	是	-	-
grok	表达式	string	是	-	-
extend	用户自定义的 GROK 表达式	string	否	-	-

示例

- 示例1：使用默认的 Pattern 提取日期和引号中的字符串。
  原始日志：

```
{"content":"2019 June 24 \"I am iron man\""}
```

加工规则：

```
ext_grok("content", grok="%{YEAR:year} %{MONTH:month} %{MONTHDAY:day} %{QUOTEDSTRING:motto}")
fields_drop("content")
```

加工结果：

```
{"day":"24", "month":"June", "motto":"I am iron man", "year":"2019"}
```

- 示例2：使用自定义 Pattern 提取键值对。

原始日志：

```
{"content":"Beijing-1104,Beijing-Beijing"}
```

加工规则：

```
//extend 参数的作用是对自定义模式进行定义。其中 ID1=%{WORD}-%{INT} 表明 ID1 模式由一个单词、一个连字符和一个整数组成；ID2=%{WORD}-%{WORD} 表示 ID2 模式由一个单词、一个连字符和另一个单词组成。
ext_grok("content", grok="%{ID1:user_id1},%{ID2:user_id2}",extend="ID1=%{WORD}-%{INT},ID2=%{WORD}-%{WORD}")
fields_drop("content")
```

加工结果：

```
{"user_id1":"Beijing-1104", "user_id2":"Beijing-Beijing"}
```

Grok 参考

类型	模式（pattern）	说明
常见模式	EXTRACTJSON	匹配 JSON 类型数据
	CHINAID	匹配中国居民身份证号
	USERNAME	匹配字母、数字和._-组合
	USER	匹配字母、数字和._-组合
	EMAILLOCALPART	匹配邮箱从开头到@字符前内容，例如：123@abc.com，匹配内容为123
	EMAILADDRESS	匹配邮箱
	HTTPDUSER	匹配邮箱或者用户名
	INT	匹配 INT 数字
	BASE10NUM	匹配十进制数
	NUMBER	匹配数字
	BASE16NUM	匹配十六进制数
	BASE16FLOAT	匹配十六进制浮点数
	POSINT	匹配正整数
	NONNEGINT	匹配非负整数
	WORD	匹配字母、数字、下划线
	NOTSPACE	匹配非空格内容

	SPACE	匹配空格
	DATA	匹配换行符
	GREEDYDATA	匹配0个或多个除换行符
	QUOTEDSTRING	匹配引用内容。例如：I am "Iron Man"，会匹配到 Iron Man
	UUID	匹配 UUID
Networking	MAC	匹配 MAC 地址
	CISCOMAC	匹配 CISCOMAC 地址
	WINDOWSMAC	匹配 WINDOWSMAC 地址
	COMMONMAC	匹配 COMMONMAC 地址
	IPV6	匹配 IPV6
	IPV4	匹配 IPV4
	IP	匹配 IPV6或 IPV4
	HOSTNAME	匹配 HOSTNAME
	IPORHOST	匹配 IP 或 HOSTNAME
	HOSTPORT	匹配 IPORHOST 或者 POSTINT
Paths	PATH	匹配 UNIXPATH 或者 WINPATH
	UNIXPATH	匹配 UNIXPATH
	WINPATH	匹配 WINPATH
	URIPROTO	匹配 URI 中的头部分，例如：http://hostname.domain.tld/_astats?application=&inf.name=eth0会匹配到 http
	TTY	匹配 TTY 路径
	URIHOST	匹配 IPORHOST 和 POSINT，例如：http://hostname.domain.tld/_astats?application=&inf.name=eth0，会匹配到 hostname.domain.tld
	URI	匹配内容中的 URI
日期	MONTH	匹配数字或者月份英文缩写或者全拼等格式月份
	MONTHNUM	匹配数字格式月份
	MONTHDAY	匹配月份中的 day
	DAY	匹配星期的英文全拼或者缩写
	YEAR	匹配年份
时间	HOUR	匹配小时
	MINUTE	匹配分钟
	SECOND	匹配秒
	TIME	匹配完整的时间
	DATE_US	匹配 Month-Day-Year 或 Month/Day/Year 形式的日期
	DATE_EU	匹配 Day-Month-Year、Day/Month/Year 或 Day.Month.Year 形式的日期
	ISO8601_TIMEZONE	匹配 ISO8601格式的小时和分钟

版权所有：腾讯云计算（北京）有限责任公司

	ISO8601_SECOND	匹配 ISO8601格式的秒钟
	TIMESTAMP_ISO8601	匹配 ISO8601格式的时间
	DATE	匹配 US 或 EU 格式的时间
	DATESTAMP	匹配完整日期和时间
	TZ	匹配 UTC
	DATESTAMP_RFC822	匹配 RFC822格式时间
	DATESTAMP_RFC2822	匹配 RFC2822格式时间
	DATESTAMP_OTHER	匹配其他格式时间
	DATESTAMP_EVENTLOG	匹配 EVENTLOG 格式的时间
	HTTPDERROR_DATE	匹配 HTTPDERROR 格式的时间
SYSLOG	SYSLOGTIMESTAMP	匹配Syslog格式的时间
	PROG	匹配 program内容
	SYSLOGPROG	匹配 program 和 pid 内容
	SYSLOGHOST	匹配 IPORHOST
	SYSLOGFACILITY	匹配 facility
	HTTPDATE	匹配日期时间
LOGFORMATL	LOGFORMAT	匹配 Syslog，默认 TraditionalFormat 格式的 Syslog 日志
	COMMONAPACHELOG	匹配 commonApache 日志
	COMBINEDAPACHELOG	匹配组合 Apache 日志
	HTTPD20_ERRORLOG	匹配 HTTPD20日志
	HTTPD24_ERRORLOG	匹配 HTTPD24日志
	HTTPD_ERRORLOG	匹配 HTTPD 日志
LOGLEVEL	LOGLEVEL	匹配 Log 的 Level，例如 warn、debug、info 等

富化函数

最近更新时间：2025-06-24 18:54:02

t_table_map 函数

函数定义

与目标表格进行映射，根据输入的字段名称返回字段值，可简单理解为日志和维度表关联。

语法描述

```
t_table_map(data, field, output_fields, missing=None, mode="fill-auto")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	目标表格（维度表）	table	是	-	-
field	日志中映射到表格的源字段。如果日志中不存在对应字段，则不进行任何操作。支持 String 和 String List	any	是	-	-
output_fields	映射后的字段。例如["province", "pop"]。支持 String 和 String List	any	是	-	-
missing	无匹配字段时，将该参数的取值赋给输出字段 output_fields	String	否	-	-
mode	字段的覆盖模式。默认为 fill-auto	String	否	fill-auto	-

示例

MySQL 中新建一个表 test，添加如下数据：

user_id	gameid	game
3	123	wangzhe
2	456	majiang

原始日志：

```
[
    {
        "user_id": 1
    },
    {
        "user_id": 3
    }
]
```

加工规则：

res_rds_mysql 用于从 MySQL 中获取维度表数据。

```
//在控制台将外部数据 MySQL 的别名配置为 hm，mysql 的 db 为 test222，表名为 test
//从 MySQL 拉取全部数据，使用 t_table_map 函数完成维度表关联

t_table_map(
    res_rds_mysql(alias="hm",database="test222",sql="select * from test"),
    "user_id",
    ["gameid", "game"]
```

```
)
```

加工结果：

```
[
{
"user_id":"1"
},
{
"game":"wangzhe",
"gameid":"123",
"user_id":"3"
}
]
```

enrich_table 函数

函数定义

使用 CSV 结构数据对日志中的字段进行匹配，当值相同时，可以将 CSV 中的其他字段和值添加到源日志中。

语法描述

```
enrich_table("csv 源数据", "csv 富化字段", output="目标字段1，目标字段2，目标字段....", mode="overwrite")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	输入的 CSV 数据，第一行为列名，其余行对应值。例如：region,count\nbj, 200\ngz, 300	String	是	-	-
fields	待匹配列名称。CSV 中的字段名称，与实际日志中同名的字段进行匹配。单个字段名或以英文半角逗号拼接的多个新字段名	String	是	-	-
output	输出字段列表，单个字段名或以英文半角逗号拼接的多个新字段名	String	是	-	-
mode	新字段的写入模式。默认强制覆盖	String	否	overwrite	-

示例

原始日志：

```
{"region": "gz"}
```

加工规则：

```
enrich_table("region,count\nbj,200\ngz,300", "region", output="count")
```

加工结果：

```
{"count":"300","region":"gz"}
```

enrich_dict 函数

函数定义

使用 Dict 结构对日志中的字段值进行匹配，当指定的字段的值和 Dict 中的 Key 相同时，将此 Key 对应的 Value 赋值给日志中的另一字段。

语法描述

```
enrich_dict(“JSON 字典”, "源字段名", output=目标字段, mode="overwrite")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	输入的 dict 数据，这里必须是 JSON 对象的转义字符串，例如：{\"200\":\"SUCCESS\",\"500\":\"FAILED\"}	String	是	-	-
fields	待匹配字段名称。当 dict 中的 key 和指定字段对应的值相同时，匹配成功。单个字段名或以英文半角逗号拼接的多个新字段名	String	是	-	-
output	目标字段列表。匹配成功后，将 dict 中对应的 value 写入到目标字段列表。单个字段名或以英文半角逗号拼接的多个新字段名	String	是	-	-
mode	新字段的写入模式。默认强制覆盖	String	否	overwrite	-

示例

原始日志：

```
{"status": "500"}
```

加工规则：

```
enrich_dict("{\"200\":\"SUCCESS\",\"500\":\"FAILED\"}", "status", output="message")
```

加工结果：

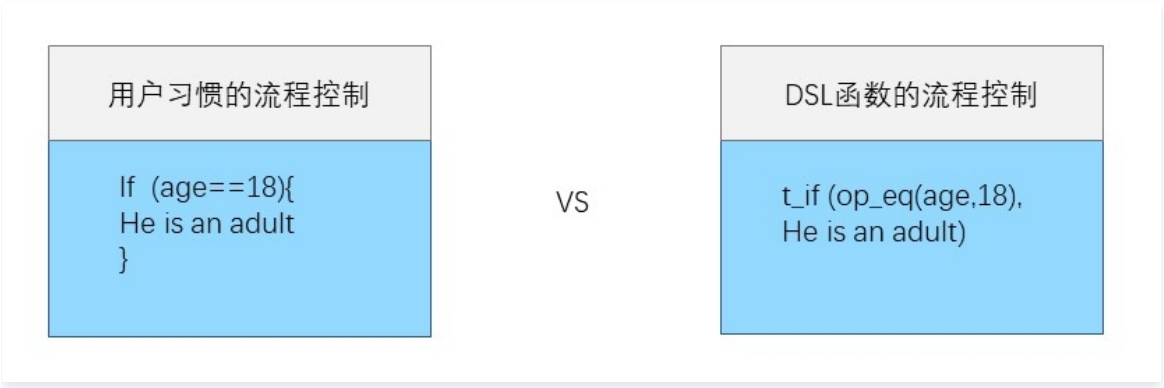
```
{"message":"FAILED","status":"500"}
```

流程控制

最近更新时间：2024-08-15 15:04:21

简介

流程控制的逻辑在常用的编程语言和 DSL 函数中是一样的，区别在于写法不同。如下图所示：



compose 函数

函数定义

组合操作函数，类似于分支代码块的组合能力，可以组合多个操作函数，并按顺序执行，可以结合分支、输出函数使用。

语法描述

```
compose(函数1,函数2, ...)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参，函数	参数必须为返回类型为 LOG 的函数	String	至少一个函数参数	-	-

示例

- 示例1：顺序调用函数，先执行 enrich 函数，然后是 fields_set 函数。
  原始日志：

  ```
  {"status": "500"}
  ```

  加工规则：

  ```
  //1. enrich函数：使用dict中的数据，对原始日志进行富化，status字段值为500，富化出新字段message，值为Failed。
  //2. fields_Set函数，新增一个字段new，赋值1。
  compose(enrich_dict("{\"200\":\"SUCCESS\",\"500\":\"FAILED\"}", "status", output="message"),
  fields_set("new", 1))
  ```

  加工结果：

  ```
  //最后结果是三个字段，如下：
  {"new":"1","message":"FAILED","status":"500"}
  ```

- 示例2
  原始日志：

```
{"status": "500"}
```

加工规则：

```
compose(fields_set("new", 1))
```

加工结果：

```
{"new":"1","status":"500"}
```

- 示例3
  原始日志：

```
{"condition1": 0,"condition2": 1, "status": "500"}
```

加工规则：

```
t_if_else(v("condition2"), compose(fields_set("new", 1),fields_set("target1","1")),
fields_set("target2","0"))
```

加工结果，target 输出：

```
{"new":"1","condition1":"0","condition2":"1","status":"500","target1":"1"}
```

t_if 函数

函数定义

对符合条件的日志，进行相应的函数处理，否则不进行任何处理。

语法描述

```
t_if(条件1, 函数1)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
condition	返回值为 BOOL 类型的函数表达式	BOOL	是	-	-
function	返回值为 LOG 类型的函数表达式	string	是	-	-

示例

- 示例1原始日志：

```
{"condition": 1, "status": "500"}
```

加工规则：

```
t_if(True, fields_set("new", 1))
```

加工结果：

```
{"new":"1","condition":"1","status":"500"}
```

- 示例2
  原始日志：

```
//如果 condition 字段的值为真，新增一个字段 new 并赋值1。
{"condition": 1, "status": "500"}
```

  加工规则：

```
t_if(v("condition"), fields_set("new", 1))
```

  加工结果：

```
{"new":"1","condition":"1","status":"500"}
```

t_if_not 函数

函数定义

对不符合条件的日志，进行相应的函数处理，否则不进行任何处理。

语法描述

```
t_if_not(条件1, 函数1)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
condition	返回值为 BOOL类型的函数表达式	BOOL	是	-	-
function	返回值为 LOG 类型的函数表达式	string	是	-	-

示例

原始日志：

```
{"condition": 0, "status": "500"}
```

加工规则：

```
t_if_not(v("condition"), fields_set("new", 1))
```

加工结果：

```
{"new":"1","condition":"0","status":"500"}
```

t_if_else 函数

函数定义

基于条件判断，分别进行不同的函数处理。

语法描述

```
t_if_else(条件1, 函数1, 函数2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
condition	返回值为 BOOL类型的函数表达式	BOOL	是	-	-
function	返回值为 LOG 类型的函数表达式	string	是	-	-
function	返回值为 LOG 类型的函数表达式	string	是	-	-

示例

原始日志：

```
{"condition": 1, "status": "500"}
```

加工规则：

```
t_if_else(v("condition"), fields_set("new", 1), fields_set("new", 2))
```

加工结果：

```
{"new":"1","condition":"1","status":"500"}
```

t_switch 函数

函数定义

基于多分支条件，分别进行不同的函数处理，如果存在不符合所有条件的数据，将被丢弃。

语法描述

```
t_switch(条件1, 函数1, 条件2, 函数2, ...)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参，交替的条件和函数表达式作为参数	参考 t_if 函数，类似多个 t_if 组合	-	-	-	-

示例

原始日志：

```
{"condition1": 0,"condition2": 1, "status": "500"}
```

加工规则：

```
//condition1字段值为真时，新增字段new并赋值1，由于此处字段值返回假，因此不会新增"new":"1",第二个条件返回真，所以新增了"new":"2"
t_switch(v("condition1"), fields_set("new", 1), v("condition2"), fields_set("new", 2))
```

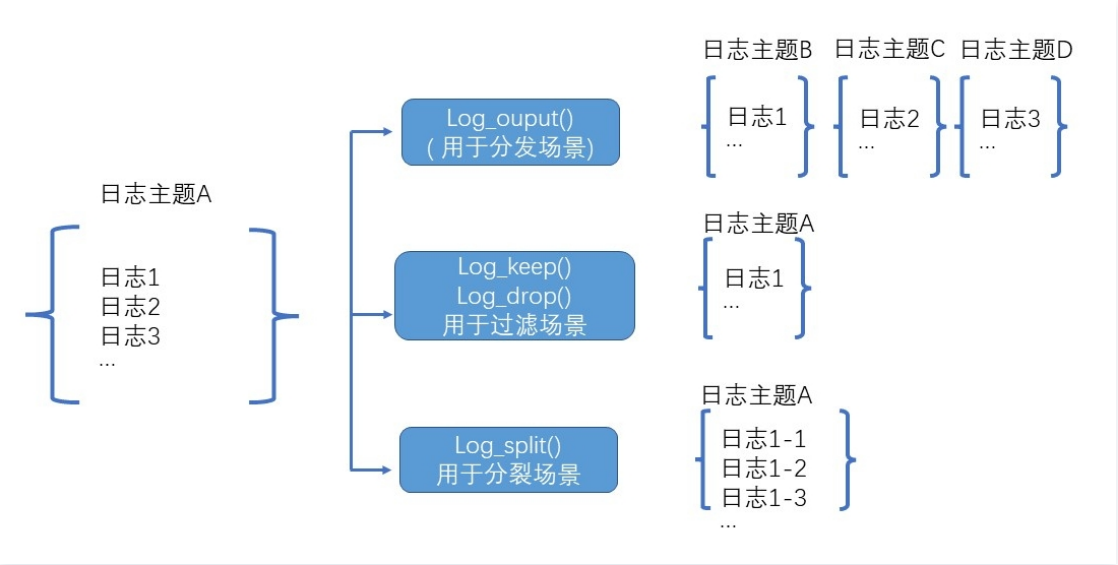
加工结果：

```
{"new":"2","condition1":"0","condition2":"1","status":"500"}
```

行处理函数

最近更新时间：2025-02-05 17:42:13

对一行日志进行处理，包括过滤、分发、分裂等函数。



log_output 函数

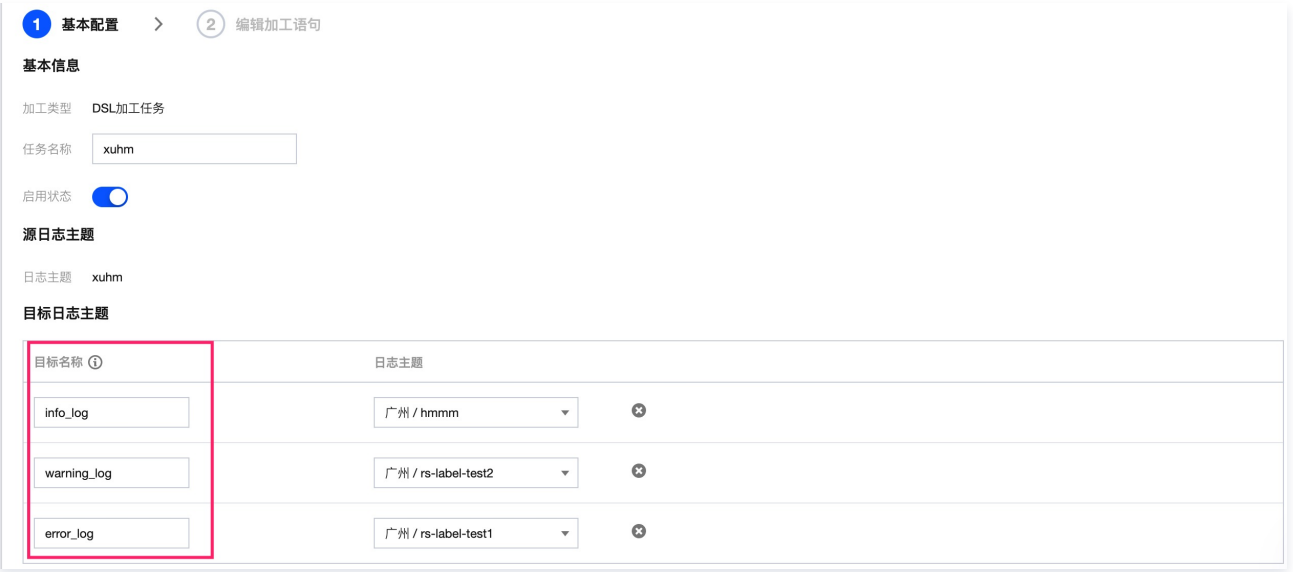
函数定义

输出到指定的目标主题。可以配合分支条件使用，也可以单独使用。

语法描述

```
log_output(别名 Alias)
```

别名在配置加工任务时定义，如下图所示：



参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
alias	目标主题的别名	string	是	-	-

示例

按照 loglevel 字段值为 waring/info/error 的情况，分发到三个不同的日志主题中。
原始日志：

```
[
  {
    "loglevel": "warning"
  },
  {
    "loglevel": "info"
  },
  {
    "loglevel": "error"
  }
]
```

加工规则：

```
//按照loglevel字段值为waring/info/error的情况，分发到三个不同的日志主题中。
t_switch(regex_match(v("loglevel"),regex="info"),log_output("info_log"),regex_match(v("loglevel"),regex="warning"),log_output("warning_log"),regex_match(v("loglevel"),regex="error"),log_output("error_log"))
```

加工结果：如下图所示

	行号	加工信息	加工日志
▶	1	原始行号：1 目标日志主题：rs-label-test2 加工结果：成功	loglevel: warning
▶	2	原始行号：2 目标日志主题：hmmm 加工结果：成功	loglevel: info
▶	3	原始行号：3 目标日志主题：rs-label-test1 加工结果：成功	loglevel: error

log_auto_output 函数

函数定义

将日志输出到动态的目标主题。例如您需要根据日志字段"pd"的值，动态创建多个目标日志主题，并将对应的日志分发至目标日志主题。假设 pd 的值为"CLB "、"Ckafka"、"COS"、"CDN"，使用该函数，将会动态创建出名为"CLB "、"Ckafka"、"COS"、"CDN"的目标日志主题，并将相关的日志写入对应的主题，同时，您可对这些新建主题的索引类型、存储周期进行配置。

语法描述

```
log_auto_output(topic_name="", logset_name="", index_options="", period=3, storage_type=" ",hot_period=0)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数说明
topic_name	日志主题名称	string	y	-	• 参数 topic_name 中包含"|"，"|"将在生成的主题名称中被去除。 • 参数 topic_name 长度超出250字符，生成的日志主题名称仅有前250个字符，超出的会被截断。
logset_name	日志集名称	string	y	-	-
index_options	• all_index：开启键值、全文索引 • no_index：不开启索引 • content_index：开启全文 • key_index：开启键值索引	string	n	all_index	如果 storage_type=cold，即低频存储，则 all_index 和 key_index 不会生效，即低频存储不支持键值索引。
period	存储周期，一般取值范围是1 - 3600 天，3640 代表永久存储	number	n	3	1 - 3600天。
storage_type	日志主题的存储类型，可选值 • hot：标准存储 • cold：低频存储	string	n	hot	当为 cold 时，period 最小为7天。
hot_period	• 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	number	n	0	-
tag_dynamic	为日志主题添加动态标签，和 extract_tag() 函数配合使用，用于从日志字段中提取标签的 KV，例如：tag_dynamic=extract_tag(v("pd"),v("env"),v("team"), v("person"))	string	n	-	和 tag_static 一起不超过10对标签。
tag_static	为日志主题添加静态标签，例如：tag_static="Ckafka:test_env,developer_team:MikeWang"	string	n	-	和 tag_dynamic 一起不超过10对标签示例。

注意：

- 修改 log_auto_output() 函数中的参数，并重启任务之后，仅对新生成的主题生效，已生成的主题不受影响。如您需要对已经生成日志集、日志主题进行修改，请前往 日志主题列表 或调用 API 进行修改。
- 在不修改 log_auto_output() 函数的前提下，您修改数据加工任务动态创建的主题名称、兜底日志集名称、兜底日志主题名称之后，数据分发逻辑不受影响。例如加工任务已动态创建日志主题“Nginx”，您将其改名为“Nginx001”，数据还是会写入这个“Nginx001”主题，而不会新创建一个叫做“Nginx”的日志主题。
- 删除数据加工任务（动态创建主题），不会删除已创建的日志集和日志主题。

示例

原始日志：

```
[
    {
        "pd": "CLB",
        "dateTime": "2023-05-25T00:00:26.579"
    },
    {
        "pd": "Ckafka",
        "time": "2023-05-25T18:00:55.350+08:00"
```

```
    },
    {
        "pd": "COS",
        "time": "2023-05-25T00:06:20.314+08:00"
    },
    {
        "pd": "CDN",
        "time": "2023-05-25T00:03:52.051+08:00"
    }
]
```

加工规则：

```
log_auto_output(v("pd"),"我的日志集",index_options="content_index", period=3)
```

加工结果：自动生成了四个日志主题，分别是"CLB "、"Ckafka"、"COS"和"CDN"，日志集名称为"我的日志集"，如下图所示：



特殊情况处理

创建日志主题

特殊场景	处理逻辑
参数 topic_name 获取失败或者为""（空字符串），该行日志将进入名称为"null"的日志主题	例如指定 app 字段的值为 topicname，即日志主题名称，但某行日志中没有 app 字段，或者其值为" "，那么该行日志将写入名称为“null”的日志主题，“null”主题所属日志集，是本函数的日志集参数中指定的日志集。 **说明：** 如果您使用了字符串拼接来指定日志主题名称，例如 `log_auto_output( str_join("-", "myPrefix", v("app")), "myLogSet",period=14,storage_type="hot",hot_period=10)` ，那么当 app 不存在或者值为" "时，日志将写入名称为“myprefix”的日志主题。
重名主题（同一日志集下）	已存在名为“Nginx”的日志主题，动态创建任务根据规则，也要创建名为“Nginx”的日志主题时，会加上 CreateByCLSETL 后缀，创建出的日志主题名为“Nginx-CreateByETL”。

创建日志集

特殊场景	处理逻辑
重名日志集（同一地域）	• 如您手动创建名为“myLogSet”的日志集，动态创建任务根据函数规则也要创建名为“myLogSet”的日志集时，会自动加上 CreateByCLSETL 为后缀。 • 如果自动创建任务1，创建出来的日志集名称为“myLogSet”，自动创建任务2同样定义了日志集名称为“myLogSet”，此时自动创建任务2会使用同一个日志集名称“myLogSet”作为目标日志集，即不会再自动创建带有 CreateByCLSETL 后缀的日志集。

注意：
日志集和日志主题名称，不区分英文大小写，abc 和 Abc 在 CLS 侧代表同一个资源。

日志主题添加标签

特殊场景	处理逻辑
动态标签的 KV 值获取失败	例如指定 A 字段的值为标签的 Key，B 字段的值为标签的 Value，但某行日志没有 A 字段或者 B 字段，或者 A 字段 B 字段的值为空，标签添加失败。
标签的 KV 值前后有空格	标签添加失败。
动态标签和静态标签的总数超过10对	动态标签优先级较高，如添加完动态标签后，还不足10对，补充静态标签，但总个数不超过10对。
标签 KV 的长度超出限制	标签的 key 长度大于127，value 的长度大于255，标签添加失败。

说明：
如果您在 POC 时需要反复重建同名加工任务，其流程如下：
1. 先删除老的数据加工任务，否则还会持续生成日志主题。
2. 删除兜底主题 auto_undertake_topic_${etl-task-name}，如果连续3次创建同名任务，超限兜底主题会创建失败，进而阻塞任务正常启动。

如果 POC 的数据是同一套，不是实时数据，建议删除老任务已经生成的日志集和日志主题。

log_split 函数

函数定义

使用分隔符结合 jmes 表达式，对特定字段进行拆分，拆分结果分裂为多行日志。

语法描述

```
log_split(字段名, sep=",", quote="\", jmes="", output="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	待提取的字段名	string	是	-	-
sep	分隔符	string	否	,	任意单字符
quote	将值包括起来的字符	string	否	-	-
jmes	Jmes 表达式，详情请参见 JMESPath **说明：** 如果 jmes 中包含了特殊字符，例如 `. \ * + ? ^ $ | () [] {} - /` 时，需要加上转义符。	string	否	-	-
output	单个字段名	string	是	-	-

示例

- 示例1：字段中有多个值的日志分裂

```
{"field": "hello Go,hello Java,hello python","status":"500"}
```

加工规则：

```
//使用分割符“，”，将日志分隔成三条。
log_split("field", sep=",",  output="new_field")
```

加工结果：

```
{"new_field":"hello Go","status":"500"}
{"new_field":"hello Java","status":"500"}
{"new_field":"hello python","status":"500"}
```

- 示例2：利用 JMES 对日志进行分裂。

```
{"field": "{\"a\":{\"b\":{\"c\":{\"d\":\"a,b,c\"}}}}", "status": "500"}
```

加工规则：

```
//a.b.c.d节点的值为“a,b,c”
log_split("field", jmes="a.b.c.d",  output="new_field")
```

加工结果：

```
{"new_field":"a","status":"500"}
{"new_field":"b","status":"500"}
{"new_field":"c","status":"500"}
```

- 示例3：包含有 JSON 数组的日志分裂。

```
{"field": "{\"a\":{\"b\":{\"c\":{\"d\":[\"a\",\"b\",\"c\"]}}}}", "status": "500"}
```

加工规则：

```
log_split("field", jmes="a.b.c.d",  output="new_field")
```

加工结果：

```
{"new_field":"a","status":"500"}
{"new_field":"b","status":"500"}
{"new_field":"c","status":"500"}
```

log_drop 函数

函数定义

丢弃符合条件的日志。

语法描述

```
log_drop(条件1)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
condition	值为 bool 类型的函数表达式	bool	是	-	-

示例

丢弃 status=200的日志，其余保留。
原始日志：

```
[
    {
        "field": "a,b,c",
        "status": "500"
    },
    {
        "field": "a,b,c",
        "status": "200"
    }
]
```

加工规则：

```
log_drop(op_eq(v("status"), 200))
```

加工结果：

```
{"field":"a,b,c","status":"500"}
```

log_keep 函数

函数定义

保留符合条件的日志。

语法描述

```
log_keep(条件1)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
condition	值为 bool 类型的函数表达式	bool	是	-	-

示例

保留 status=500的日志，其余丢弃。
原始日志：

```
[
    {
        "field": "a,b,c",
        "status": "500"
    },
```

```
    {
        "field": "a,b,c",
        "status": "200"
    }
]
```

加工规则：

```
log_keep(op_eq(v("status"), 500))
```

加工结果：

```
{"field":"a,b,c","status":"500"}
```

log_split_jsonarray_jmes 函数

函数定义

将日志根据 Jmes 语法将 JSON 数组拆分和展开。

语法描述

```
log_split_jsonarray_jmes("field", jmes="items", prefix="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	待提取的字段名	string	是	-	-

示例

- 示例1
  原始日志：

  ```
  {"common":"common","result":"{\"target\":[{\"a\":\"a\"},{\"b\":\"b\"}]}"}
  ```

  加工规则：

  ```
  log_split_jsonarray_jmes("result",jmes="target")
  fields_drop("result")
  ```

  加工结果：

  ```
  {"common":"common", "a":"a"}
  {"common":"common", "b":"b"}
  ```

- 示例2
  原始日志：

  ```
  {"common":"common","target":"[{\"a\":\"a\"},{\"b\":\"b\"}]"}
  ```

  加工规则：

  ```
  log_split_jsonarray_jmes("target",prefix="prefix_")
  ```

```
fields_drop("target")
```

加工结果：

```
{"prefix_a":"a", "common":"common"}
{"prefix_b":"b", "common":"common"}
```

字段处理函数

最近更新时间：2025-06-24 15:47:32

简介

字段处理函数，日志中的字段进行处理，包括取值、增删查改等。

v 函数

函数定义

获取字段值，返回对应字符串。

语法描述

```
v(字段名)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	字段名	string	是	-	-

示例

获取 "message" 字段的值，将它赋值给一个新字段 "new_message"。
原始日志：

```
{"message": "failed", "status": "500"}
```

加工规则：

```
fields_set("new_message", v("message"))
```

加工结果：

```
{"message": "failed", "new_message": "failed","status": "500"}
```

fields_drop 函数

函数定义

根据字段名进行匹配，删除匹配到的字段。

语法描述

```
fields_drop(字段名1, 字段名2, ..., regex=False,nest=False)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参，字段名或字段名的正则表达式	可变参，字段名或字段名的正则表达式	string	是	-	-
regex	是否启用正则表达式，匹配模式为全匹配	bool	否	False	-
nest	是否是嵌套字段	bool	否	False	-

版权所有：腾讯云计算（北京）有限责任公司

示例

- 示例1：删除字段名为 "field" 的字段。

  原始日志：

  ```
  {"field": "a,b,c", "status": "500"}
  ```

  加工规则：

  ```
  fields_drop("field")
  ```

  加工结果：

  ```
  {"status":"500"}
  ```

- 示例2：嵌套字段的处理。

  原始日志：

  ```
  {"condition":"{\"a\":\"aaa\", \"c\":\"ccc\", \"e\":\"eee\"}","status":"500"}
  ```

  加工规则：

  ```
  //nest=True,表示该字段是嵌套字段，将condition.a和condition.c丢弃后，只剩下condition.e字段。
  t_if(if_json(v("condition")), fields_drop("condition.a", "condition.c", nest=True))
  ```

  加工结果：

  ```
  {"condition":"{\"e\":\"eee\"}","status":"500"}
  ```

- 示例3：嵌套字段的处理。

  原始日志：

  ```
  {"App": "thcomm","Message": "{\"f_httpstatus\": \"200\",\"f_requestId\": \"2021-11-09
  08:40:17.832\tINFO\tservices/http_service.go:361\tbb20ac02-fcbc-4a56-b1f1-4064853b79da\",\"f_url\":
  \"wechat.wecity.qq.com/trpcapi/MbpsPaymentServer/scanCode\"}"}
  ```

  加工规则：

  ```
  //nest=True,表示该字段是嵌套字段，将Message.f_requestId", Message.f_url丢弃后，只剩下字f_httpstatus段。
  t_if(if_json(v("Message")), fields_drop("Message.f_requestId", "Message.f_url", nest=True))
  ```

  加工结果：

  ```
  {"App":"thcomm","Message":"{\"f_httpstatus\":\"200\"}"}
  ```

fields_keep 函数

函数定义

根据字段名进行匹配，保留匹配到的字段。

语法描述

```
fields_keep(字段名1, 字段名2, ..., regex=False)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参，字段名或字段名的正则表达式	可变参，字段名或字段名的正则表达式	string	是	-	-
regex	是否启用正则表达式，匹配模式为全匹配	bool	否	False	-

示例

保存字段名为 "field" 的字段，其余字段丢弃。
原始日志：

```
{"field": "a,b,c", "status": "500"}
```

加工规则：

```
fields_keep("field")
```

加工结果：

```
{"field":"a,b,c"}
```

fields_pack 函数

函数定义

根据正则表达式来匹配字段名，并将匹配到的字段打包到新的字段，新字段值使用 JSON 格式进行组织。

语法描述

```
fields_pack(目标字段名, include=".*", exclude="", drop_packed=False)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
output	打包后的新字段名称	string	是	-	-
include	包含字段名称的正则	string	否	-	-
exclude	排除字段名称的正则	string	否	-	-
drop_packed	是否删除原有被打包的字段	bool	否	False	-

示例

原始日志：

```
{"field_a": "a,b,c","field_b": "abc", "status": "500"}
```

加工规则：

```
fields_pack("new_field","field.*", drop_packed=False)
```

加工结果：

```
{"new_field":"
{\"field_a\":\"a,b,c\",\"field_b\":\"abc\"}","field_a":"a,b,c","field_b":"abc","status":"500"}
```

fields_set 函数

函数定义

用来设置字段值，或者增加新字段。

语法描述

```
fields_set(字段名1, 字段值1, 字段名2, 字段值2, mode="overwrite")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参	key、value 交替的参数列表	string	-	-	-
mode	字段覆写模式	string	否	overwrite	-

示例

- 示例1：将级别从 “Info” 修改为 “Waring” 。

  原始日志：

  ```
  {"级别": "Info"}
  ```

  加工规则：

  ```
  fields_set("级别", "Warning")
  ```

  加工结果：

  ```
  {"级别":"Warning"}
  ```

- 示例2：新增两个字段，"new"和"new2"。

  原始日志：

  ```
  {"a": "1", "b": "2", "c": "3"}
  ```

  加工规则：

  ```
  fields_set("new", v("b"), "new2", v("c"))
  ```

  加工结果：

  ```
  {"a":"1","b":"2","c":"3","new":"2","new2":"3"}
  ```

fields_rename 函数

函数定义

字段重命名。

语法描述

```
fields_rename(字段名1, 新字段名1, 字段名2, 新字段名2, regex=False)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参	旧字段名、新字段名交替的参数列表	string	-	-	-
regex	字段名是否启用正则匹配，启用使用正则匹配旧字段名，不启用则是相等匹配	bool	否	False	-

示例

原始日志：

```
{"regieeen": "bj", "status": "500"}
```

加工规则：

```
fields_rename("reg.*", "region", regex=True)
```

加工结果：

```
{"region":"bj","status":"500"}
```

has_field 函数

函数定义

字段存在时，返回 True，否则返回 False。

语法描述

```
has_field(字段名)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	字段名	string	是	-	-

示例

原始日志：

```
{"regiooon": "bj", "status": "500"}
```

加工规则：

```
t_if(has_field("regiooon"), fields_rename("regiooon", "region"))
```

加工结果：

```
{"region":"bj","status":"500"}
```

not_has_field 函数

函数定义

字段不存在时，返回 True，否则返回 False。

语法描述

```
not_has_field(字段名)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	字段名	string	是	-	-

示例

原始日志：

```
{"status": "500"}
```

加工规则：

```
t_if(not_has_field("message"), fields_set("no_message", true))
```

加工结果：

```
{"no_message":"true","status":"500"}
```

extract_tag 函数

函数定义

为动态生成的主题打标签。从日志字段中提取标签值，并将这些标签值作为动态生成主题的标签。

> **注意：**
> - 仅可用于 log_auto_output() 函数 之后。
> - 其中静态标签和动态标签的总数不超过10对。
>
> 例如这条语句：`log_auto_output(v("pd"),"我的日志集",index_options="content_index", period=3, tag_static ="Ckafka:test_env,developer:MikeWang", tag_dynamic =extract_tag("pd",v("pd"),"team", v("team")))`
> 使用了 tag_Static 静态标签和 tag_dynamic 动态标签。

语法描述

```
extract_tag(标签名1，标签值1，标签名2，标签值2....)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
标签名	标签名	string	是	-	-
标签值	标签值	string	是	-	-

示例

原始日志：

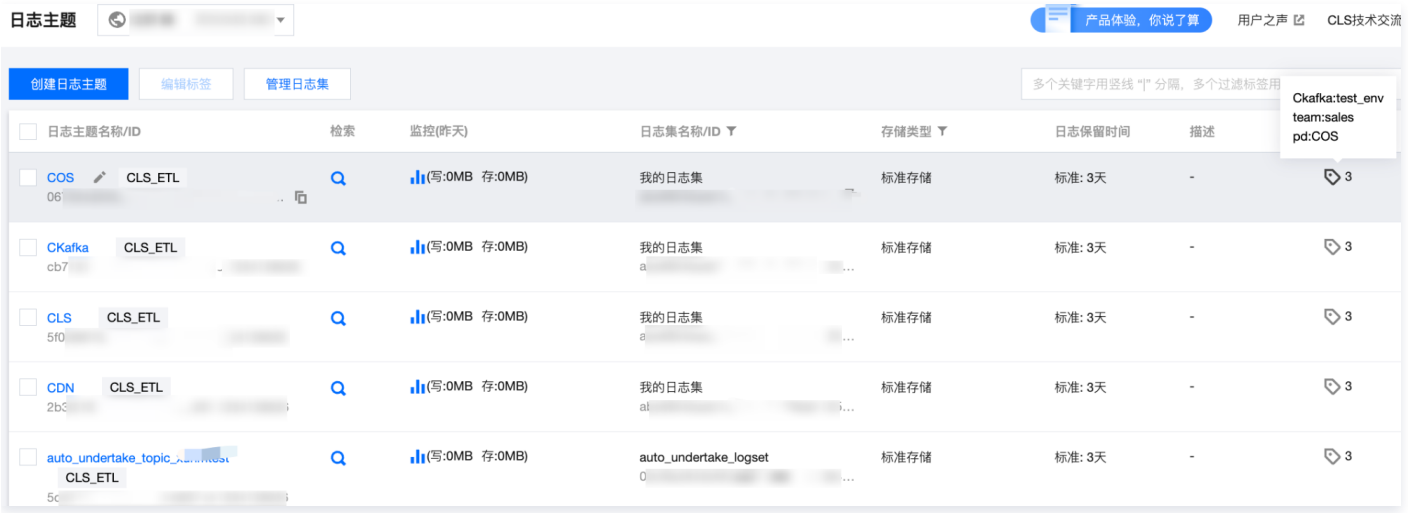
```
[
{"pd": "CDN", "team": "test"},
{"pd": "CLS", "team": "product"},
{"pd": "COS", "team": "sales"},
{"pd": "CLS", "team": "test"},
{"pd": "CKafka", "team": "product"}
]
```

说明：
同一个日志主题，相同的标签名，仅最早的标签值生效。该示例中，名称为 CLS 的日志主题，team:produc 是最早的标签值，该标签生效，后续的 team:test 标签不生效，因为它们的标签名相同。

加工规则：

```
log_auto_output(v("pd"),"我的日志集",index_options="content_index",
period=3,tag_static="Ckafka:test_env",tag_dynamic=extract_tag("pd",v("pd"),"team", v("team")))
```

加工结果：
生成了名为 COS、CKafka、CLS、CDN 的日志主题，并为这几个日志主题打标签。语句中有静态标签 tag_static 1对，动态标签 tag_dynamic 2对，因此都生成了3对标签。



字典和列表函数

最近更新时间：2025-06-24 15:47:32

字典函数

json_add 函数

函数定义

json 中新增节点。

语法描述

```
json_add(key,data)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
key	json 对应的 key	string	是	-	-
data	补充新的节点	dict	是	-	-

示例

原始日志：

```
{"content": "{\"a\":{\"b\":{\"c\":\"cc\"}}}"}
```

加工规则：

```
json_add("content", {"a":{"b":{"d":"dd"}}})
```

加工结果：

```
{"content": "{\"a\":{\"b\":{\"c\":\"cc\",\"d\":\"dd\"}}}"}
```

json_edit 函数

函数定义

增加/修改/删除字典中的键值对，返回 dict。

语法描述

```
json_edit("result", path="", key="",value="333",index=1,mode="edit")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
field	嵌套 json 对应的 key	string	是	-	-
path	需要删除或修改目标字段对应的目录，删除或修改的字段为嵌套 json 的第一层级时，不填；对数组元素操作时填写至数组对应的 key，支持 JMES 语法	string	否	-	-
key	需要删除或修改的目标字段，当对数组元素操作时，无需填写	string	否	-	-

版权所有：腾讯云计算（北京）有限责任公司

value	需要设定的新值，修改值时才需要填写	string	否	-	-
index	对数组操作时填写该字段，数组元素从1开始	number	否	0	-
mode	模式。修改：edit，删除：move，默认 move	string	否	move	-

示例1：修改指定节点的值

原始日志：

```
{"content":"{\"p9\":[\"0.0\",\"0.0\"],\"p18\":\"CN\"}","time":"1650440364"}
```

加工规则：

```
json_edit("content", path="", key="p18", value="hello", mode="edit")
```

加工结果：

```
{"content":"{\"p9\":[\"0.0\",\"0.0\"],\"p18\":\"hello\"}", "time":"1650440364"}
```

示例2：修改指定节点的值（多层级）

原始日志：

```
{"content":"{\"p9\":[\"0.0\",\"0.0\"],\"p18\":\"CN\",\"info\":{\"province\":\"hubei\",\"geo\":
{\"long\":\"111\",\"lati\":\"222\"}}}","time":"1650440364"}
```

加工规则：

```
json_edit("content", path="info.geo", key="long", value="333", mode="edit")
```

加工结果：

```
{"content":"{\"p9\":[\"0.0\",\"0.0\"],\"p18\":\"CN\",\"info\":{\"province\":\"hubei\",\"geo\":
{\"long\":\"333\",\"lati\":\"222\"}}}","time":"1650440364"}
```

示例3：删除节点

原始日志：

```
{"content":"{\"p9\":[\"0.0\",\"0.0\"],\"p18\":\"CN\",\"info\":{\"province\":\"hubei\",\"geo\":
{\"long\":\"111\",\"lati\":\"222\"}}}","time":"1650440364"}
```

加工规则：

```
json_edit("content", path="", key="p18", mode="move")
```

加工结果：

```
{"content":"{\"p9\":[\"0.0\",\"0.0\"],\"info\":{\"province\":\"hubei\",\"geo\":
{\"long\":\"111\",\"lati\":\"222\"}}}","time":"1650440364"}
```

示例4：删除节点（数组元素）

原始日志：

```
{"content":"{\"p9\":[\"0.0\",\"0.0\"],\"p18\":\"CN\",\"info\":{\"province\":[\"hello\",\"world\"],\"geo\":{\"long\":[\"1.0\",\"2.0\"],\"lati\":\"222\"}}}","time":"1650440364"}
```


加工规则：

```
json_edit("content", path="info.province", index=1, mode="move")
```

加工结果：


```
{"content":"{\"p9\":[\"0.0\",\"0.0\"],\"p18\":\"CN\",\"info\":{\"province\":[\"world\"],\"geo\":{\"long\":[\"1.0\",\"2.0\"],\"lati\":\"222\"}}}","time":"1650440364"}
```


json_select 函数

函数定义

通过 JMES表达式，提取 JSON 字段值，并返回 Jmes 提取的节点值（String）。

语法描述

```
json_select(data, jmes="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值，可以通过其他函数提取字段值	string	是	-	-
jmes	JMES 表达式	string	是	-	-

示例

原始日志：

```
{"field": "{\"a\":{\"b\":{\"c\":{\"d\":\"success\"}}}}", "status": "500"}
```

加工规则：

```
fields_set("message", json_select(v("field"), jmes="a.b.c.d"))
```

加工结果：


```
{"field":"{\"a\":{\"b\":{\"c\":{\"d\":\"success\"}}}}","message":"success","status":"500"}
```


xml_to_json 函数

函数定义

解析 XML 值并转换为 JSON 字符串，输入值必须为 XML 字符串结构，否则会导致转换异常，返回 dict。

语法描述

```
xml_to_json(data)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围

版权所有：腾讯云计算（北京）有限责任公司

data	字段值	string	是	-	-

示例

原始日志：

```
{"xml_field": "<note><to>B</to><from>A</from><heading>Reminder</heading><body>Don't forget me this weekend!</body></note>", "status": "500"}
```

加工规则：

```
fields_set("json_field", xml_to_json(v("xml_field")))
```

加工结果：

```
{"xml_field":"<note><to>B</to><from>A</from><heading>Reminder</heading><body>Don't forget me this weekend!</body></note>","json_field":"{\"to\":\"B\",\"from\":\"A\",\"heading\":\"Reminder\",\"body\":\"Don't forget me this weekend!\"}","status":"500"}
```

json_to_xml 函数

函数定义

解析 JSON 字符串值并转换为 XML 字符串，返回 String。

语法描述

```
json_to_xml(data)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值	string	是	-	-

示例

原始日志：

```
{"json_field":"{\"to\":\"B\",\"from\":\"A\",\"heading\":\"Reminder\",\"body\":\"Don't forget me this weekend!\"}", "status": "200"}
```

加工规则：

```
fields_set("xml_field", json_to_xml(v("json_field")))
```

加工结果：

```
{"json_field":"{\"to\":\"B\",\"from\":\"A\",\"heading\":\"Reminder\",\"body\":\"Don't forget me this weekend!\"}","xml_field":"<ObjectNode><to>B</to><from>A</from><heading>Reminder</heading><body>Don't forget me this weekend!</body></ObjectNode>","status":"200"}
```

if_json 函数

函数定义

判断是否为 JSON 字符串，返回 TRUE/FALSE。

语法描述

```
if_json(data)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值	string	是	-	-

示例

- 示例1

  原始日志：

  ```
  {"condition":"{\"a\":\"b\"}","status":"500"}
  ```

  加工语句：

  ```
  t_if(if_json(v("condition")), fields_set("new", 1))
  ```

  加工结果：

  ```
  {"new":"1","condition":"{\"a\":\"b\"}","status":"500"}
  ```

- 示例2

  原始日志：

  ```
  {"condition":"haha","status":"500"}
  ```

  加工语句：

  ```
  t_if(if_json(v("condition")), fields_set("new", 1))
  ```

  加工结果：

  ```
  {"condition":"haha","status":"500"}
  ```

列表函数

array_get 函数

函数定义

获取数组的值，返回 String。

语法描述

```
array_get(数组, 下标位置)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
数组	数组	string	是	-	-

下标位置	取数组第几个值	int	是	-	-

示例1

原始日志：

```
{
    "field1": "[1,2,3]"
}
```

加工规则：

```
fields_set("field2", array_get(v("field1"), 0))
```

加工结果：

```
{"field1":"[1,2,3]","field2":"1"}
```

示例2

原始日志：

```
{
    "field1": "['tom','jerry','bobo']"
}
```

加工规则：

```
fields_set("field2", array_get(v("field1"), 0))
```

加工结果：

```
{"field1":"['tom','jerry','bobo']","field2":"tom"}
```

版权所有：腾讯云计算（北京）有限责任公司

正则处理函数

最近更新时间：2025-06-25 19:05:52

简介

日志含有大量的文本，在对文本的处理过程中，正则函数可以较为灵活的提取关键字、做脱敏、或者判断是否包含有指定的字符。
日志场景中常见的正则举例，请参见 在线测试正则公式 。

用途	日志原文	正则表达式	提取结果
提取大括号中的内容	[2021-11-24 11:11:08,232][328495eb-b562-478f-9d5d-3bf7e][INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}	\{[^\}]+\}	{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 10], "orderField": "createTime"}}}
提取中括号的内容	[2021-11-24 11:11:08,232][328495eb-b562-478f-9d5d-3bf7e][INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}	\[\S+\]	[328495eb-b562-478f-9d5d-3bf7e][INFO]
提取时间	[2021-11-24 11:11:08,232][328495eb-b562-478f-9d5d-3bf7e][INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}	\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2},\d{3}	2021-11-08 11:11:08,232
提取特定长度的大写字符	[2021-11-24 11:11:08,232][328495eb-b562-478f-9d5d-3bf7e][INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}	[A-Z]{4}	INFO
提取特定长度的小写字符	[2021-11-24 11:11:08,232][328495eb-b562-478f-9d5d-3bf7e][INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 15], "orderField": "createTime"}}}	[a-z]{6}	versio passwo timest interf create
提取字母+数字	[2021-11-24 11:11:08,232][328495eb-b562-478f-9d5d-3bf7e][INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}	([a-z]{3}):([0-9]{4})	com:8080

regex_match 函数

函数定义

基于正则对数据进行匹配，返回是否匹配成功，可以选择全匹配还是部分匹配。

> **说明：**
> 在部分场景下，该函数可使用 str_exist 函数代替，处理效率更高。

语法描述

```
regex_match(data, regex="", full=true)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值	string	是	-	-
regex	正则表达式	string	是	-	-
full	• full=True, 完全匹配，代表regex完全匹配data，regex_match 函数才会返回True • full=False，部分匹配，代表regex匹配了data的一部分，regex_match函数就会返回True.	bool	否	True	T/F

示例

- 示例1：判断正则公式"\d{3}.\d{3}.\d.\d"和192.168.0.1是否完全匹配，regex_match 函数返回 True。正则表达式"\d{3}.\d{3}"和192.168.0.1是否完全匹配，regex_match 函数返回 False.
  原始日志：

```
{"IP":"192.168.0.1", "status": "500"}
```

  加工规则：

```
//判断正则公式"\d{3}.\d{3}.\d.\d"和字段ip的值192.168.0.1是否完全匹配，将结果保存到新字段"matched"中。
t_if_else(regex_match(v("IP"), regex="\d{3}.\d{3}.\d.\d", full=True), fields_set("matched",
True),fields_set("matched", False))
//正则公式"\d{3}.\d{3}"和字段ip的值192.168.0.1是否完全匹配：
t_if_else(regex_match(v("IP"), regex="\d{3}.\d{3}", full=True), fields_set("matched2",
True),fields_set("matched2", False))
```

```
{"IP":"192.168.0.1","matched":"true","matched2":"false","status":"500"}
```

- 示例2：判断正则公式"\d{3}.\d{3}"和字段 IP 的值192.168.0.1是否部分匹配，regex_match 函数返回 True。
  原始日志：

```
{"IP":"192.168.0.1", "status": "500"}
```

  加工规则：

```
t_if(regex_match(v("IP"), regex="\d{3}.\d{3}", full=False), fields_set("matched", True))
```

  加工结果：

```
{"IP":"192.168.0.1","matched":"true","status":"500"}
```

regex_select 函数

函数定义

基于正则对数据进行匹配，返回相应的部分匹配结果，可以指定匹配结果的第几个表达式，以及第几个分组（部分匹配+指定捕获组序号），如果最终没有匹配结果，则返回空字符串。

语法描述

```
regex_select(data, regex="", index=1, group=1)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值	string	是	-	-
regex	正则表达式	string	是	-	-
index	匹配结果中的第几个表达式	number	否	默认第一个	-
group	匹配结果中的第几个分组	number	否	默认第一个	-

示例

根据正则表达式，对字段值进行不同的截取。
原始日志：

```
{"data":"hello123,world456", "status": "500"}
```

加工规则：

```
fields_set("match_result", regex_select(v("data"), regex="[a-z]+(\d+)",index=0, group=0))
fields_set("match_result1", regex_select(v("data"), regex="[a-z]+(\d+)", index=1, group=0))
fields_set("match_result2", regex_select(v("data"), regex="([a-z]+)(\d+)",index=0, group=0))
fields_set("match_result3", regex_select(v("data"), regex="([a-z]+)(\d+)",index=0, group=1))
```

加工结果：

```
{"match_result2":"hello123","match_result1":"world456","data":"hello123,world456","match_result3":"hello",
"match_result":"hello123","status":"500"}
```

regex_split 函数

函数定义

基于正则对数据进行分割，返回 JSON Array 字符串（部分匹配）。

语法描述

```
regex_split(data, regex=\"\", limit=100)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值	string	是	-	-
regex	正则表达式	string	是	-	-
limit	分割最大数组长度，当超过长度时，剩余未分割部分将作为一个元素，添加到数组	number	否	默认值100	-

示例

原始日志：

```
{"data":"hello123world456", "status": "500"}
```

加工规则：

```
fields_set("split_result", regex_split(v("data"), regex="\d+"))
```

加工结果：

```
{"data":"hello123world456","split_result":"[\"hello\",\"world\"]","status":"500"}
```

regex_replace 函数

函数定义

基于正则匹配并替换（部分匹配），主要用于脱敏场景。

语法描述

```
regex_replace(data, regex="", replace="", count=0)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值	string	是	-	-
regex	正则表达式	string	是	-	-
replace	目标字符串，使用此字符串替换匹配结果	string	是	-	-
count	替换次数，默认0，进行全部替换	number	否	默认值0	-

示例

- 示例1：根据正则表达式，对字段值进行替换。
  原始日志：

  ```
  {"data":"hello123world456", "status": "500"}
  ```

  加工规则：

  ```
  fields_set("replace_result", regex_replace(v("data"), regex="\d+", replace="", count=0))
  ```

  加工结果：

  ```
  {"replace_result":"helloworld","data":"hello123world456","status":"500"}
  ```

- 示例2：对用户 ID、手机号码、IP 地址进行脱敏。
  原始日志：

  ```
  {"Id": "dev@12345","Ip": "11.111.137.225","phonenumber": "13912345678"}
  ```

  加工规则：

```
//对 Id 字段进行脱敏处理，结果为dev@***45
fields_set("Id",regex_replace(v("Id"),regex="\d{3}", replace="***",count=0))
fields_set("Id",regex_replace(v("Id"),regex="\S{2}", replace="**",count=1))
//对 phonenumber 字段进行脱敏处理，将中间的4位数替换为****，结果为139****5678
fields_set("phonenumber",regex_replace(v("phonenumber"),regex="(\d{0,3})\d{4}(\d{4})",
replace="$1****$2"))
//对 IP 字段进行脱敏处理，将第二段替换为***，结果为11.***137.225。
fields_set("Ip",regex_replace(v("Ip"),regex="(\d+\.)\d+(\.\d+\.\d+)", replace="$1***$2",count=0))
```

加工结果：

```
{"Id":"**v@***45","Ip":"11.***.137.225","phonenumber":"139****5678"}
```

regex_findall 函数

函数定义

基于正则进行匹配，并将匹配结果添加到 JSON 数组中，并返回 Array 字符串（部分匹配）。

语法描述

```
regex_findall(data, regex="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值	string	是	-	-
regex	正则表达式	string	是	-	-

示例

原始日志：

```
{"data":"hello123world456", "status": "500"}
```

加工规则：

```
fields_set("result", regex_findall(v("data"), regex="\d+"))
```

加工结果：

```
{"result":"[\"123\",\"456\"]","data":"hello123world456","status":"500"}
```

sensitive_detection函数

函数定义

检测敏感信息，例如身份证、银行卡等敏感信息。

> **注意：**
> 请谨慎使用脱敏遮盖选项 replace_items，脱敏后的数据无法复原。

语法描述

```
sensitive_detection(scope="", ratio=1, discover_items="", replace_items="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
scope	检测字段名称	string	是	ALL_FIELDS	-
sample_ratio	检测比例，取值如下： • 1：表示全部检测 • 0.5：表示采样50%进行检测	number	是	-	-
discover_items	检测项，以 , 分隔	string	是	-	CHINA_PHONE_NUM,EMAIL,CHINA_IDCARD, ADDR,DEBIT_CARD,CREDIT_CARD,CHINA_PASSPORT,MAC_ADDR,IP,DOMAIN,LOCATION,VIN,PLATE_NUMBER,NAME,PASSWORD,TOKEN
replace_items	遮盖项，以 , 分隔	string	是	-	CHINA_PHONE_NUM,EMAIL,CHINA_IDCARD, ADDR,DEBIT_CARD,CREDIT_CARD,CHINA_PASSPORT,MAC_ADDR,IP,VIN,PASSWORD,TOKEN,NAME

示例

原始日志：

```
{
    "sensitive_field1": "CLS log 13912345678 my car  JTJHT00W274025559 www.tencent.com CLS数据加工",
    "sensitive_field2": "etl@tenctent.com ",
    "NON_sensitive_field": "hello world"
}
```

加工规则：

```
/*scope=ALL_FIELDS意为搜索整条日志中的所有字段
sample_ratio=1,意为对所有的日志进行敏感信息检测
电话、Email进行检测
并将电话信息脱敏（遮盖），请谨慎使用replace_items选项，遮盖后无法复原.*/
sensitive_detection(scope="ALL_FIELDS", sample_ratio=1,
discover_items="CHINA_PHONE_NUM,EMAIL",replace_items="CHINA_PHONE_NUM")
```

加工结果：

```
{
"NON_sensitive_field":"hello world",
"SENSITIVE_FLAGS":"CHINA_PHONE_NUM,EMAIL",//检测到了电话号码和邮件两种敏感信息
//遮盖了电话号码
"sensitive_field1":"CLS log  139****5678 my car  JTJHT00W274025559 www.tencent.com CLS数据加工",
"sensitive_field2":"etl@tenctent.com "
}
```

敏感检测项说明：

敏感信息项	描述	例子	正则公式	遮盖结果

CHINA_PHONE_NUM	中国手机号	13123456789	(1\\d{2})(\\d{4})(\\d{4})	139****1234
EMAIL	邮箱	abcd@nio.com	([A-Za-z0-9._%+-]+)(@[A-Za-z0-9.-]+\\.[A-Za-z]{2,})	***@nio.com
CHINA_IDCARD	中国身份证	420101199004135043	(1[1-5]|2[1-3]|3[1-7]|4[1-6]|5[0-4]|6[1-5]|[7-9]1)\\d{4}((18|19|20)\\d{2}((0[1-9])|(1[0-2]))((0[1-9]|1\\d|2[0-8])|(1[0-2](29|30)))|(1[013-9]|2[0-35-9])31)\\d{3}[0-9Xx]	420101****5043
ADDR	中文地址	北京市海淀区北三环西路43号	((.{1,6}?(省|市|自治区|自治州|县|区|镇|乡))){1,3}((.{1,6}(路|街|里|街道|村|屯|组|小区|大厦|号|广场))){1,3}((.{1,6}(号楼|单元|层|室|户|号|房))|(\\d+-\\d+-\\d+)){0,3}	全部遮盖为****
DEBIT_CARD	借记卡	6225092716776464882	(62\\d{5,11})(\\d{6})	622509271677********
CREDIT_CARD	信用卡	4539138994741478	([1-9]\\d{3}[\\s-]?\\d{4}[\\s-]?\\d{4}[\\s-]?)(\\d{4})	453913899474***
CHINA_PASSPORT	中国护照	G86067430	((1[45]\\d{7})|([P|p|S|s]\\d{7})|([S|s|G|g|E|e]\\d{8})|([Gg|Tt|Ss|Ll|Qq|Dd|Aa|Ff]\\d{8})|([H|h|M|m]\\d{8,10}))	全部遮盖为****
MACADDR	Mac 地址	06-06-06-aa-bb-cc	([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})	06-06-06-aa-****cc
IP	IP 地址	120.32.23.137	((?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?))	120.32.23.***
DOMAIN	域名	www.abc123.com	((\\w|(\\w[\\w-]{0,86}\\w))\\.(\\w|(\\w[\\w-]{0,73}\\w))\\.((\\w{2,12}\\.\\w{2,12})|(\\w{2,25})))|((\\w|(\\w[\\w-]{0,162}\\w))\\.((\\w{2,12}\\.\\w{2,12})|(\\w{2,25})))	不提供遮盖功能
LOCATION	经纬度	31.886551,120.443934	[\\-\+]?0(\.\d{4,10})|([1-9](\d)?)(\.\d{4,10})|1[0-7]\d{1}(\.\d{4,10})|180\.0{1,10}	不提供遮盖功能
VIN	车架号	LJ1EEAUU8J7700492	([A-HJ-NPR-Z\\d]{10})([A-HJ-NPR-Z\\d]{7})	LJ1EEAUU8J****
plate_number	车牌号	京 N5J980	[京津沪渝冀豫云辽黑湘皖鲁新苏浙赣鄂桂甘晋蒙陕吉闽贵粤青藏川宁琼使领][A-Z]{1}[A-HJ-NP-Z0-9]{4}[A-HJ-NP-Z0-9挂学警港澳]	不提供遮盖功能
NAME	名字	日志字段名为:姓名	字段名:["real_name","family_name","last_name","姓名","名字","用户名", "收件人","recv_person", "receive_person"]	全部遮盖为****
password	密码	日志字段名为:password	字段名:["password", "passwd", "secret", "pass", "密码", "凭证"]	全部遮盖为****
token	token	日志字段名为:token	字段名:["token", "account_key", "api_key", "授权码"]	全部遮盖为****

版权所有:腾讯云计算(北京)有限责任公司

时间值处理函数

最近更新时间：2024-07-08 15:22:01

简介

日志中的时间处理函数，包括将 Date 类型转为 String 类型、时间类字段值和 UTC 时间互转以及获取当前时间。

dt_str 函数

函数定义

将时间类的字段值（特定格式的日期字符串或者时间戳），转换为指定时区、格式的目标日期字符串。

语法描述

```
dt_str(值, format="格式化字符串", zone="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值，可支持的解析格式可参考 dateparser	string	是	-	-
format	格式化日期格式可参考 DateTimeFormatter	string	否	-	-
zone	默认 UTC 时间，不指定时区。时区定义可参考 ZoneId	string	否	UTC+00:00	-

示例

原始日志：

```
{"date":"2014-04-26 13:13:44 +09:00"}
```

加工规则：

```
fields_set("result", dt_str(v("date"), format="yyyy-MM-dd HH:mm:ss", zone="UTC+8"))
```

加工结果：

```
{"date":"2014-04-26 13:13:44 +09:00","result":"2014-04-26 12:13:44"}
```

dt_to_timestamp 函数

函数定义

将时间类的字段值（特定格式的日期字符串），同时指定字段对应的时区，转换为 UTC 时间戳。

语法描述

```
dt_to_timestamp(值, zone="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值，可支持的解析格式可参考 dateparser	string	是	-	-

zone	默认 UTC 时间，不指定时区。如果指定此值，则必须和时间字段值对应，否则会出现时区错误问题。时区定义可参考 ZoneId	string	否	UTC+00:00	-

示例

原始日志：

```
{"date":"2021-10-26 15:48:15"}
```

加工规则：

```
fields_set("result", dt_to_timestamp(v("date"), zone="UTC+8"))
```

加工结果：

```
{"date":"2021-10-26 15:48:15","result":"1635234495000"}
```

dt_from_timestamp 函数

函数定义

将时间类的时间戳字段，指定目标时区后，转换为时间字符串。

语法描述

```
dt_from_timestamp(值, zone="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字段值，可支持的解析格式可参考 dateparser	string	是	-	-
zone	默认 UTC 时间，不指定时区。时区定义可参考 ZoneId	string	否	UTC+00:00	-

示例

原始日志：

```
{"date":"1635234495000"}
```

加工规则：

```
fields_set("result", dt_from_timestamp(v("date"), zone="UTC+8"))
```

加工结果：

```
{"date":"1635234495000","result":"2021-10-26 15:48:15"}
```

dt_now 函数

函数定义

获取加工计算时的本地时间。

语法描述

```
dt_now(format="格式化字符串", zone="")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
format	格式化日期格式，可参考 DateTimeFormatter	string	否	-	-
zone	默认 UTC 时间，不指定时区。时区定义可参考 ZoneId	string	否	UTC+00:00	-

示例

原始日志：

```
{"date":"1635234495000"}
```

加工规则：

```
fields_set("now", dt_now(format="yyyy-MM-dd HH:mm:ss", zone="UTC+8"))
```

加工结果，仅参考，具体结果和系统时间相关：

```
{"date":"1635234495000","now":"2021-MM-dd HH:mm:ss"}
```

custom_cls_log_time 函数

函数定义

自定义日志时间，会按照您的加工规则生成新的日志时间，支持 秒、毫秒、微秒、纳秒。点击**执行预览**后，您可在加工结果中，查看__TIMESTAMP__字段的值，是否已改变为您的预期结果。

语法描述

```
custom_cls_log_time(time)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
time	UTC 时间戳类型，时区定义可参考 ZoneId，支持 秒、毫秒、微秒、纳秒，例如1565064739000	string	否	UTC+00:00	-

返回值

UTC 时间戳类型，例如1565064739000。

示例

原始日志：

```
{"field1": "1","time":"06/Aug/2019 12:12:19"}
```

加工规则：

```
custom_cls_log_time(dt_to_timestamp(v("time"), zone="UTC+8"))
```

加工结果，仅参考，具体结果和系统时间相关：

```
{"__TIMESTAMP__":"1565064739000", "field1":"1", "time":"06/Aug/2019 12:12:19"}
```

字符串处理函数

最近更新时间：2024-04-28 10:41:11

简介

字符串处理函数，支持字符串长度计算、大小写转换、拼接字符串、替换、剔除、查找字符所在位置、前后缀匹配等。

str_exist 函数

函数定义

在值中指定范围内查找子串，返回 True 或 False。

语法描述

```
str_exist(data1, data2, ignore_upper=False)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	字符串类型的值	String	是		-
data2	字符串类型的值	String	是		-
ignore_upper	是否区分大小写	Bool	否	False	-

示例

原始日志：

```
{"data": "cls nihao"}
```

加工规则：

```
fields_set("result", str_exist(v(data), "nihao"))
```

加工结果：

```
{"result":"true","data":"cls nihao"}
```

str_count 函数

函数定义

在值中指定范围内查找子串，返回子串出现的次数。

语法描述

```
str_count(值, sub="", start=0, end=-1)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-
sub	需要查找的子字符串	string	是	-	-
start	查找的起始位置	number	否	默认为0	-
end	查找的结束位置	number	否	默认为-1	-

版权所有：腾讯云计算（北京）有限责任公司

示例

原始日志：

```
{"data": "warn,error,error"}
```

加工规则：

```
fields_set("result", str_count(v("data"), sub="err"))
```

加工结果：

```
{"result":"2","data":"warn,error,error"}
```

str_len 函数

函数定义

返回字符串长度。

语法描述

```
str_len(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-

示例

原始日志：

```
{"data": "warn,error,error"}
```

加工规则：

```
fields_set("result", str_len(v("data")))
```

加工结果：

```
{"result":"16","data":"warn,error,error"}
```

str_uppercase 函数

函数定义

返回大写字符串。

语法描述

```
str_uppercase(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围

data	字符串类型的值	string	是	-	-

示例

原始日志：

```
{"data": "warn,error,error"}
```

加工规则：

```
fields_set("result", str_uppercase(v("data")))
```

加工结果：

```
{"result":"WARN,ERROR,ERROR","data":"warn,error,error"}
```

str_lowercase 函数

函数定义

返回小写字符串。

语法描述

```
str_lowercase(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-

示例

原始日志：

```
{"data": "WARN,ERROR,ERROR"}
```

加工规则：

```
fields_set("result", str_lowercase(v("data")))
```

加工结果：

```
{"result":"warn,error,error","data":"WARN,ERROR,ERROR"}
```

str_join 函数

函数定义

使用拼接字符串，拼接多值。

语法描述

```
str_join(拼接字符串1, 值1, 值2, ...)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
join	字符串类型的值	string	是	-	-
值参数，可变参数列表	字符串类型的值	string	是	-	-

示例

原始日志：

```
{"data": "WARN,ERROR,ERROR"}
```

加工规则：

```
fields_set("result", str_join(",", v("data"), "INFO"))
```

加工结果：

```
{"result":"WARN,ERROR,ERROR,INFO","data":"WARN,ERROR,ERROR"}
```

str_replace 函数

函数定义

替换字符串，返回替换结果字符串。

语法描述

```
str_replace(值, old="", new="", count=0)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-
old	将要被替换的字符串	string	是	-	-
new	替换的目标字符串	string	是	-	-
count	最大替换次数，默认全部替换	number	否	默认为0	-

示例

将字段 data 的值中的 "WARN" 替换为 "ERROR"。
原始日志：

```
{"data": "WARN,ERROR,ERROR"}
```

加工规则：

```
fields_set("result", str_replace( v("data"), old="WARN", new="ERROR"))
```

将替换的结果保存到新字段 "result" 中。
加工结果：

```
{"result":"ERROR,ERROR,ERROR","data":"WARN,ERROR,ERROR"}
```

版权所有：腾讯云计算（北京）有限责任公司

str_format 函数

函数定义

格式化字符串，返回格式化结果。

语法描述

```
str_format(格式化字符串, 值1, 值2, ...)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
format	目标格式，占位符使用{}，例如："The disk contains {0} files."。其中{}中的数字，对应后面参数值的序号，从0开始。具体使用可参考 MessageFormat.format 函数的使用	string	是	-	-
值参数，可变参数列表	字符串类型的值	string	是	-	-

示例

原始日志：

```
{"status": 200, "message":"OK"}
```

加工规则：

```
fields_set("result", str_format("status:{0}, message:{1}", v("status"), v("message")))
```

加工结果：

```
{"result":"status:200, message:OK","message":"OK","status":"200"}
```

str_strip 函数

函数定义

移除字符串的开头和结尾处的指定字符。

语法描述

```
str_strip(值, chars="\t\r\n")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-
chars	将要被剔除的字符串	string	否	\t\r\n	-

示例

- 示例1
  原始日志：

```
{"data": " abc  "}
```

加工规则：

```
fields_set("result", str_strip(v("data"), chars=" "))
```

加工结果：

```
{"result":"abc","data":" abc  "}
```

- 示例2
  原始日志：

```
{"data": " **abc**  "}
```

加工规则：

```
fields_set("result", str_strip(v("data"), chars=" *"))
```

加工结果：

```
{"result":"abc","data":" **abc**  "}
```

str_lstrip 函数

函数定义

移除字符串开头处的指定字符。

语法描述

```
str_strip(值, chars="\t\r\n")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-
chars	将要被剔除的字符串	string	否	\t\r\n	-

示例

原始日志：

```
{"data": " abc  "}
```

加工规则：

```
fields_set("result", str_lstrip(v("data"), chars=" "))
```

加工结果：

```
{"result":"abc  ","data":" abc  "}
```

str_rstrip 函数

函数定义

移除字符串结尾处的指定字符。

语法描述

```
str_strip(值, chars="\t\r\n")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-
chars	将要被剔除的字符串	string	否	\t\r\n	-

示例

原始日志：

```
{"data": " abc  "}
```

加工规则：

```
fields_set("result", str_rstrip(v("data"), chars=" "))
```

加工结果：

```
{"result":" abc","data":" abc  "}
```

str_find 函数

函数定义

在值中查找子串，并返回子串出现的位置。

语法描述

```
str_find(值, sub="", start=0, end=-1)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-
sub	准备查找的子字符串	string	是	-	-
start	查找的起始位置	number	否	默认为0	-
end	查找的结束位置	number	否	默认为-1	-

示例

原始日志：

```
{"data": "warn,error,error"}
```

加工规则：

```
fields_set("result", str_find(v("data"), sub="err"))
```

加工结果：

```
{"result":"5","data":"warn,error,error"}
```

str_start_with 函数

函数定义

判断字符串是否以指定前缀开头。

语法描述

```
str_start_with(值, sub="", start=0, end=-1)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-
sub	前缀字符串	string	是	-	-
start	查找的起始位置	number	否	默认为0	-
end	查找的结束位置	number	否	默认为-1	-

示例

- 示例1
  原始日志：

  ```
  {"data": "something"}
  ```

  加工规则：

  ```
  fields_set("result", str_start_with(v("data"), sub="some"))
  ```

  加工结果：

  ```
  {"result":"true","data":"something"}
  ```

- 示例2
  原始日志：

  ```
  {"data": "something"}
  ```

  加工规则：

  ```
  fields_set("result", str_start_with(v("data"), sub="*"))
  ```

  加工结果：

```
{"result":"false","data":"something"}
```

str_end_with 函数

函数定义

判断字符串是否以指定后缀结尾。

语法描述

```
str_end_with(值, sub="", start=0, end=-1)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	字符串类型的值	string	是	-	-
sub	后缀字符串	string	是	-	-
start	查找的起始位置	number	否	默认为0	-
end	查找的结束位置	number	否	默认为-1	-

示例

原始日志：

```
{"data": "endwith something"}
```

加工规则：

```
fields_set("result", str_end_with(v("data"), sub="ing"))
```

加工结果：

```
{"result":"true","data":"endwith something"}
```

类型转换函数

最近更新时间：2024-05-29 11:46:11

简介

函数提供了常见的类型转换，将字段值转为 int、float、bool 和 Str。

ct_int 函数

函数定义

对值进行整型转换，可指定原值的进制，转为十进制数值。

语法描述

```
ct_int(值1, base=10)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	数值类型的值或可转为数值的字符串	number	是	-	-
base	进制	number	否	默认为10	[2-36]

示例

- 示例1
  原始日志：

```
{"field1": "10"}
```

  加工规则：

```
fields_set("result", ct_int(v("field1")))
```

  加工结果：

```
{"result":"10","field1":"10"}
```

- 示例2
  原始日志：

```
{"field1": "AB"}
```

  加工规则：

```
fields_set("result", ct_int(v("field1"), 16))
```

  加工结果：

```
{"result":"171","field1":"AB"}
```

ct_float 函数

函数定义

将值转换为浮点型数值。

语法描述

```
ct_float(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	数值类型的值或可转为数值的字符串	number	是	-	-

示例

原始日志：

```
{"field1": "123"}
```

加工规则：

```
fields_set("result", ct_float(v("field1")))
```

加工结果：

```
{"result":"123.0","field1":"123"}
```

ct_str 函数

函数定义

将值转换为字符串。

语法描述

```
ct_str(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	数值类型的值或可转为数值的字符串	number	是	-	-

示例

原始日志：

```
{"field1": 123}
```

加工规则：

```
fields_set("result", ct_str(v("field1")))
```

加工结果：

```
{"result":"123","field1":"123"}
```

ct_bool 函数

函数定义

将值转换为布尔值。

语法描述

```
ct_bool(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	数值类型的值或可转为数值的字符串	number	是	-	-

示例

- 示例1
  原始日志：

  ```
  {}
  ```

  加工规则：

  ```
  fields_set("result", ct_bool(0))
  ```

  加工结果：

  ```
  {"result":"false"}
  ```

- 示例2
  原始日志：

  ```
  {}
  ```

  加工规则：

  ```
  fields_set("result", ct_bool(1))
  ```

  加工结果：

  ```
  {"result":"true"}
  ```

- 示例3
  原始日志：

  ```
  {"field1": 1}
  ```

  加工规则：

  ```
  fields_set("result", ct_bool(v("field1")))
  ```

  加工结果：

  ```
  {"result":"true","field1":"1"}
  ```

版权所有：腾讯云计算（北京）有限责任公司

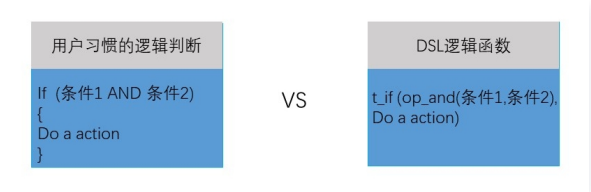
逻辑和数学函数

最近更新时间：2025-06-24 15:47:32

版权所有：腾讯云计算（北京）有限责任公司

简介

逻辑与数学函数，包含了与或非、大于小于等于、加减乘除取模的运算。写起来和我们在程序语言中稍有区别，如下图所示：



op_if 函数

函数定义

根据条件判断，返回相应的值。

语法描述

```
op_if(条件1, 值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
condition	条件表达式	bool	是	-	-
data1	条件为 True 时，返回此参数值	string	是	-	-
data2	条件为 False 时，返回此参数值	string	是	-	-

示例

- 示例1

  原始日志：

  ```
  {"data": "abc"}
  ```

  加工规则：

  ```
  fields_set("result", op_if(True, v("data"), "false"))
  ```

  加工结果：

  ```
  {"result":"abc","data":"abc"}
  ```

- 示例2

  原始日志：

  ```
  {"data": "abc"}
  ```

  加工规则：

  ```
  fields_set("result", op_if(False, v("data"), "123"))
  ```

  加工结果：

```
{"result":"123","data":"abc"}
```

op_and 函数

函数定义

对值进行 and 运算，均为 True 时，返回 True，否则返回 False。

语法描述

```
op_and(值1, 值2, ...)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参列表	参与计算的参数或表达式	string	是	-	-

示例

- 示例1

  原始日志：

  ```
  {}
  ```

  加工规则：

  ```
  fields_set("result", op_and(True, False))
  ```

  加工结果：

  ```
  {"result":"false"}
  ```

- 示例2

  原始日志：

  ```
  {}
  ```

  加工规则：

  ```
  fields_set("result", op_and(1, 1))
  ```

  加工结果：

  ```
  {"result":"true"}
  ```

- 示例3

  原始日志：

  ```
  {"data":"false"}
  ```

  加工规则：

  ```
  fields_set("result", op_and(1, v("data")))
  ```

版权所有：腾讯云计算（北京）有限责任公司

加工结果：

```
{"result":"false","data":"false"}
```

op_or 函数

函数定义

对值进行 or 运算，若存在参数值为 False 则返回 False，否则返回 True。

语法描述

```
op_or(值1, 值2, ...)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参列表	参与计算的参数或表达式	string	是	-	-

示例

原始日志：

```
{}
```

加工规则：

```
fields_set("result", op_or(True, False))
```

加工结果：

```
{"result":"true"}
```

op_not 函数

函数定义

对值进行 not 运算。

语法描述

```
op_not(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	任意类型的值	any	是	-	-

示例

- 示例1

  原始日志：

  ```
  {}
  ```

  加工规则：

```
fields_set("result", op_not(True))
```

加工结果：

```
{"result":"false"}
```

- 示例2

原始日志：

```
{}
```

加工规则：

```
fields_set("result", op_not("True"))
```

加工结果：

```
{"result":"false"}
```

op_eq 函数

函数定义

对值进行比较，相等则返回 True。

语法描述

```
op_eq(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-
data2	数值类型的值或可转为数值的字符串	number	是	-	-

示例

- 示例1：判断字段 "Post" 和 "Get" 的值是否相等。

原始日志：

```
{"Post": "10", "Get": "11"}
```

加工规则：

```
fields_set("result", op_eq(v("Post"), v("Get")))
```

将结果保存到 result。
加工结果：

```
{"result":"false","Post":"10","Get":"11"}
```

- 示例2：判断字段 "field1" 和 "field2" 的值是否相等。

原始日志：

```
{"field1": "1", "field2": "1"}
```

加工规则：

```
fields_set("result", op_eq(v("field1"), v("field2")))
```

加工结果：

```
{"result":"true","field1":"1","field2":"1"}
```

op_ge 函数

函数定义

对值进行比较，值1大于或等于值2时返回 True。

语法描述

```
op_ge(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-
data2	数值类型的值或可转为数值的字符串	number	是	-	-

示例

- 示例1

  原始日志：

  ```
  {"field1": "20", "field2": "9"}
  ```

  加工规则：

  ```
  fields_set("result", op_ge(v("field1"), v("field2")))
  ```

  加工结果：

  ```
  {"result":"true","field1":"20","field2":"9"}
  ```

- 示例2

  原始日志：

  ```
  {"field1": "2", "field2": "2"}
  ```

  加工规则：

  ```
  fields_set("result", op_ge(v("field1"), v("field2")))
  ```

  加工结果：

```
{"result":"true","field1":"2","field2":"2"}
```

op_gt 函数

函数定义

对值进行比较，值1大于值2时返回 True。

语法描述

```
op_gt(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-
data2	数值类型的值或可转为数值的字符串	number	是	-	-

示例

原始日志：

```
{"field1": "20", "field2": "9"}
```

加工规则：

```
fields_set("result", op_ge(v("field1"), v("field2")))
```

加工结果：

```
{"result":"true","field1":"20","field2":"9"}
```

op_le 函数

函数定义

对值进行比较，值1小于或等于值2时返回 True。

语法描述

```
op_le(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-
data2	数值类型的值或可转为数值的字符串	number	是	-	-

示例

原始日志：

```
{"field1": "2", "field2": "2"}
```

版权所有：腾讯云计算（北京）有限责任公司

加工规则：

```
fields_set("result", op_le(v("field1"), v("field2")))
```

加工结果：

```
{"result":"true","field1":"2","field2":"2"}
```

op_lt 函数

函数定义

对值进行比较，值1小于值2时返回 True。

语法描述

```
op_lt(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-
data2	数值类型的值或可转为数值的字符串	number	是	-	-

示例

原始日志：

```
{"field1": "2", "field2": "3"}
```

加工规则：

```
fields_set("result", op_lt(v("field1"), v("field2")))
```

加工结果：

```
{"result":"true","field1":"2","field2":"3"}
```

op_add 函数

函数定义

对值进行求和运算。

语法描述

```
op_add(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-
data2	数值类型的值或可转为数值的字符串	number	是	-	-

版权所有：腾讯云计算（北京）有限责任公司

示例

原始日志：

```
{"field1": "1", "field2": "2"}
```

加工规则：

```
fields_set("result", op_add(v("field1"), v("field2")))
```

加工结果：

```
{"result":"3","field1":"1","field2":"2"}
```

op_sub 函数

函数定义

对值进行求差运算。

语法描述

```
op_sub(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-
data2	数值类型的值或可转为数值的字符串	number	是	-	-

示例

原始日志：

```
{"field1": "1", "field2": "2"}
```

加工规则：

```
fields_set("result", op_sub(v("field1"), v("field2")))
```

加工结果：

```
{"result":"-1","field1":"1","field2":"2"}
```

op_mul 函数

函数定义

对值进行乘积运算。

语法描述

```
op_mul(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-
data2	数值类型的值或可转为数值的字符串	number	是	-	-

示例

原始日志：

```
{"field1": "1", "field2": "2"}
```

加工规则：

```
fields_set("result", op_mul(v("field1"), v("field2")))
```

加工结果：

```
{"result":"2","field1":"1","field2":"2"}
```

op_div 函数

函数定义

对值进行除法运算。

语法描述

```
op_div(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-
data2	数值类型的值或可转为数值的字符串	number	是	-	-

示例

- 示例1

  原始日志：

  ```
  {"field1": "1", "field2": "2"}
  ```

  加工规则：

  ```
  fields_set("result", op_div(v("field1"), v("field2")))
  ```

  加工结果：

  ```
  {"result":"0","field1":"1","field2":"2"}
  ```

- 示例2

  原始日志：

```
{"field1": "1.0", "field2": "2"}
```

加工规则：

```
fields_set("result", op_div(v("field1"), v("field2")))
```

加工结果：

```
{"result":"0.5","field1":"1.0","field2":"2"}
```

op_sum 函数

函数定义

对多值累加求和。

语法描述

```
op_sum(值1, 值2, ...)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
可变参列表	数值类型的值或可转为数值的字符串	string	是	-	-

示例

原始日志：

```
{"field1": "1.0", "field2": "10"}
```

加工规则：

```
fields_set("result", op_sum(v("field1"), v("field2")))
```

加工结果：

```
{"result":"11.0","field1":"1.0","field2":"10"}
```

op_mod 函数

函数定义

对值进行模计算。

语法描述

```
op_mod(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-

示例

- 示例1

  原始日志：

```
{
    "field1": "3",
    "field2": "2"
}
```

  加工规则：

```
fields_set("result", op_mod(v("field1"), v("field2")))
```

  加工结果：

```
{"field1":"3","field2":"2","result":"1"}
```

- 示例2

  原始日志：

```
{"field1": "1.0", "field2": "5"}
```

  加工规则：

```
fields_set("result", op_mod(v("field1"), v("field2")))
```

  加工结果：

```
{"result":"1.0","field1":"1.0","field2":"5"}
```

- 示例3

  原始日志：

```
{"field1": "6", "field2": "4"}
```

  加工规则：

```
fields_set("result", op_mod(v("field1"), v("field2")))
```

  加工结果：

```
{"result":"2","field1":"6","field2":"4"}
```

op_null 函数

函数定义

对值进行是否为 null 判断，是则返回 true，否则返回 false。

语法描述

```
op_null(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	任意类型的值	any	是	-	-

示例

- 示例1

  原始日志：

```
{}
```

  加工规则：

```
fields_set("result", op_null("null"))
```

  加工结果：

```
{"result":"true"}
```

- 示例2

  原始日志：

```
{"data": null}
```

  加工规则：

```
fields_set("result", op_null(v("data")))
```

  加工结果：

```
{"data": "null", "result":"true"}
```

op_notnull 函数

函数定义

对值进行是否为非 null 判断，是则返回 true，否则返回 false。

语法描述

```
op_notnull(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	任意类型的值	any	是	-	-

示例

- 示例1

  原始日志：

```
{}
```

  加工规则：

```
fields_set("result", op_notnull("null"))
```

加工结果：

```
{"result":"false"}
```

- 示例2

原始日志：

```
{"data": null}
```

加工规则：

```
fields_set("result", op_notnull(v("data")))
```

加工结果：

```
{"data": "null", "result":"false"}
```

op_str_eq 函数

函数定义

对字符串值进行比较，相等则返回 true。

语法描述

```
op_str_eq(值1, 值2, ignore_upper=False)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	字符串类型的值	string	是	-	-
data2	字符串类型的值	string	是	-	-
ignore_upper	是否区分大小写	bool	否	False	-

示例

- 示例1

原始日志：

```
{"field": "cls"}
```

加工规则：

```
fields_set("result", op_str_eq(v("field"), "cls"))
```

加工结果：

```
{"result":"true","field":"cls"}
```

- 示例2

原始日志：

```
{"field": "cls"}
```

加工规则：

```
fields_set("result", op_str_eq(v("field"), "etl|cls|data"))
```

加工结果：

```
{"result":"true","field":"cls"}
```

- 示例3

原始日志：

```
{"field": "CLS"}
```

加工规则：

```
fields_set("result", op_str_eq(v("field"), "cls", ignore_upper=True))
```

加工结果：

```
{"result":"true","field":"CLS"}
```

- 示例4

原始日志：

```
{"field": "CLS"}
```

加工规则：

```
fields_set("result", op_str_eq(v("field"), "etl|cls|data", ignore_upper=True))
```

加工结果：

```
{"result":"true","field":"CLS"}
```

random 函数

函数定义

生成随机数（两个参数之间的一个随机数），左闭右闭。

语法描述

```
random(值1, 值2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data1	数值类型的值或可转为数值的字符串	number	是	-	-

data2	数值类型的值或可转为数值的字符串	number	是	-	-

示例

- 示例1

  原始日志：

```
{"field1": "1"}
```

  加工规则：

```
log_keep(op_eq(random(3, 5), 4))
```

  加工结果：

```
{"field1": "1"}
```

- 示例2

  原始日志：

```
{"field1": "1"}
```

  加工规则：

```
fields_set("field2", random(1, 5))
```

  加工结果：

```
{"field1":"1", "field2":"4"}
```

op_len 函数

函数定义

计算文本字符串中的字符数，可用于字符串和其他返回元组、列表、字典的表达式。

语法描述

```
op_len(value)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
value	-	字符串、元组、列表或字典等	是	-	-

示例

原始日志：

```
{"data":"[1,2]"}
```

加工规则：

```
fields_set("length", op_len(v("data")))
```

加工结果：

```
{"length":"2","data":"[1,2]"}
```

op_ne 函数

函数定义

按照 a!=b 条件进行计算，返回 True 或 False。

语法描述

```
op_ne(value1, value2)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
value1	被对比的值	数值类型的值或可转为数值的字符串	是	-	-
value2	对比 value1	数值类型的值或可转为数值的字符串	是	-	-

示例

原始日志：

```
{"data1": "1","data2": "2"}
```

加工规则：

```
fields_set("result", op_ne(v("data1"), v("data2")))
```

加工结果：

```
{"result":"true","data1": "1","data2": "2"}
```

编解码函数

最近更新时间：2025-06-24 15:47:32

decode_url 函数

函数定义

将编码 URL 进行解码。

语法描述

```
decode_url(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
url	URL 值	String	是	-	-

示例

原始日志：

```
{"url":"https%3A%2F%2Fcloud.tencent.com%2F"}
```

加工规则：

```
fields_set("result",decode_url(v("url")))
```

加工结果：

```
{"result":"https://cloud.tencent.com/","url":"https%3A%2F%2Fcloud.tencent.com%2F"}
```

md5_encoding 函数

函数定义

计算并返回 MD5值。

语法描述

```
md5_encoding(值)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
值	待计算 MD5值的数据	String	是	-	-

示例

原始日志：

```
{"field": "haha"}
```

加工规则：

```
fields_set("field", md5_encoding(v("field")))
```

加工结果：

```
{"field":"4e4d6c332b6fe62a63afe56171fd3725"}
```

uuid 函数

函数定义

生成 universally unique identifier (UUID)，即唯一识别码。

语法描述

```
uuid()
```

参数说明

无入参。

示例

原始日志：

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

加工规则：

```
fields_set("field",uuid())
```

加工结果：

```
{"field":"8c2db704-45c0-4ea1-9e2c-cf9c966e35cd","key":"value"}
```

str_encode 函数

函数定义

对字符串进行指定格式的编码。

语法描述

```
str_encode(data, encoding="utf8", errors="ignore")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	待编码的数据	String	是	-	-
encoding	编码格式，默认为 utf8，支持 ASCII、latin1、unicode-escape	String	否	utf8	-
errors	编码格式无法识别字符时的处理方式 • ignore（默认值）：忽略不做编码 • strict：直接报错，丢弃此条日志数据 • replace：使用半角问号（？）替换无法识别部分	String	否	ignore	-

	• xmlcharrefreplace：使用对应 XML 字符替换无法识别部分				

示例

原始日志：

```
{"field1": "asd编码编码\\u1234"}
```

加工规则：

```
fields_set("field2", str_decode(str_encode(v("field1"), "unicode-escape"), "unicode-escape"))
```

加工结果：

```
{"field1":"asd编码编码\\u1234","field2":"asd编码编码ሴ"}
```

str_decode 函数

函数定义

对字符串进行指定格式的解码。

语法描述

```
str_decode(data, encoding="utf8", errors="ignore")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	待解码的数据	String	是	-	-
encoding	编码格式，默认为 utf8，支持 ASCII、latin1、unicode-escape	String	否	utf8	-
errors	编码格式无法识别字符时的处理方式 • ignore（默认值）：忽略不做解码 • strict：直接报错，丢弃此条日志数据 • replace：使用半角问号（？）替换无法解码部分 • xmlcharrefreplace：使用对应 XML 字符替换无法解码部分	String	否	ignore	-

示例

原始日志：

```
{"field1": "英文和中文测试:qwertyuiopasdfghjklzxcvbnm QWERTYUIOPASDFGHJKLZXCVBNM 特殊符号：】】【!@#$%^&*()_++~"}
```

加工规则：

```
fields_set("field2", str_decode(str_encode(v("field1"))))
```

加工结果：

```
{"field1":"英文和中文测试:qwertyuiopasdfghjklzxcvbnm QWERTYUIOPASDFGHJKLZXCVBNM 特殊符号：】】【!@#$%^&*()_++~", "field2":"英文和中文测试:qwertyuiopasdfghjklzxcvbnm QWERTYUIOPASDFGHJKLZXCVBNM 特殊符号：】】【!@#$%^&*()_++~"}
```

base64_encode 函数

函数定义

将字符串进行 base64编码。

语法描述

```
base64_encode(值, format="RFC3548")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
value	待编码字符串	string	是	-	-
format	编码格式，支持 RFC4648（默认）、RFC3548	string	否	RFC4648	-

示例

原始日志：

```
{"field": "hello world"}
```

加工规则：

```
fields_set("encode", base64_encode(v("field")))
```

加工结果：

```
{"encode":"aGVsbG8gd29ybGQ=", "field":"hello world"}
```

base64_decode 函数

函数定义

将字符串进行 base64解码。

语法描述

```
base64_decode(值, format="RFC3548")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
value	待解码字符串	string	是	-	-
format	解码格式，支持 RFC4648（默认）、RFC3548	string	否	RFC4648	-

示例

原始日志：

```
{"field": "aGVsbG8gd29ybGQ="}
```

加工规则：

```
fields_set("decode", base64_decode(v("field")))
```

加工结果：

```
{"decode":"hello world", "field":"aGVsbG8gd29ybGQ="}
```

hashids_encode 函数

函数定义

使用 Hashids 库对数据进行编码。

语法描述

```
hashids_encoding(value, salt="", min_length=0)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
value	待编码字符串	Number、List、Tuple	是	-	当 value 为数字时，不能小于0
salt	为数据添加盐值，默认为空字符串	String	是	""	
min_length	生成的哈希值的最小长度，默认为0	Number	是	0	-

示例1

原始日志：

```
{"data": "[123, 456]"}
```

加工规则：

```
fields_set("hashid", hashids_encode(v("data"), salt="test", min_length=32))
```

加工结果：

```
{"data":"[123, 456]", "hashid":"O6jN0Z7VARqDzbpBHYOakeowE5Xnr41g"}
```

示例2

原始日志：

```
{"data1": "123",  "data2": "456"}
```

加工规则：

```
fields_set("hashid", hashids_encode(v("data1"),v("data2"), salt="test", min_length=32))
```

加工结果：

```
{"data1": "123",  "data2": "456", "hashid":"O6jN0Z7VARqDzbpBHYOakeowE5Xnr41g"}
```

hashids_decode 函数

函数定义

对已使用 Hashids 库编码的数据进行解码。

语法描述

```
hashids_decoding(hash_value, salt="", min_length=0)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
hash_value	待解码的字符串	String	是	-	-
salt	为数据添加盐值，默认为空字符串	String	是	""	-
min_length	生成的哈希值的最小长度，默认为0	Number	是	0	-

示例

原始日志：

```
{"hashid":"O6jN0Z7VARqDzbpBHYOakeowE5Xnr41g"}
```

加工规则：

```
fields_set("result", hashids_decode(v("hashid"), salt="test", min_length=32))
```

加工结果：

```
{"result":"[123, 456]","hashid":"O6jN0Z7VARqDzbpBHYOakeowE5Xnr41g"}
```

IP 解析函数

最近更新时间：2025-06-25 19:05:52

geo_parse 函数

函数定义

解析出 IP 的地理位置。

语法描述

```
geo_parse(字段值, keep=("country","province","city"), ip_sep=",")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
data	IP 值，支持多个 IP 通过分隔符分隔	string	是	-	-
keep	要留存的字段	string	否	("country","province","city","provider","geo")。 geo 仅支持返回中国大陆城市的经纬度。	-
ip_sep	匹配结果中的第几个表达式	string	否	-	-

示例

- **示例1**

  原始日志：

  ```
  {
      "ip": "120.235.55.160"
  }
  ```

  加工规则：

  ```
  fields_set("result", geo_parse(v("ip")))
  ```

  加工结果：

  ```
  {
  "ip":"120.235.55.160"
  "result":"{"country":"中国","province":"广东省","city":"东莞市","provider":"中国移动","latitude":"23.027","longitude":"113.758"}"
  }
  ```

- **示例2**

  原始日志：

  ```
  {"ip":"101.132.57.150,101.14.57.157"}
  ```

  加工规则：

  ```
  fields_set("result", geo_parse(v("ip"),keep="province,city",ip_sep=","))
  ```

加工结果：

```
{"ip":"101.132.57.150,101.14.57.157", "result":"{\"101.14.57.157\":{\"province\":\"台湾省\",\"city\":\"台北市\"},\"101.132.57.150\":{\"province\":\"上海市\",\"city\":\"上海市\"}}"}
```

is_subnet_of 函数

函数定义

判断 IP 是否在该网段内，支持多个网段。

语法描述

```
is_subnet_of(网段列表, ip)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
网段列表	网段，支持多个网段，通过','分隔	String	是	-	-
IP	待判断的 IP	String	是	-	-

示例

- 示例1

  原始日志：

  ```
  {"ip": "192.168.1.127"}
  ```

  加工规则：

  ```
  log_keep(is_subnet_of("192.168.1.64/26",v("ip")))
  ```

  加工结果：

  ```
  {"ip": "192.168.1.127"}
  ```

- 示例2

  原始日志：

  ```
  {"ip": "192.168.1.127"}
  ```

  加工规则：

  ```
  fields_set("is_subnet",is_subnet_of("192.168.1.64/26",v("ip")))
  ```

  加工结果：

  ```
  {"ip": "192.168.1.127", "is_subnet":"true"}
  ```

- 示例3

  原始日志：

  ```
  {"ip": "192.168.1.127"}
  ```

加工规则：

```
fields_set("is_subnet",is_subnet_of("172.16.0.0/16",v("ip")))
```

加工结果：

```
{"ip": "192.168.1.127", "is_subnet":"false"}
```

- **示例4**

原始日志：

```
{"ip": "192.168.1.127"}
```

加工规则：

```
fields_set("is_subnet",is_subnet_of("172.16.0.0/16,192.168.1.64/26",v("ip")))
```

加工结果：

```
{"ip": "192.168.1.127", "is_subnet":"true"}
```

资源函数

最近更新时间：2025-06-24 18:55:32

res_local 函数

函数定义

使用 res_local 函数在当前数据加工任务中拉取高级配置中的环境变量。

语法描述

```
res_local(param, default=None, type="auto")
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
param	对应高级配置中的环境变量的字段名	String	是	-	-
default	字段值不存在时，返回该参数的值，默认值为 None	String	是	None	-
type	数据输出时的数据格式。 • auto（默认值）：将原始值转化为 JSON 格式。如果转换失败则返回原始值 • JSON：将原始值转化为 JSON 格式。如果转换失败则返回 default 参数的值 • raw：返回原始值	String	是	auto	-

示例

在数据加工的高级配置中，添加了1个环境变量字段为 time_session，值为30。
原始日志：

```
{}
```

加工规则：

```
fields_set("time_session", res_local("time_session"))
```

加工结果：

```
{"time_session":"30"}
```

res_rds_mysql 函数

函数定义

使用 res_rds_mysql 函数从云 MySQL 数据库中拉取数据库表内容或 SQL 执行结果。首次会默认拉取 MySQL 全量数据。该函数可以和 t_table_map() 函数一起使用，用于实现维度表关联的效果。

语法描述

```
res_rds_mysql(alias, database="数据库的名称", sql="select name from person_info", refresh_interval=0,
base_retry_back_off=1, max_retry_back_off=60, update_time_key=None, use_ssl=False)
```

参数说明

参数名称	参数描述	参数类型	是否必须	参数默认值	参数取值范围
alias	配置的数据库信息别名	string	是	-	-
database	数据库名称	string	是	-	-
sql	获取数据的 SQL 语句	string	是	-	-
refresh_interval	拉取数据的时间间隔，单位：秒。默认值为0，表示仅全量拉取一次	number	否	0	-
base_retry_back_off	拉取数据失败后重新拉取的时间间隔，默认值为1，单位：秒	number	否	0	-
max_retry_back_off	拉取数据失败后重试请求的最大时间间隔。默认值为60，单位：秒，建议使用默认值	number	否	60	-
update_time_key	用于增量获取数据。如果不配置此参数，则执行全量更新	string	否	-	-
use_ssl	是否使用 SSL 协议进行安全连接。	bool	否	False	-

示例

MySQL 中新建一个表 test，添加如下数据：

user_id	gameid	game
3	123	wangzhe
2	456	majiang

原始日志：

```
[
    {
        "user_id": 1
    },
    {
        "user_id": 3
    }
]
```

加工规则：

```
//在控制台将外部数据 MySQL 的别名配置为 hm，mysql 的 db 为 test222，表名为 test
//从 MySQL 拉取全部数据，使用 t_table_map 函数完成维度表关联

t_table_map(
    res_rds_mysql(alias="hm",database="test222",sql="select * from test"),
    "user_id",
    ["gameid", "game"]
)
```

加工结果：

```
[
{
"user_id":"1"
},
{
"game":"wangzhe",
```

```
"gameid":"123",
"user_id":"3"
}
]
```

加工实战案例
案例总览

最近更新时间：2024-12-19 16:03:12

操作场景

您可以参考本文提供的案例，以获得对数据加工的整体和直观的了解，同时也可以复制这些案例中的函数，来完成自己的数据加工。

注意事项

- v 函数：v("字段 A")，表示 value of “字段 A”，即字段 A 的值。在数据加工函数的参数中，有些是字段，例如 fields_drop("A 字段")，有些是**字段值**，此时需要使用 v 函数取值，例如 str_len(v("A 字段"))。常见的错误是：参数为字段值，却没有使用 v 函数取字段值，导致函数执行失败。
- **日志分发**：需要提前配好**目标日志主题**及其**目标名称**，目标名称用于**分发函数**。
- fields_set 函数：通过 fields_set 函数设置字段值，保存数据加工函数处理后的内容。例如 `fields_set("A+B",op_add(v("字段A"),v("字段B")))`，意思是将字段 A 的值和字段 B 的值相加，op_add 函数需要借助 fields_set 函数来完成结果的写入和保存。

实践案例

您可参考以下案例，完成自己的数据加工。

- 日志结构化-单行文本
- 日志结构化- JSON
- 日志结构化-多种格式
- 日志结构化-分隔符
- 日志结构化-Grok
- 日志过滤
- 日志分发
- 日志脱敏

版权所有：腾讯云计算（北京）有限责任公司

日志结构化-单行文本

最近更新时间：2024-12-16 15:47:32

场景描述

某用户将日志采集到日志服务（Cloud Log Service，CLS），是单行文本格式。现在用户想将日志结构化，从文本中**提取日志时间、日志级别、requestid**等信息，便于后续的检索分析。

场景分析

梳理用户的加工需求，加工思路如下：

1. {...}中的内容是**操作**的详情，可以通过正则提取。
2. 使用正则提取**日志时间、日志级别、URL**。

原始日志

```
{
    "content": "[2021-11-24 11:11:08,232][328495eb-b562-478f-9d5d-3bf7e][INFO] curl -H 'Host: '
http://abc.com:8080/pc/api -d {\"version\": \"1.0\",\"user\": \"CGW\",\"password\": \"123\"}"
}
```

加工结果

```
{
"level":"INFO",
"password":"123",
"requestid":"328495eb-b562-478f-9d5d-3bf7e",
"time":"2021-11-24 11:11:08,232",
"user":"CGW",
"version":"1.0"
}
```

DSL 加工函数

```
ext_regex("content", regex="\[(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2},\d{3})\]\[(\w+-\w+-\w+-\w+-\w+)\]\[([A-
Z]{4})\].*(\{[^\}]+\})", output="time,requestid,level,msg")
ext_json("msg")
fields_drop("content","msg")
```

DSL 加工函数详解

1. 使用正则键值提取函数 ext_regex，提取时间、请求、日志级别、消息等信息，并将相关的内容保存至 time、requestid、level 和 msg 字段中。

   ```
   ext_regex("content", regex="\[(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2},\d{3})\]\[(\w+-\w+-\w+-\w+-\w+)\]\
   [([A-Z]{4})\].*(\{[^\}]+\})", output="time,requestid,level,msg")
   ```

2. 使用 ext_json 对 msg 字段进行平铺，msg 的值是{\"version\": \"1.0\",\"user\": \"CGW\",\"password\": \"123\"}。

   ```
   ext_json("msg")
   ```

3. 丢弃 **content** 和 **msg** 字段。

   ```
   fields_drop("content","msg")
   ```

版权所有：腾讯云计算（北京）有限责任公司

日志结构化-JSON

最近更新时间：2024-11-22 16:58:22

版权所有：腾讯云计算（北京）有限责任公司

场景描述

小王将日志以 JSON 格式采集到日志服务（Cloud Log Service，CLS），有以下两种情况：
1. JSON 是多层嵌套，小王想提取 user 和 App 字段，其中 user 是二级嵌套字段。
2. 小王的 JSON 日志是数组，需要数组中拆分出来多条日志。

原始日志

日志1：嵌套 JSON

```
[
    {
        "content": {
            "App": "App-1",
            "start_time": "2021-10-14T02:15:08.221",
            "resonsebody": {
                "method": "GET",
                "user": "Tom"
            },
            "response_code_details": "3000",
            "bytes_sent": 69
        }
    },
    {
        "content": {
            "App": "App-2",
            "start_time": "2222-10-14T02:15:08.221",
            "resonsebody": {
                "method": "POST",
                "user": "Jerry"
            },
            "response_code_details": "2222",
            "bytes_sent": 1
        }
    }
]
```

日志2：JSON 数组

```
{
    "timestamp": 1732099684144000,
    "topic": "log-containers",
    "records": [
        {
            "category": "kube-request",
            "log": "{\"requestID\":\"12345\",\"stage\":\"Complete\"}"
        },
        {
            "category": "db-request",
            "log": "{\"requestID\":\"67890\",\"stage\":\"Response\"}"
        }
    ]
}
```

加工结果

日志1

```
[{"App":"App-1","user":"Tom"},
{"App":"App-2","user":"Jerry"}]
```

日志2

```
[
{
"category":"kube-request",
"requestID":"12345",
"stage":"Complete",
"timestamp":"1732099684144000",
"topic":"log-containers"
},
{
"category":"db-request",
"requestID":"67890",
"stage":"Response",
"timestamp":"1732099684144000",
"topic":"log-containers"
}
]
```

DSL 加工函数

日志1：加工语句

```
//使用 ext_json 函数从 JSON 数据中提取结构化数据，默认会平铺所有的字段
ext_json("content")
//丢弃 content 字段
fields_drop("content")
//丢弃不需要的字段 bytes_sent,method,response_code_details,start_time
fields_drop("bytes_sent","method","response_code_details","start_time")
```

日志2：加工语句

```
//从数组中拆分日志,拆出来2条日志
log_split_jsonarray_jmes("records")
//丢弃原始字段records
fields_drop("records")
//展开log的KV对
ext_json("log")
//丢弃原始字段log
fields_drop("log")
```

日志结构化-多种格式

最近更新时间：2024-10-18 10:20:11

场景描述

小王把**用户操作和结果**的日志，以单行文本的格式采集到日志服务（Cloud Log Service，CLS），日志的格式和内容**不完全相同**。小王想编写一套语句，对不同格式的日志进行结构化。
经过梳理查看，日志基本分为三种格式：第一种包含有 **uin、requestid、action、Reqbody** 四个字段、第二种包含 **uin、requestid、action** 三个字段、第三种包含 **requestid、action、TaskId** 三个字段。

场景分析

梳理一下小王的加工需求，加工思路如下：
1. 由于三种格式的日志都包含 **requestid、action** 字段，因此使用正则提取 **requestid** 和 **action** 字段。
2. 对 **uin、reqbody、TaskId** 字段进行特殊处理，先判断这个字段是否存在，如果存在，再进行提取。

原始日志

```
[
    {
        "__CONTENT__": "2021-11-29 15:51:33.201 INFO request 7143a51d-caa4-4a6d-bbf3-771b4ac9e135 action:
Describe uin: 15432829 reqbody {\"Key\": \"config\",\"Values\": \"appisrunnning\",\"Action\":
\"Describe\",\"RequestId\": \"7143a51d-caa4-4a6d-bbf3-771b4ac9e135\",\"AppId\": 1302953499,\"Uin\":
\"100015432829\"}"
    },
    {
        "__CONTENT__": "2021-11-2915: 51: 33.272 ERROR request 2ade9fc4-2db2-49d8-b3e0-a6ea78ce8d96 has
error action DataETL uin 15432829"
    },
    {
        "__CONTENT__": "2021-11-2915: 51: 33.200 INFO request 6059b946-25b3-4164-ae93-9178c9e73d75 action:
UploadData hUWZSs69yGc5HxgQ TaskId 51d-caa-a6d-bf3-7ac9e"
    }
]
```

DSL 加工函数

```
fields_set("requestid",regex_select(v("__CONTENT__"),regex="request [A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-
9]+-[A-Za-z0-9]+-[A-Za-z0-9]+",index=0,group=0))
fields_set("action",regex_select(v("__CONTENT__"),regex="action: \S+|action \S+",index=0,group=0))
t_if(regex_match(v("__CONTENT__"),regex="uin",
full=False),fields_set("uin",regex_select(v("__CONTENT__"),regex="uin: \d+|uin \d+",index=0,group=0)))
t_if(regex_match(v("__CONTENT__"),regex="TaskId",
full=False),fields_set("TaskId",regex_select(v("__CONTENT__"),regex="TaskId [A-Za-z0-9]+-[A-Za-z0-9]+-[A-
Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+",index=0,group=0)))
t_if(regex_match(v("__CONTENT__"),regex="reqbody",
full=False),fields_set("requestbody",regex_select(v("__CONTENT__"),regex="reqbody \{[^\}]+\}")))
t_if(has_field("requestbody"),fields_set("requestbody",str_replace(v("requestbody"),old="reqbody",new="")))
)
fields_drop("__CONTENT__")
fields_set("requestid",str_replace(v("requestid"),old="request",new=""))
t_if(has_field("action"),fields_set("action",str_replace(v("action"),old="action:|action",new="")))
t_if(has_field("uin"),fields_set("uin",str_replace(v("uin"),old="uin:|uin",new="")))
t_if(has_field("TaskId"),fields_set("TaskId",str_replace(v("TaskId"),old="TaskId",new="")))
```

DSL 加工函数详解

1. 新建一个字段 **requestid**，使用正则公式匹配 “**request 7143a51d-caa4-4a6d-bbf3-771b4ac9e135**”。

```
fields_set("requestid",regex_select(v("__CONTENT__"),regex="request [A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+",index=0,group=0))
```

2. 新建一个字段 action，使用正则公式匹配 “action: UploadData” 或者 “action DataETL”。因为原始日志中两种格式都有。

```
fields_set("action",regex_select(v("__CONTENT__"),regex="action: \S+|action \S+",index=0,group=0))
```

- 如果 __CONTENT__ 字段中有 uin 这个字符，新建 uin 字段，使用正则"uin: \d+|uin \d+"来匹配uin: 15432829或者uin 15432829。

```
t_if(regex_match(v("__CONTENT__"),regex="uin",
full=False),fields_set("uin",regex_select(v("__CONTENT__"),regex="uin: \d+|uin \d+",index=0,group=0)))
```

- 如果有 TaskId 这个字符，新建 TaskId 字段，使用正则来匹配 “TaskId 51d-caa-a6d-bf3-7ac9e”。

```
t_if(regex_match(v("__CONTENT__"),regex="TaskId",
full=False),fields_set("TaskId",regex_select(v("__CONTENT__"),regex="TaskId [A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+",index=0,group=0)))
```

- 如果有 reqbody 这个字符，新建 requestbody 字段，使用正则来匹配 reqbody{...}。

```
t_if(regex_match(v("__CONTENT__"),regex="reqbody",
full=False),fields_set("requestbody",regex_select(v("__CONTENT__"),regex="reqbody \{[^\}]+\}")))
```

3. 丢弃__CONTENT__字段。

```
fields_drop("__CONTENT__")
```

以上我们已经提取出了我们需要的字段，但是由于在正则匹配过程中，产生了多余的字符 action、uin、requestbody、requestid、TaskId。所以需要使用 str_replace() 函数将多余的字符去掉，并且使用 fields_set() 函数对字段值进行重置。

1. 如果有 requestbody 字段，将字段值中多余的 reqbody 字符去掉。

```
t_if(has_field("requestbody"),fields_set("requestbody",str_replace(v("requestbody"),old="reqbody",new="")))
```

2. 将字段值v("requestid")中多余的 requestid 去掉。因为每条日志中都有 requestid，所以没有判断这个字段是否存在。

```
fields_set("requestid",str_replace(v("requestid"),old="request",new=""))
```

3. 如果有 action 字段，那么字段值中多余的 action: 或 action 字符去掉。

```
t_if(has_field("action"),fields_set("action",str_replace(v("action"),old="action:|action",new="")))
```

4. 如果有 uin 字段，那么字段值中多余的 uin: 或者 uin 去掉。

```
t_if(has_field("uin"),fields_set("uin",str_replace(v("uin"),old="uin:|uin",new="")))
```

5. 如果有 TaskId 字段，那么字段值中多余的 TaskId 字符去掉。

```
t_if(has_field("tTaskId"),fields_set("TaskId",str_replace(v("TaskId"),old="TaskId",new="")))
```

加工结果

```
[
{"action":" Describe","requestid":" 7143a51d-caa4-4a6d-bbf3-771b4ac9e135","requestbody":" {\"Key\":
\"config\",\"Values\": \"appisrunnning\",\"Action\": \"Describe\",\"RequestId\": \"7143a51d-caa4-4a6d-
bbf3-771b4ac9e135\",\"AppId\": 1302953499,\"Uin\": \"100015432829\"}","uin":" 15432829"},
{"action":" DataETL","requestid":" 2ade9fc4-2db2-49d8-b3e0-a6ea78ce8d96","uin":" 15432829"},
{"action":" UploadData","requestid":" 6059b946-25b3-4164-ae93-9178c9e73d75","TaskId":" 51d-caa-a6d-bf3-
7ac9e"}
]
```


版权所有：腾讯云计算（北京）有限责任公司

日志结构化-分隔符

最近更新时间：2025-06-25 19:05:52

场景描述

场景1：用户的日志使用 | 进行分隔，现需要在其中提取结构化的字段。
场景2：用户将 Flink 任务运行的日志，以单行文本采集到日志服务（Cloud Log Service，CLS）。其中有一段 JSON，是 Flink 返回的包，用户想将这个 JSON 结构化。

场景1

原始日志

```
{
    "message": "2021-12-09 11:34:28.279|INFO|605c643e29e4|BIN--Python|192.168.1.1"
}
```

加工语句

```
//使用|作为分隔符(竖线需要转义)，从 message 中提取 time、loglevel、taskId、ProcessName、ip 字段.
ext_sepstr("message","time,loglevel,taskId,ProcessName,ip",sep="\|")
//丢弃 message
fields_drop("message")
```

加工结果

```
{
"ProcessName":"BIN--Python",
"ip":"192.168.1.1",
"loglevel":"INFO",
"taskId":"605c643e29e4",
"time":"2021-12-09 11:34:28.279"
}
```

场景2

原始日志

```
{
    "regex": "2021-12-02 14:33:35.022 [1] INFO  org.apache.Load - Response:status: 200, resp msg: OK, resp
content: {    \"TxnId\": 58322,    \"Label\": \"flink_connector_20211202_1de749d8c80015a8\",
\"Status\": \"Success\",    \"Message\": \"OK\",    \"TotalRows\": 1,    \"LoadedRows\": 1,
\"FilteredRows\": 0,  \"CommitAndPublishTimeMs\": 16}"
}
```

DSL 加工函数

```
//使用逗号作为分隔符，将日志切成三段
ext_sepstr("regex", "f1, f2, f3", sep=",")
//仅保留 f3字段:即 resp content:及其后续的字符
fields_drop("regex","f1","f2")
//使用冒号来切割 f3字段值，仅保留{}和括号内的字符，存至 resp_content 字段，该字段是一个 json，是 flink 返回的消息包
ext_sepstr("f3", "f1,resp_content", sep=":")
fields_drop("f1","f3")
//对 resp_content 字段结构化
ext_json("resp_content", prefix="")
```

```
//丢弃原字段
fields_drop("resp_content")
```

加工结果

```
{
"CommitAndPublishTimeMs":"16",
"FilteredRows":"0",
"Label":"flink_connector_20211202_1de749d8c80015a8",
"LoadedRows":"1",
"Message":"OK",
"Status":"Success",
"TotalRows":"1",
"TxnId":"58322"
}
```

版权所有：腾讯云计算（北京）有限责任公司

日志结构化-Grok

最近更新时间：2024-10-18 11:45:21

场景描述

小王通过 Kafka 协议上传日志 将 Beats 采集的日志上报到 CLS，思路如下：

1. 使用 grok 函数对日志进行结构化。
2. 使用日志中 time 字段替换 CLS 的日志时间(__TIMESTAMP__)。



原始日志

```
  {
    "__FILENAME__": "",
    "__SOURCE__": "192.168.100.123",
    "message": "2024-10-11 15:32:10.003 DEBUG [gateway,746db87efd1bbcf5434cb9835c59e522,47c3036810e0c33b]
[scheduled-Thread-1] c.i.g.c.f.d.a.task.AppleHealthCheckTask"
  }
```

加工结果

```
{
"__FILENAME__":"",
"__SOURCE__":"192.168.100.123",
"__TIMESTAMP__":"1728631930003",
"level":"DEBUG",
"service":"gateway",
"spanid":"47c3036810e0c33b",
"time":"2024-10-11 15:32:10.003",
"traceid":"746db87efd1bbcf5434cb9835c59e522"
}
```

加工语句

```
// 使用grok函数从日志中提取时间time,日志级别level,service,traceid和spanid
ext_grok("message",grok="%{TIMESTAMP_ISO8601:time} %{DATA:level} \[%{DATA:service},%{DATA:traceid},%
{DATA:spanid}\]")
//删除message字段
fields_drop("message")
```

```
// custom_cls_log_time函数，使用新字段time替换CLS的日志时间（__TIMESTAMP__）
custom_cls_log_time(dt_to_timestamp(v("time"), zone="UTC+8"))
```

日志过滤

最近更新时间：2024-12-13 11:59:43

场景描述

小王将工具的日志采集到了 CLS，现需要将 cmdb 日志中操作方法为 PUT 和 POST 的日志过滤出来。

原始日志

```
[
    {
        "path": "/eks/pod/running",
        "clientIP": "1.*.*.123",
        "method": "POST"
    },
    {
        "path": "/cmdb/login",
        "clientIP": "1.*.*.123",
        "method": "PUT"
    },
    {
        "path": "/cmdb/start",
        "clientIP": "1.*.*.123",
        "method": "GET"
    }
]
```

加工语句

```
//path中包含有cmdb字符，保留该日志，其余日志过滤掉
log_keep(regex_match(v("path"),regex="cmdb",full=False))
//method中包含POST或者PUT字符，保留该日志
log_keep(regex_match(v("method"),regex="POST|PUT",full=False))
```

加工结果

```
{
"clientIP":"1.*.*.123",
"method":"PUT",
"path":"/cmdb/login"
}
```

日志分发

最近更新时间：2024-12-16 10:03:23

场景描述

某用户将日志采集到日志服务（Cloud Log Service，CLS），日志通过双竖线 `||` 分割，有日志的时间、日志级别、任务 ID、进程名称、主机 IP 等。现在用户想将日志结构化，并按照 ERROR、WARNING、INFO 三个级别，把日志**分发**到三个不同的目标日志主题中，便于后续的分析。

场景分析

- 日志结构化：按照**双竖线** `||` 分割符，对日志进行结构化。
- 日志分发：按照 ERROR、WARNING、INFO 三个级别，把日志分发到三个不同的目标日志主题。

> **说明：**
> 在 新建数据加工任务 时，目标日志主题选择**固定日志主题**，并提前定义目标名称。该名称将用在 log_output **("目标名称")** 函数中。



原始日志

```
[
    {
        "message": "2021-12-09 11:34:28.279||INFO||605c643e29e4||BIN--Python||192.168.1.1"
    },
    {
        "message": "2021-12-09 11:35:28.279||WARNING||615c643e22e4||BIN--Java||192.168.1.1"
    },
    {
        "message": "2021-12-09 11:36:28.279||ERROR||635c643e22e4||BIN--Go||192.168.1.1"
    }
]
```

DSL 加工函数

```
//使用||作为分隔符(竖线需要转义),从message中提取time,loglevel,taskId,processName,ip字段.
ext_sepstr("message","time,loglevel,taskId,ProcessName,ip",sep="\|\|")

//丢弃message
fields_drop("message")

//使用switch语句进行分发
t_switch(

//loglevel为"INFO",输出到info_log
op_str_eq(v("loglevel"),"INFO"),log_output("info_log"),
//loglevel为"WARNING",输出到warning_log
op_str_eq(v("loglevel"),"WARNING"),log_output("warning_log"),
```

```
//loglevel为"ERROR",输出到error_log
op_str_eq(v("loglevel"),"ERROR"),log_output("error_log")

)
```

加工结果

error_log 中:

```
{
"processName":"BIN--Go",
"ip":"192.168.1.1",
"loglevel":"ERROR",
"taskId":"635c643e22e4",
"time":"2021-12-09 11:36:28.279"
}
```

warning_log 中:

```
{
"processName":"BIN--Java",
"ip":"192.168.1.1",
"loglevel":"WARNING",
"taskId":"615c643e22e4",
"time":"2021-12-09 11:35:28.279"
}
```

info_log 中:

```
{
"processName":"BIN--PYTHON",
"ip":"192.168.1.1",
"loglevel":"INFO",
"taskId":"605c643e29e4",
"time":"2021-12-09 11:34:28.279"
}
```

> **注意**
> 必须要提前配置好目标日志主题和目标名称。

日志脱敏

最近更新时间：2025-07-01 16:21:52

场景1

场景描述

某公司的日志，部分日志中可能有用户姓名、身份证号、银行卡号等敏感信息，为优化日志写入规范，需要对日志中的敏感数据进行检测和统计。

原始日志

```
[
    {
        "operation_time": "2025-04-12 15:32:01",
        "user_info": "张三申请开通服务，身份证号 522301199201230123",
        "operation_detail": "成交金额200元"
    },
    {
        "operation_time": "2025-04-12 15:32:01",
        "user_info": "李四申请开通服务，身份证号 522301199201230123，银行卡号 622888012345678 9123",
        "operation_detail": "成交金额100元"
    },
    {
        "operation_time": "2025-04-12 15:32:01",
        "user_info": "王五申请开通服务，身份证号 522***1992****0123，银行卡号 6228****2345****123",
        "operation_detail": "成交金额500元"
    }
]
```

场景分析

使用数据加工，从原始日志中，检测身份证号、银行卡号等敏感信息，发现敏感信息后，在该日志中新增一个字段 SENSITIVE_FLAGS，该字段用于填充发现的 敏感信息项，并将加工结果写入目标日志主题。后续，您可在目标日志主题中，使用 SQL 统计敏感信息的类型和个数。

DSL 加工函数

```
//scope=ALL_FIELDS 意为搜索整条日志中的所有字段
//sample_ratio=1意为对所有的日志进行敏感信息检测，0.5代表对50%的日志进行检测
sensitive_detection(scope="ALL_FIELDS", sample_ratio=1, discover_items="CHINA_IDCARD,DEBIT_CARD")
```

加工结果

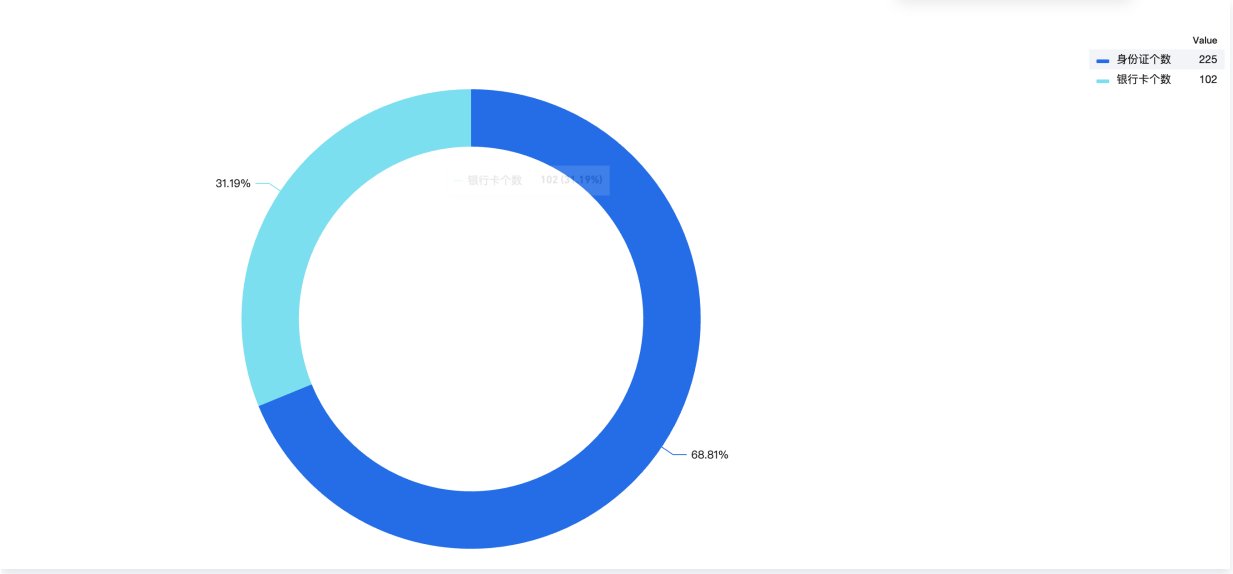
```
[
//第一条数据中只有身份证敏感项
{
"SENSITIVE_FLAGS":"CHINA_IDCARD",
"operation_detail":"成交金额200元",
"operation_time":"2025-04-12 15:32:01",
"user_info":"张三申请开通服务，身份证号 522301199201230123"
},
//第二条数据中有身份证、银行卡两种敏感项
{
"SENSITIVE_FLAGS":"CHINA_IDCARD,DEBIT_CARD",
"operation_detail":"成交金额100元",
"operation_time":"2025-04-12 15:32:01",
"user_info":"李四申请开通服务，身份证号 522301199201230123，银行卡号 6228880123456789123"
},
//第三条数据是安全的，因此标记为SAFE
```

```
{
"SENSITIVE_FLAGS":"SAFE",
"operation_detail":"成交金额500元",
"operation_time":"2025-04-12 15:32:01",
"user_info":"王五申请开通服务，身份证号 522***1992****0123，银行卡号 6228****2345****123"
}
]
```

敏感项统计分析

```
select
    case when
           SENSITIVE_FLAGS like '%CHINA_ID%'
         then '身份证个数'
         when
           SENSITIVE_FLAGS like '%DEBIT_CARD%'
         then '银行卡个数' else 'other'
    end as type ,
    count(*) as "敏感项"
    group by type HAVING type != 'other'
 limit 10000
```

统计分析结果：



场景2

场景描述

某用户将日志采集到日志服务（Cloud Log Service，CLS），日志数据中含有用户 ID（dev@12345）、登录的 IP 地址（11.111.137.225）和手机号码（13912345678），用户想将这些敏感信息脱敏。

场景分析

该日志本身是一个结构化的日志，因此可以直接对字段进行脱敏处理。

原始日志

```
    {
        "Id": "dev@12345",
        "Ip": "11.111.137.225",
        "phonenumber": "13912345678"
```

```
    }
```

DSL 加工函数

```
fields_set("Id",regex_replace(v("Id"),regex="\d{3}", replace="***",count=0))
fields_set("Id",regex_replace(v("Id"),regex="\S{2}", replace="**",count=1))
fields_set("phonenumber",regex_replace(v("phonenumber"),regex="(\d{0,3})\d{4}(\d{4})",
replace="$1****$2"))
fields_set("Ip",regex_replace(v("Ip"),regex="(\d+\.)\d+(\.\d+\.\d+)", replace="$1***$2",count=0))
```

DSL 加工函数详解

1. 对 **Id** 字段进行脱敏处理，结果为 dev@***45。

```
fields_set("Id",regex_replace(v("Id"),regex="\d{3}", replace="***",count=0))
```

2. 对 **Id** 字段进行二次脱敏处理，结果为**v@***45。

```
fields_set("Id",regex_replace(v("Id"),regex="\S{2}", replace="**",count=1))
```

3. 对 **phonenumber** 字段进行脱敏处理，将中间的4位数替换为****，结果为139****5678。

```
fields_set("phonenumber",regex_replace(v("phonenumber"),regex="(\d{0,3})\d{4}(\d{4})",
replace="$1****$2"))
```

4. 对 **IP** 字段进行脱敏处理，将第二段替换为***，结果为11.***.137.225。

```
fields_set("Ip",regex_replace(v("Ip"),regex="(\d+\.)\d+(\.\d+\.\d+)", replace="$1***$2",count=0))
```

加工结果

```
{"Id":"**v@***45","Ip":"11.***.137.225","phonenumber":"139****5678"}
```

Logstash 语句参照表

最近更新时间：2024-11-07 09:37:32

部分 Logstash 语句，在数据加工中对应的函数如下表。

场景	Logstash		数据加工
重命名字段	mutate	mutate { rename => {"old_field_name" => "new_field_name" } }	fields_rename("old_field_name","new_field_name")

删除字段		mutate {
		remove_field => ["password_hash"]
		}
			fields_drop("password_hash")

更新字段值		mutate {
		update => {"status_code" => "Not Found"
		} } //将status_code更新为Not Found
			fields_set("status_code", "Not Found")

提取键值-Grok	grok	grok { match => { "message" => "%{TIMESTAMP_ISO8601:time} %{LOGLEVEL:level} " }}	ext_grok("message",grok="%{TIMESTAMP_ISO8601:time} %{DATA:level}")

提取键值-分隔符	split	mutate {
		split => { "message" => "|" }
		add_field => {
		"time" => "%{[message][0]}"
		"level" => "%{[message][1]}"
		"taskId" => "%{[message][2]}"
		"ProcessName" => "%{[message][3]}"
		"ip" => "%{[message][4]}"
		}
		remove_field => ["message"]
		}
			ext_sepstr("message","time,loglevel,taskId,ProcessName,ip",sep="|")
			fields_drop("message")

提取键值-JSON	json	json {
		source => "message"
		target => "parsed_data"
		}
			ext_json("message")

删除日志	drop	if [status] == 404 { //如果status=404
		drop { } //删除日志
		}
			log_drop(删除日志
			op_eq(v("status"),404)// 当status的值=404
			)

逻辑判断	if else	• if [log] //如果存在log字段
		• if "Cost" in [message] //message字段值有"Cost"字符时
			t_if(has_field("log"))//如果存在log字段
			t_if(
			str_exist(v(message), "Cost", ignore_upper=True)
			)//message字段值有"Cost"字符时

	or , and	if "Cost" in [message] or "cost" in [message]	op_or(str_exist(v(message), "Cost", ignore_upper=False), str_exist(v(message), "cost", ignore_upper=False))

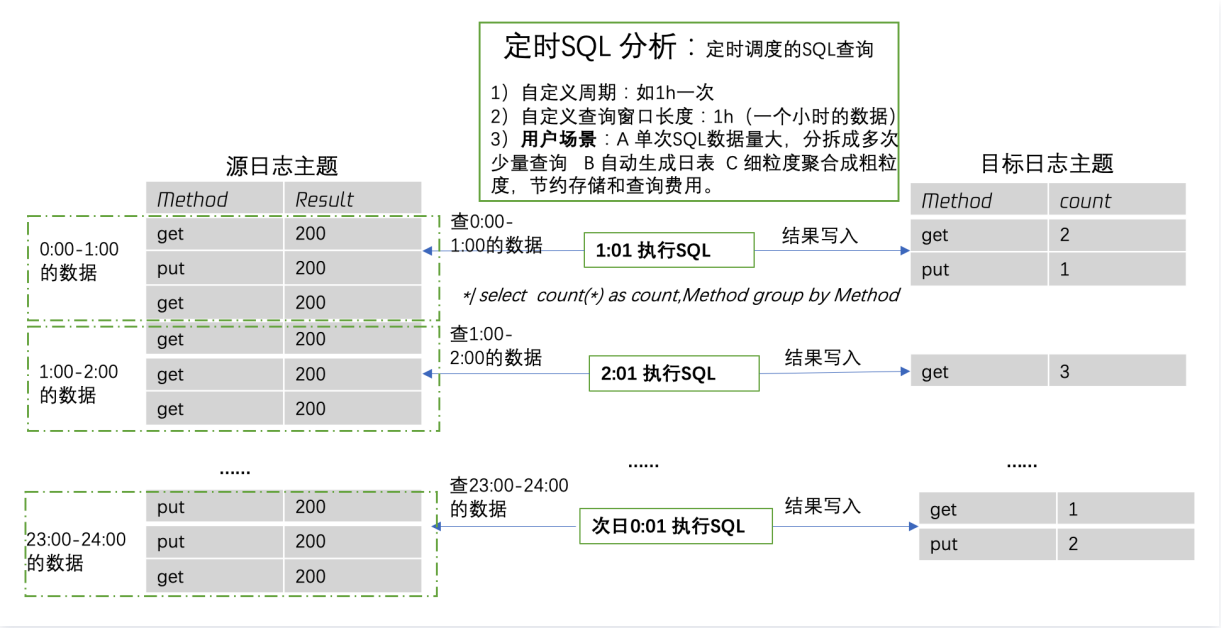
分发日志至多个sink(目标)	output	if [container] == "scm-pfc" {
		elasticsearch {
		hosts => ["xx.xx.x.xxx:9200"]
		index => "p-k8s"
		}
		
		} else {
		elasticsearch {
		hosts => ["xx.xx.x.xx:9200"]
		index => "p-container"
		}}
			t_if_else(
			op_str_eq(v("container"),"scm-pfc"),
			log_output("p-k8s"), //if 分支
			log_output("p-container")//else分支
			)

定时 SQL 分析

概述

最近更新时间：2024-12-26 18:12:22

定时 SQL 分析，可根据您定义的时间窗口、调度周期， 周期性地对日志数据进行查询，并将查询的结果保存到您指定的日志主题中，分析过程如下图所示：



基本概念

定时 SQL 分析相关的名词和概念，请您参见 基本概念 。

功能特性

- 日志的计算分析。基于原始日志，周期性地去执行查询，将结果保存到指定的日志主题，例如您的业务需求是生成日报表、周报表，就可以使用定时 SQL 分析。
- 日志聚合。可以省去大量的索引存储和日志存储费，例如，将1分钟的历史日志聚合成1小时的日志来保存，可以有效节约成本。
- 过滤日志。例如过滤出 loglevel 为 ERROR 的日志。

> **注意：**
> 定时 SQL 默认返回100条数据，如您需修改返回的数据条数，可参考 LIMIT 语法 主动定义 limit，最大可返回100万条结果。

费用说明

定时 SQL 分析免费。

规格限制

定时 SQL 分析的规格与限制，请您参见 规格与限制 。

新建任务

最近更新时间：2025-07-03 15:06:02

本文为您介绍如何新建定时 SQL 分析任务。

前提条件

- 已开通日志服务，源日志主题开启 键值索引 。
- 确保当前操作账号拥有配置定时 SQL 分析的权限，详情请参见 CLS 访问策略模板 。

操作步骤

1. 登录 日志服务控制台 ，在左侧导航栏中选择**定时 SQL 分析**，单击**新建**。
2. 配置定时 SQL 分析任务，其中**写入目标**可分为**指标主题**和**日志主题**。
   - 写入目标为指标主题：使用查询语句，将 日志转为指标 ，存在 指标主题 中。
   - 写入目标为日志主题：用于日志过滤、日志预聚合等场景。



写入目标为指标主题

使用场景：

预聚合。例如：日志10秒上报一次，如果业务需要查询60天的数据：`*| select count(*) group by ip, request_method`，IP 在60天内可能超出1亿条数据，由于维度太高，会查询失败。那么可以使用日志转指标，将日志预聚合为分钟级别的指标数据，然后再查询聚合后的指标主题就不会再超时。

转指标存储的优势：

- 查询费用低：指标仅收取存储和写流量费，无索引流量和索引存储费用（日志有）。同等数据量下，指标查询费用更低，请参见 产品定价 。
- 存储费用：尤其针对需要查询长周期时间范围的业务，转指标后，可将原始 日志沉降 至低频存储（低频不支持 SQL，可检索），进一步节约日志存储费。
- 查询速度更快。请参见 日志转指标 。

配置项：

配置项	说明
任务名称	定时 SQL 的任务名称，长度255字符，下划线，英文字母和数字组合。
启用状态	默认开启，开启/关闭任务。
服务日志	开启/关闭，打开之后，定时 SQL 任务运行的监控指标将写入服务日志，例如单次查询结果、查询时间等。

源日志主题	定时 SQL 任务的输入，即需要转换为指标的原始日志所在的日志主题。
写入目标	指标主题/日志主题。 定时 SQL 任务的输出，选择**指标主题**，将转换后的指标存储在指定的指标主题中。如当前没有指标主题，可 创建指标主题 。
执行语句	使用 SQL 语句 对原始日志进行统计以获取指标，例如： `* | select avg(cpu_usage)as cpu_usage ,avg(mem_usage) as mem_usage,namespace,cluster group by namespace,cluster` **说明：** 如果您修改了 SQL 语句，请单击**预览**，刷新 SQL 查询结果。
指标名称	多选，一般为数值类型。从下拉列表中选择字段作为指标名称，例如上面的例子中，cpu_usage 和 mem_usage 为指标名称，其值为指标值。
指标维度	多选，**不可选择**时间类型的字段作为指标维度。 从下拉列表中选择字段作为指标维度，例如上面的例子中，namespace、cluster 为维度，其值为维度值。可理解为按照不同的 cluster 和 namespace 组合来查看 cpu_usage 和 mem_usage。指标维度一般就是您的 SQL 语句中 group by 的维度字段（时间类型字段除外）。

自定义维度	新增指标的维度，维度值是静态的。例如：您原来的指标维度为
	`go_gc_duration_seconds{job="prometheus",quantile="0"}`
	添加自定义维度 `tcloud_region_name="ap-guangzhou"` 之后，维度变为：
	`go_gc_duration_seconds{job="prometheus", quantile="0",tcloud_region_name="ap-guangzhou" }`

时间戳	• 默认值：指标的默认时间戳为查询时间窗口的左侧时间，例如 查询时间窗口为5月7日00:00:00 – 5月7日00:10:00，那么该次计算生成的指标时间戳为5月7日00:00:00。 • 自定义时间戳：如果您在 SQL 语句中自定义了时间类型的字段，可以在下拉列表中，选择该字段为指标的指标戳。如果没有看到该时间类型字段，请单击 SQL 语句下方的**预览**，刷新下拉列表中的字段值。例如，您使用 检索分析语句 `* | select count(*) as cnt,histogram(__TIMESTAMP__,INTERVAL 15 SECOND) as time group by time limit 100` 每15秒计算一次日志条数，可以选择 time 字段为您的自定义时间戳。这样您在指标主题中，将每隔15秒生成一条指标数据。**注意：自定义时间戳的间隔最小为15秒。** **说明：** • 自定义时间戳支持选择 UNIX 时间、TimeStamp 的类型，精度为毫秒。例如：1629876543.123，2021-08-25T12:22:23.123+08:00。 • 自定义时间戳的最小间隔为15秒。例如15:00:00，15:00:15，15:00:30。
调度范围(Scope)	定时 SQL 任务处理的日志数据的范围，例如2023年1月1日00:00:00-2023年3月31日00:00:00的日志，如果结束时间不选择，表示任务将持续运行，将新采集的日志持续地转换为指标。
调度周期	取值范围1分钟 – 1440分钟，如您配置为 X，则意为每隔 X 分钟会发起一次日志查询，日志查询的结果将保存为指标。 推荐配置**1分钟**，更加及时的将日志转化为指标。
查询时间窗口(Query Window)	推荐配置 **@m-1m ~ @m**，即最近1分钟，每次查询一分钟的日志数据，并将其转化为指标。 **注意：** 转指标时，该参数切勿超过30分钟，否则可能导致转指标失败。 • 快速配置：如选择近5分钟，如果当前时间为17:30，那么查询时间窗口为17:25 – 17:30。 • 时间表达式：适用于相对时间的表达，如果指定时间表达式为 @m-10m 至 @m-5m，当前时间为17:30，那么查询时间窗口为17:20 – 17:25。
高级设置	查询延迟的时间，在控制台**高级设置**中，取值范围60秒 – 120秒。日志生成索引一般会有延迟，在索引生成之前，不可查询，因此设置60秒延迟查询，此时索引已生成（99.9%的索引数据将在5秒内生成）。

写入目标为日志主题

使用场景：

- **过滤日志：** `*| SELECT ip where region="guangzhou"`

> ⚠ **注意：**
> 定时 SQL 的执行会延迟1分钟，例如查询11:01-11:02的数据，查询会在11:03发起，且最多返回100W条数据，需评估是否符合您的场景。

- **预聚合：** 例如日志10秒上报一次，如果业务需要查询60天的数据：`*| SELECT ip , AVG(request_time) AS avg_request_time GROUP BY ip`，IP 在60天内可能超出1亿条数据，由于维度太高，会查询失败。那么可以使用定时 SQL，将秒级别日志预聚合为分钟级别数据，然后再查询聚合后的数据就不会再超时。

配置项：

配置项	说明
任务名称	定时 SQL 任务的名称，长度255字符，下划线，英文字母和数字组合。
启用状态	默认开启，开启/关闭任务。
服务日志	开启/关闭，打开之后，定时 SQL 任务运行的监控指标将写入服务日志，例如单次查询结果、查询时间等。
源日志主题	定时 SQL 任务的输入，需要被处理的原始日志。
写入目标	定时 SQL 任务的输出（1个），选择**日志主题**，建议**提前配置目标主题的索引**。
执行语句	使用 SQL 语句 对原始日志进行统计，例如：调度周期配置为1分钟，查询时间窗口配置为1小时，SQL 语句为 `*| select sum(active_user) as hourly_active_user`，表示为每分钟统计一次近一小时的活跃客户数。
调度范围(Scope)	定时 SQL 任务处理的日志数据的范围，例如2023年1月1日 00:00:00-2023年3月31日00:00:00的日志，如果不指定结束时间，则默认持续运行。
调度周期	取值范围1分钟 - 1440分钟，如您配置为1分钟，则意为每隔 1 分钟会发起一次日志查询。
查询时间窗口(Query Window)	查询时间窗口，单次日志查询的时间窗口. • 快速配置：如选择近5分钟，如果当前时间为17:30，那么查询时间窗口为17:25-17:30。 • 时间表达式：适用于相对时间的表达，如果指定时间表达式为 @m-10m 至 @m-5m，当前时间为17:30，那么查询时间窗口为17:20 - 17:25。
高级设置	查询延迟的时间，在控制台**高级设置**中，取值范围60秒 - 120秒。日志生成索引一般会有延迟，在索引生成之前，不可查询，因此设置60秒延迟查询，此时索引已生成（99.9%的索引数据将在5秒内生成）。

时间表达式语法

您在创建定时 SQL 分析时，可指定 SQL 时间窗口。定时 SQL 任务运行时，仅查询该 SQL 时间窗口内的日志。如下内容介绍 SQL 时间窗口相关的时间表达式语法。

操作符	说明
+	加号
-	减号
@	取整操作符，根据时间向下取整。例如以小时为单位对时间01:40进行取整，取整后为01:00。

时间单位	说明
h	时
m	分
s	秒

示例

时间表达式	说明
@m-15m 到 @m-5m	先减15分钟再向下取整到分钟。 例如，创建定时 SQL 任务时，配置调度间隔为每天00:00，延迟执行为30秒，SQL 时间窗口为[@m-15m,@m-5m），则表示在00:00:30时刻执行 SQL 任务，分析[23:45~23:55)期间的数据。
@h-1h 到 @h	先减1小时再向下取整到小时。 例如，创建定时 SQL 任务时，配置调度间隔为每天00:00，延迟执行为30秒，SQL 时间窗口为[@h-1h@h,@m-5m），则表示在00:00:30时刻执行 SQL 任务，分析[23:00~23:55)期间的数据。
@h-12h+5m 到 @m-5m	先减12小时再加5分钟，即减11小时55分钟。 例如，创建定时 SQL 任务时，配置调度间隔为每天00:00，延迟执行为30秒，SQL 时间窗口为[@h-12h+5m,@m-5m），则表示在00:00:30时刻执行 SQL 任务，分析[12:05~23:55)期间的数据。

版权所有：腾讯云计算（北京）有限责任公司

查看任务

最近更新时间：2024-11-14 09:29:12

操作场景

本文为您介绍如何查看定时 SQL 分析任务信息。

操作步骤

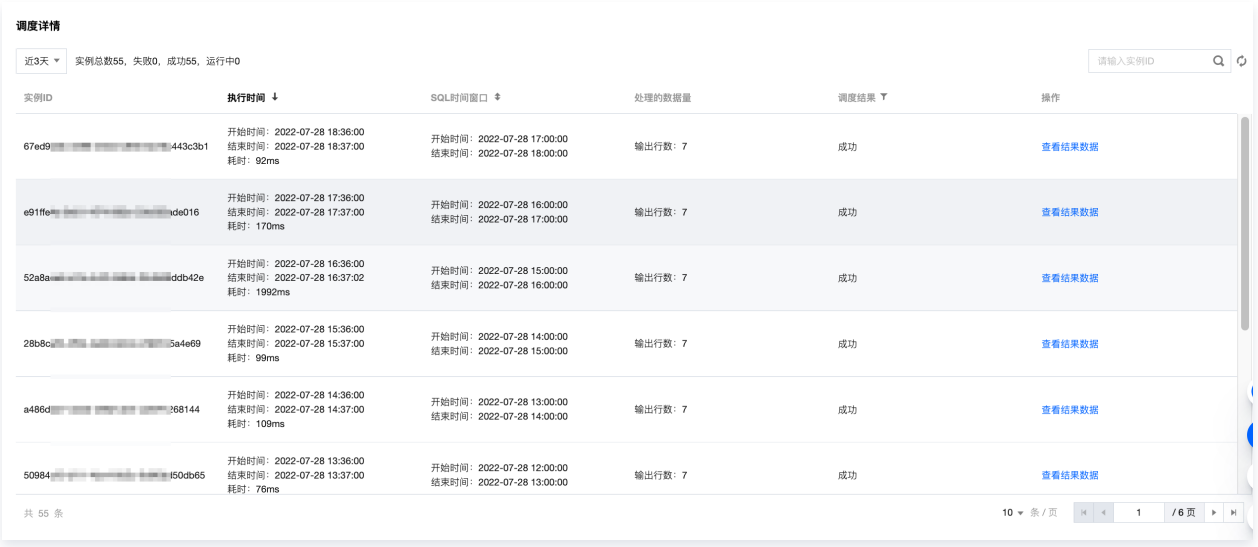
1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**数据处理 > 定时 SQL 分析**，在任务列表单击一个任务，即可查看任务基本信息、调度详情、任务监控。

   任务基本信息：
   - 任务的名称、任务状态、任务的 ID、任务创建时间、最近修改时间。
   - 源日志主题、目标日志主题、服务日志开关。
   - 调度周期、调度范围、SQL 时间窗口、执行语句。

调度详情

调度详情：每一次 SQL 查询的实例 ID、SQL 查询执行的时间、SQL 时间窗口、处理的数据量、调度结果。

> **说明：**
> SQL 执行时间：对于调度结果失败的 SQL 查询而言，由于查询会重试，因此执行时间中的**结束时间**为最后一次重试的时间，可能与**开始时间**的相差较大。



任务监控

> **说明：**
> 请在您的定时 SQL 任务中，打开服务日志开关，否则您**无法使用**下文中的**添加到** 监控告警 功能，仅能查看监控数据。

监控项添加告警：

您可在监控图表的右上角，单击添加到监控告警的图标，将监控指标添加到您的告警。例如可将 SQL 任务分析调度延时、SQL 分析耗时配置为您的监控告警项。



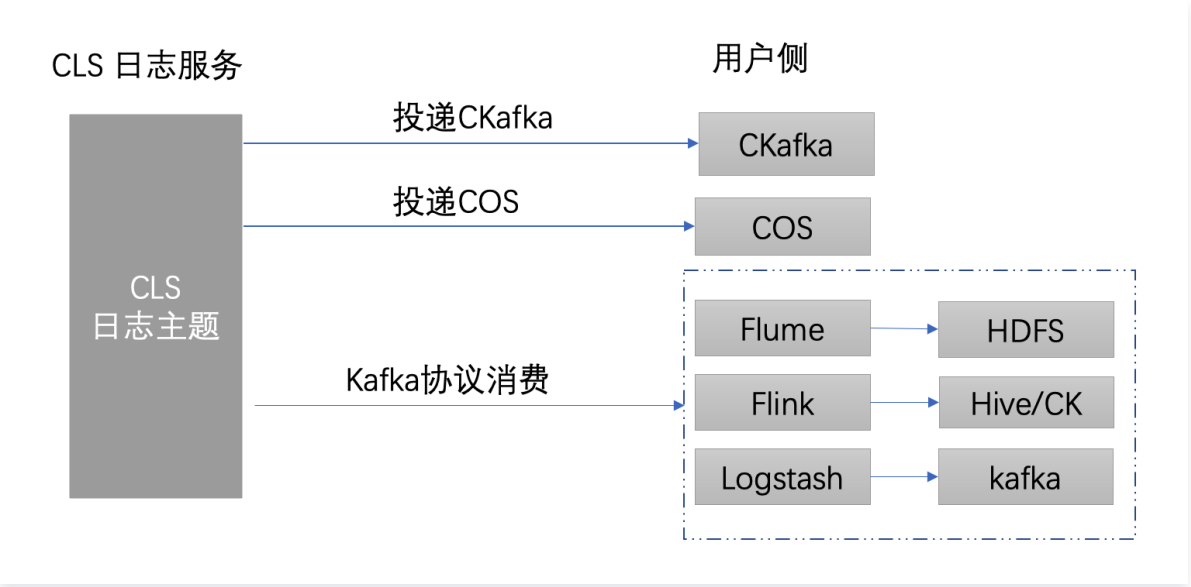
监控指标说明：

监控图表名称	单位	监控指标
处理日志量	行	• 输入行数。 • 输出行数。 • 写目标主题失败行数。 • 写目标主题成功行数。
SQL 分析耗时	毫秒/秒	单次 SQL 分析(查询)写入目标主题的时间-分析开始时间。 **例如**：某个 SQL 分析(查询)的结果写入目标主题的时间是15:01:00.900，分析开始时间(实际调度时间)是15:01:00.300，那么该次 SQL 分析总共耗时600毫秒。
SQL 分析调度延时	毫秒/秒	单次 SQL 分析调度的延时。 **例如**：用户配置的 SQL 分析每分钟调度一次，假定某个 SQL 分析(查询)按照计划是15:01:00调度，实际调度时间是15:01:00.100，那么调度的延时是100毫秒。
SQL 分析结果	个	SQL 分析的结果的统计。 定时 SQL 分析的结果有这几种：成功、语法错误、查询超时、内部错误、并发超限。 **例如**：您在右上角选择查询1天的数据，那么时间轴的步长是15分钟，如果您的定时 SQL 任务，调度周期是1分钟，即每分钟调度一次，假定全部调度成功，那么分析结果是成功15个。

投递与消费
投递与消费概述

最近更新时间：2024-08-15 09:57:01

日志服务（Cloud Log Service，CLS）可以将日志主题中的数据投递到腾讯云 COS（对象存储）、CKafka（消息队列），以满足归档或者计算的需求。投递和消费都会产生读流量费用。
下图是 CLS 日志在大数据中的常见链路：



- 日志归档场景，您可以将日志投递到 COS，默认是 COS 的标准存储。您也可以选择 CLS 日志沉降，不产生读流量费用，整体费用更低，同时可以继续使用全文和键值检索（如果您之前开启过全文、键值索引）。
- 日志数据的离线计算场景。您将日志投递到 COS 之后，可以使用 Hive 进行批处理，参见 使用 DLC（Hive）分析 CLS 日志。您也可以使用 CLS 的 定时 SQL 分析 对日志数据进行计算分析（结构化日志，开启键值索引），并将计算结果保存到您指定的日志主题中。
- 日志数据的实时计算场景。您可以将日志投递到 CKafka，下游使用大数据流处理。您也可以使用 kafka 协议消费，将 CLS 日志主题当做一个 kafka Topic 进行消费。这样可以省去购买 Kafka 实例的费用，投递和消费的读流量费用基本相同。

功能特性

产品功能	投递数据格式	使用场景
投递到 COS	CSV	可用于日志数据批处理。 可以利用 CSV 投递实现 投递日志原文（单行、多行、分隔符采集的日志）。
	JSON	日志数据计算。
	Parquet	日志数据需为结构化数据，支持数据类型转换（非单行、多行采集），多用于 Hive。
投递到 CKafka	原文	日志流处理。
	JSON	
Kafka 协议消费	原文	
	JSON	

注意
- 请提前购买 COS/CKafka。
- 若需要在投递前对日志进行清洗加工过滤，可参见使用 数据加工 操作。

产品优势

- 操作简单，在控制台完成配置即可将日志数据投递到腾讯云 COS 和 CKafka。
- 数据投递监控。您可以查看投递日志的流量、条数、投递任务时延的监控图表。

费用说明

投递会产生内网读流量费用（暂不支持跨地域投递），请参见 计费概述，单价参见 产品定价 。

规格与限制

您可参见 规格与限制 > 投递与消费 。

投递任务角色授权

最近更新时间：2025-06-26 18:56:52

操作场景

当您创建投递到 COS/CKafka/DLC 的任务时，因为涉及云产品之间的访问，需要您授权日志服务（Cloud Log Service，CLS）服务角色访问 COS/CKafka /DLC 的权限。您可通过控制台操作，系统会引导用户完成授权。

操作步骤

您在第一次通过控制台创建投递到 COS/CKafka/DLC 的任务时，页面会弹出授予权限的窗口，请根据页面指引创建角色和策略。

1. 在弹出的**该功能需创建服务角色**窗口中，单击**前往访问管理**。以下以投递 COS 为例说明，如下图所示：



2. 在**角色管理**页面中，单击**同意授权**，如下图所示：



至此您已完成 CLS 服务角色访问 COS/CKafka/DLC 的授权。若您是主账号，则可直接进行投递操作；若您是子账号或协作者账号，需要主账号授予您投递 COS/CKafka/DLC 的权限，授权步骤参考 基于 CAM 管理权限， 复制授权策略参考 自定义权限策略示例 。

投递到 COS
新建投递任务

最近更新时间：2025-07-01 16:21:52

操作场景

日志采集到 CLS 后，支持将日志数据投递至 COS 中进行存储或分析。本文介绍创建投递 COS 投递的操作步骤。

前提条件

1. 开通日志服务，创建 日志集 与 日志主题，并成功采集到日志数据。
2. 开通腾讯云对象存储服务，并且在待投递日志主题的地域已创建存储桶，详细配置请参见 创建存储桶 文档。
3. 子账号/协作者需要主账号授权，授权步骤参见 基于 CAM 管理权限，授权策略参见 CLS 访问策略模板 。
4. 已授权给腾讯云 CLS 服务角色访问 COS 的权限。大部分用户通过控制台操作时，系统会引导用户完成授权；小部分用户跨过控制台，直接调用 API，这部分客户需要手动去授权，详情请参见 投递任务角色授权 。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**日志主题**。
3. 单击需要投递的日志主题 ID/名称，进入日志主题管理页面。
4. 单击**投递到 COS** 页签，进入投递到 COS 配置页面，单击**添加投递配置**。
5. 如您**需对日志进行归档**，可使用 CLS 的 日志沉降 功能，将冷数据沉降至低频存储。沉降之后，存储费用可有效下降，但仍然可对日志进行检索。例如您可将总存储时间配置为180天，其中标准存储（热数据）30天，低频存储（冷数据）150天，即可满足日志的审计需求。如您**仍选择投递到 COS**，参考如下配置：

配置项	解释说明	规则	是否必填
投递任务名称	配置投递任务的名称。	-	必填
投递时间范围	• **开始时间**：您要投递的日志数据的开始时间。 默认给出该日志主题生命周期最早的时间点。 • **结束时间**：您要投递的日志数据的结束时间，不可选择未来时间，不填写表示持续投递日志。 **注意：** • 如果您的开始时间为历史时间，结束时间不填写，即持续投递，任务会同时投递历史日志和实时日志。 例如：您选择投递 2023年1月1日00:03 - ∞ 的数据，提交投递任务的时间为2023年2月13日 19:05，那么历史日志为2023年1月1日 00:03-2023年2月13日19:05，实时日志为2023年2月13日 19:05 - ∞，两种数据将同时向 COS 投递，您可在投递任务列表查看历史数据投递的进度和历史数据投递完成所需的时长。 • 任务提交以后，投递时间范围不支持修改。	时间选择	必填
投递文件大小	需要投递的原始日志文件的大小，和投递间隔时间配合使用，哪个条件先触发，就按照哪个规则去压缩文件，然后投递到 COS。 例如：配置256M，15分钟，如果文件大小在5分钟就到了256MB，那么文件大小这个条件先触发投递任务。	5 - 256，单位：MB	必填
投递间隔时间	指定间隔多长时间，触发一次投递，和投递文件大小配合使用，哪个条件先触发，就按照哪个规则去压缩文件，然后投递到 COS。 例如：配置256MB，15分钟，如果文件大小在15分钟时仅为200MB，间隔时间这个条件先触发投递任务。	5 - 15分钟	必填

6. 单击**下一步**，进入目标桶配置。

配置项	解释说明	规则	是否必填
目标 COS 桶归属	**当前主账号** 投递 CLS 日志到当前主账号的 COS 桶。 **其他主账号** 投递 CLS 日志到其他主账号的 COS 桶，例如将 A 账号的 CLS 日志投递至 B 账号的 COS 桶，需要 B 账号在 CAM（访问管理）侧配置访问 角色，配置完成之后，由 A 账号将**角色 ARN和外部 ID** 填写到 CLS 控制台，方可进行跨账号投递。配置角色的步骤如下： 1. 新建角色。账号 B 登录 CAM 角色管理页面 。 1.1 创建访问策略，策略名称例如：**cross_shipper**，策略语法参考如下： **说明：** 示例中的授权按照最小权限的原则，resource 配置为仅可以投递至广州地域的 COS 桶：test123-123456789，请您按照实际情况进行授权。 `{ "statement": [ { "action": [ "cam:GetRole" ], "effect": "allow", "resource": [ "*" ] }, { "action": [ "cos:PutObject" ], "effect": "allow", "resource": [ "qcs::cos:ap-guangzhou:uid/123456789:test123-123456789/*" ]//投递至广州地域的COS桶:test123-123456789,请您按照实际情况进行授权。 } ], "version": "2.0" }` 1.2 新建角色，选择**腾讯云账户**角色载体，云账号类型选择**其他主账号**，然后输入 A 账号的 ID，例如100012345678，勾选**开启校验**并配置外部 ID，例如 **Hello123**。 1.3 配置角色的访问策略，选择第一步配置好的访问策略 cross_shipper（示例）。 1.4 保存该角色，例如：**uinA_writeCLS_to_COS**。 2. 为角色配置载体。在 CAM 角色列表中找到 uinA_writeCLS_to_COS（示例），单击该角色，选中下方的**角色载体 > 管理载体 > 添加产品服务 > 选中日志服务**，然后单击**更新**。 可以看到当前角色的载体是两个：一个是 A 账号，另一个是 cls.cloud.tencent.com（CLS 日志服务）。 3. A 账号登录 CLS，填入**角色 Arn 和外部 ID**。	当前主账号/其他主账号	可选

	以下两项信息需 B 账号来提供： ○ B 账号在 CAM 角色列表中找到角色 uinA_writeCLS_to_COS（示例），单击可查看该角色的 RoleArn，例如 qcs::cam::uin/100001112345:roleName/uinA_writeCLS_to_COS。 ○ 在角色载体中可看到外部 ID，例如 Hello123。 ⚠ 注意： • 填写角色 ARN、外部 ID 时，注意不要输入多余的空格，否则会导致权限校验失败。 • 跨账号投递会在 A 账号下，产生日志主题的读流量费。		
COS 存储桶	投递日志的目标存储桶，跨账号投递需要用户手动填写目标存储桶的名称。	列表选择	必填
文件命名	• 投递时间命名：默认选项，例如202208251645_000_132612782.gz 表示投递时间_日志主题分区_offset，Hive 可加载此文件。 • 随机数命名：旧版的命名方式，Hive 可不识别，Hive 不识别_开头的文件，可以在 COS 路径配置项里面添加自定义前缀，例如/%Y%M%d/%H/Yourname。	/	必填
文件压缩	不压缩\snappy\lzop\gzip。	gzip\snappy\lzop	必填
COS 路径	COS 存储桶中保存日志的路径。默认按照格式 `/年/月/日/小时/` 保存，例如/2022/7/31/14/，路径配置支持 strftime 的语法，如： • /%Y/%m/%d/生成的路径是/2022/7/31/ • /%Y%M%d/%H/生成的路径是/20220731/14/	非/开头	可选
COS 存储类型	日志在 COS 桶存储的类型：标准存储、低频存储、智能分层存储、归档存储、深度归档存储，详情请参见 存储类型概述 。 ⓘ 说明： 跨账号投递暂不支持选择 COS 存储类型。	列表选择	必填

7. 单击**下一步**，进入高级配置，您可以选择数据的格式，以 JSON/CSV/Parquet 进行投递。

投递数据格式	使用场景
按 CSV 格式投递	• 可用于腾讯云 DLC 入湖计算 • 可以利用 CSV 投递实现 投递日志原文（单行、多行、分隔符采集的日志）
按 JSON 格式投递	常见数据格式，请用户参考自己的业务场景进行选择
按 Parquet 格式投递	日志数据需为结构化数据，支持数据类型转换（非单行、多行采集），多用于 Hive

投递格式为 JSON

1. 选择要投递的日志内容（字段），其中_CONTENT_是用户的日志，_SOURCE_、_FILENAME_、_HOSTNAME_、_TIMESTAMP_、_TAG_元信息均为 CLS 的元数据字段，请您根据实际情况勾选。
2. 转义选项：

转义选项	说明
不转义	不对您的 JSON 结构和层级做修改，日志格式和采集侧保持一致。 例子： • 日志原文：{"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} • 投递到 COS：{"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} ⚠ 注意：

	当 JSON 的第一层节点中包含有数值时，投递后会自动转为 int、float。 • 日志原文：{"a":123, "b":"123", "c":"-123", "d":"123.45", "e":{"e1":123,"f1":"123"}} • 投递到 COS：{"a":123,"b":123,"c":-123,"d":123.45,"e":{"e1":123,"f1":"123"}}
转义	将 JSON 第一层节点的值转为 String，如您的节点值是 Struct，在下游入库或者计算时，需要提前转为 String，可以选这个选项。 例1： • 日志原文：{"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} • 投递到 COS:{"a":"aa","b":"{\"b1\":\"b1b1\", \"c1\":\"c1c1\"}"} 例2： • 日志原文：{"a":123, "b":"123", "c":"-123", "d":"123.45", "e":{"e1":123,"f1":"123"}} • 投递到 COS:{"a":"123","b":"123","c":"-123","d":"123.45","e":"{\"e1\":123,\"f1\":\"123\"}"}

投递格式为 CSV

选择以 CSV 格式投递日志到 COS。

> **说明：**
> 可删除不需要的 CLS 元数据字段。

配置项说明如下：

配置项	解释说明	规则	是否必填
键值名称（key）	指定写入 CSV 文件的键值（key）字段（填写的 key 必须是日志结构化后的 key 名称或保留字段，否则将视为无效 key）	仅支持字母、数字和_-./@	必填
分隔符	CSV 文件中各字段间的分隔符	列表选择	必填
转义符	若正常字段内出现了分隔符的字符，需用转义符包裹该字符，防止读取数据时被错误识别	列表选择	必填
无效字段填充	若配置的键值字段（key）不存在时，则会用无效字段进行填充	列表选择	必填
首行 Key	在 CSV 文件的首行增加字段名的描述，即将键值（key）写入 CSV 文件的首行，默认不写入	开/关	必填

投递格式为 Parquet

Parquet 文件可以被 Hive 加载，多用于大数据的计算分析，下面将为您详细介绍如何创建 Parquet 格式日志投递任务。

> **注意：**
> • Parquet 文件大多用于大数据平台，由于 Parquet 本身有一定的压缩率，加上文件压缩格式（snappy/lzop/gzip），因此，投递文件大小要配置的大一些，建议不小于200MB（投递到 COS 大约在50M）。
> • Parquet 格式不支持投递数据预览。
> • 如您需要使用 导入 COS 数据，将投递至 COS 的日志，再导入 CLS 日志主题。Parquet 格式不支持导入 CLS 日志主题，CSV 和 JSON 格式可支持。

配置项说明如下：

配置项	解释说明	是否必填
键值名称（key）	写入 Parquet 文件 Header 部分的键值（key）字段。如系统自动拉取日志中的键值不满足您的需求，您可添加字段，最多100个，字段名仅支持字母、数字和_-./@。 如您的某行日志中缺少已定义的 Key，那么该行日志对应的 Parquet 文件 Body 中不会有这个 Key。这不会影响您的 Spark、Flink 等大数据计算。	必填
数据类型	该字段在 Parquet 文件中的数据类型：String、Boolean、Int32、Int64、Float、Double	必填

解析失败赋值	数据解析失败时，给予的默认值。如某字段值（String 类型）解析失败，您可以指定其值为空字符串""，也可以指定 NULL 或您自定义的字符串。其他数据类型：布尔、整型、浮点型同理。	必填

投递日志原文

最近更新时间：2025-06-25 19:05:52

使用场景

如果您想把采集的日志原文，投递到 COS，可以参考本文进行操作。目前只有单行、多行采集的日志，以及部分分隔符采集的日志适用于本场景。

日志采集格式	是否支持原文投递
单行全文	支持，参见 单行采集的日志 。
多行全文	支持，参见 多行采集的日志 。
分隔符（CSV）格式	不一定，参见 分隔符格式，只能支持原始日志分隔符为空格、制表符、逗号、分号、竖线的情况。
JSON 格式	不支持。
完全正则	不支持。

操作步骤

单行（或多行）文本采集的日志

单行或多行采集的日志可在 CSV 格式投递 的基础上，通过参数配置达到按原文投递的效果。

1. 新建投递任务，详情可参见 新建投递任务 。
2. 投递格式选择 CSV，仅保留 `__CONTENT__` 字段，其他字段删除，分隔符选择空格，转义符选择空，无效字段填充选空，关闭“首行 Key”。如下图所示：
   配置项详细说明如下：

配置项	填写	解释说明
键值名称（key）	__CONTENT__	单行或多行全文系统会默认 __CONTENT__ 作为键名称（key），日志原文作为值（value），原文投递时只保留 __CONTENT__字段。
分隔符	空格	对于单行或多行全文而言，分隔符选择空格。
转义符	空	为防止因转义符改变原文内容，转义符选择空。
无效字段	空	无效字段选择空。
首行Key	关	原文投递无需在 CSV 文件的首行增加字段名的描述。

3. 单击确定，即可看到投递任务已开启。

分隔符格式采集的日志

> **注意：**
> 分隔符格式投递 仅支持有限的分隔符（空格、制表符、逗号、分号、竖线），所以当且仅当日志原文中的分隔符与 CSV 格式投递所支持的分隔符一致时，才可以按原文投递，否则不可以投递原文。

1. 新建投递任务，详情可参见 新建投递任务 。
2. 投递格式选择 CSV，删除无需投递的 CLS 元数据字段。
   配置项详细说明如下：

配置项	填写内容	解释说明
键值名称（key）	键（key）名称	仅保留用户字段。
分隔符	列表选择	选择原文对应的分隔符，若无相同的分隔符则不能按原文投递，目前仅支持空格、 制表符、逗号、分号、竖线。
转义符	空	为防止因转义符改变原文内容，转义符选择空即可。
无效字段	空	无效字段选择空。
首行 Key	关	原文投递无需在 CSV 文件的首行增加字段名的描述。

3. 单击**确定**，即可看到投递任务状态已开启。

投递任务管理

最近更新时间：2025-06-27 11:06:31

查看投递 COS 任务

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**投递任务**，进入投递任务管理页面。
3. 在投递任务管理页面，选择投递任务的**地域**、**主题类型**（日志主题）、**日志集**、**日志主题**，查看投递至 COS 任务的列表。
4. 单击任务名称，即可查看任务的详情和监控信息。
   - 投递流量：投递 COS 成功的流量。当用户配置的文件大小或者投递间隔时间触发投递时，会有投递日志流量值，其余时间值为0。
   - 投递行数：投递 COS 成功的行数。当用户配置的文件大小或者投递间隔时间触发投递时，会有投递日志行数值，其余时间值为0。
   - 投递延时：Latency 指的是日志数据到达 CLS 之后，等待被投递任务处理的时间，60秒内都属于正常。 该指标主要监控有无数据积压，日志是否被及时的投递到 COS。
   - 投递任务异常记录：
     - 10001为 COS 桶不存在，请检查您的 COS 桶的有效性。
     - 10002为 COS 桶没有权限，请确认您有访问该 COS 桶的权限。
     - 10003为内部错误，请重试。如果还失败，请 提交工单 。

修改投递 COS 任务

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**投递任务**，进入投递任务管理页面。
3. 在投递任务管理页面中，选择投递任务的**地域**、**主题类型**（日志主题）、**日志集**、**日志主题**、投递任务名称，找到对应的投递任务，单击右侧的**修改配置**，即可修改投递任务，投递时间范围不支持修改。
4. 如果您选择的数据包含了历史数据，任务列表中会展示历史数据投递进度、投递完成所需时长。例如说您配置开始时间为 2022年12月1日 20:55:16，结束时间为无限，提交配置的时间为2023年2月13日 20:55:48，那么历史数据为2022年12月1日 20:55:16-2023年2月13日 20:55:48的数据，实时数据部分就是从2023年2月13日 20:55:48开始的数据，在投递任务列表会展示历史数据投递进度、历史数据投递完成所需的时长。 实时数据会持续的投递到 COS。

删除投递 COS 任务

1. 登录 日志服务控制台.
2. 在左侧导航栏中，选择**投递任务**，进入投递任务管理页面。
3. 在投递任务管理界面中，选择投递任务的**地域**、**主题类型**（日志主题）、**日志集**、**日志主题**、投递任务名称，找到对应的投递任务，单击右侧的**删除**。
4. 在弹出的窗口中，单击**删除**，即可删除日志投递。

版权所有：腾讯云计算（北京）有限责任公司

配置告警

最近更新时间：2025-06-27 11:06:31

配置 CLS 投递 COS 任务的告警

1. 登录 腾讯云可观测平台 。
2. 在左侧导航栏中，选择**告警管理 > 告警配置 > 告警策略**，单击**新建策略**。
3. 配置告警策略。详细的配置说明请参见 新建告警策略 。
   - 输入策略名称。
   - **监控类型**选择**云产品监控**。
   - **策略类型**选择**日志服务/COS 投递状态监控**。
   - **告警对象**选择您投递 COS 的日志主题。
   - 支持对预置触发条件进行修改。

> **说明：**
> 投递任务状态=1时，表示任务正常运行，当任务异常（可在投递任务状态中查看）时，异常状态码从10001开始,因此使用"投递任务状态>1"对任务状态进行监控。



投递到 Ckafka
创建投递任务

最近更新时间：2025-06-27 11:06:31

操作场景

您可以将日志主题的数据投递到腾讯云 CKafka，然后用于您的实时流计算场景。如果您没有购买腾讯云 CKafka 实例，可以考虑使用日志服务（Cloud Log Service，CLS）自带的 Kafka 协议消费功能。

前提条件

- 已 开通腾讯云消息队列（CKafka）。

> **注意：**
> - 如果您的 CKafka 开启了 ACL，请注意 CKafka 的版本限制：CKafka1.X 版本，需在1.1.1及以上；CKafka2.X 版本，需在2.4.2及以上，其他版本开启 ACL 均会导致投递失败，可通过关闭 ACL 解决投递失败的问题。
> - 当前仅支持实时日志投递，不支持历史日志的投递。
> - 不支持投递至 CKafka 弹性 Topic。
> - CKafka 的支撑路由访问方式需配置为 PLAINTEXT，否则会投递失败。

- 确保当前操作账号拥有开通投递到 CKafka 的权限。详情请参见 CLS 访问策略模板。

操作步骤

1. 准备和配置 CKafka Topic。
   1.1 在日志主题同地域下，创建一个 CKafka 实例，详情请参见 创建实例。
   1.2 创建一个 Topic，详情请参见 创建 Topic。进入该 Topic 的高级配置，修改如下配置，其余参数可按需配置。
   - **cleanup.policy**：选择 delete，否则会投递失败。
   - **max.message.bytes**：设置为12MB。



2. 进入投递 CKafka 的任务管理页面，首先登录 日志服务控制台，有两种方式进入投递任务管理：
   - 入口1：在左侧导航栏中，单击**投递任务**，在页面顶部，依次选择**地域**、**主题类型**、**日志集**和**日志主题**，然后选择**投递到 CKafka** 页签。
   - 入口2：在日志主题列表，选择并进入需要投递 CKafka 的**日志主题**，在日志主题详情页面，选择页面左侧的 **投递到 CKafka** 页签。
3. 单击右侧的**编辑**，开启投递到 CKafka 开关，选择相应的 CKafka 实例以及对应的 Topic，开始投递。您可以选择以原始内容或者 JSON 投递日志。
   - 以原始内容投递到 CKafka。**配置项说明如下：**

配置项	解释说明	规则	是否必填

目标 CKafka Topic 归属	**当前主账号** 投递 CLS 日志至当前主账号的 CKafka Topic。 **其他主账号** 投递 CLS 日志至其他主账号的 CKafka，例如 A 账号在 CLS 将日志投递至 B 账号的 CKafka Topic，需要 B 账号在 CAM(访问管理)侧配置访问 角色，配置完成之后，由 A 账号将**角色 ARN 和外部 ID** 填写到 CLS 控制台，方可进行跨账号投递。 配置角色的步骤如下： 1. 新建角色。账号 B 登录 CAM 角色管理页面 。 1.1 创建访问策略，策略名称例如：**cross_shipper**，策略语法参考如下： **说明：** 示例中的授权按照最小权限的原则，resource 配置为仅可以投递至广州地域的 CKafka：ckafka-12abcde3，请您按照实际情况进行授权。 1.2 新建角色，选择**腾讯云账户**角色载体，云账号类型选择**其他主账号**，然后输入 A 账号的 ID，例如100012345678，勾选**开启校验**并配置外部 ID，例如：**Hello123**。	当前主账号/其他主账号	否

```
{
  "statement": [
      {
          "action": [
              "cam:GetRole"
          ],
          "effect": "allow",
          "resource": [
              "*"
          ]
      },
      {
          "action": [
              "ckafka:ListRoute",
              "ckafka:AddRoute",

"ckafka:DescribeInstanceAttributes",
              "ckafka:AuthorizeToken",
              "ckafka:CreateToken",

"ckafka:DescribeTopicAttributes"
          ],
          "effect": "allow",
          "resource": [
              "qcs::ckafka:ap-
guangzhou:uin/100001234567:ckafkaId/ckafka-
12abcde3"
          ]//resource 配置为仅可以投递至广州地域的 CKafka：ckafka-12abcde3，请您按照实际情况进行授权
      }
  ],
  "version": "2.0"
}
```

1.3 配置角色策略，配置角色的访问策略，选择第一步配置好的访问策略 cross_shipper(示例)。

1.4 保存该角色，例如：uinA_writeCLS_to_CKafka。

2. 为角色配置载体。在 CAM 角色列表中找到 uinA_writeCLS_to_CKafka(示例)，单击该角色，选中下方的**角色载体 > 管理载体 > 添加产品服务 > 选中日志服务**，然后单击**更新**。

   可以看到当前角色的载体是两个：一个是 A 账号，另一个是 cls.cloud.tencent.com(CLS 日志服务)。

3. A 账号登录 CLS，填入**角色 ARN 和外部 ID**。

   该两项信息需 **B 账号来提供**：

   - B 账号在 CAM 角色列表中找到角色 uinA_writeCLS_to_CKafka(示例)，单击可查看该角色的 RoleArn，例如 qcs::cam::uin/100001112345:roleName/uinA_writeCLS_to_CKafka。
   - 在角色载体中可看到外部 ID，例如 Hello123。

> **注意：**
> - 填写角色 ARN，外部 ID 时，注意不要输入多余的空格，会导致权限校验失败。
> - 跨账号投递会在 A 账号下，产生日志主题的读流量费。

CKafka 实例	与当前日志主题同地域的 CKafka Topic 作为投递目标。 跨账号投递场景下，需用户手动填写 CKafka 实例 ID 和 Topic 名称。	列表选择	必填
投递数据格式	选项**原始内容**，投递用户的原始日志。	列表选择	必填
数据压缩格式	不压缩\SNAPPY\LZ4。	列表选择	必填
投递日志预览	预览您投递的日志数据。	-	-

- 以 JSON 格式投递至 CKafka。**配置项说明如下：**

配置项	解释说明	规则	是否必填
目标 CKafka Topic 归属	**当前主账号** 投递 CLS 日志至当前主账号的 CKafka Topic。 **其他主账号** 投递 CLS 日志至其他主账号的 CKafka，例如 A 账号在 CLS 将日志投递至 B 账号的 CKafka Topic，需要 B 账号在 CAM(访问管理)侧配置访问 角色，配置完成之后，由 A 账号将**角色 ARN 和外部 ID** 填写到 CLS 控制台，方可进行跨账号投递。 配置角色的步骤如下： 1. 新建角色。账号 B 登录 CAM 角色管理页面 。 1.1 创建访问策略，策略名称例如：**cross_shipper**，策略语法参考如下： **说明：** 示例中的授权按照最小权限的原则，resource 配置为仅可以投递至广州地域的 CKafka：ckafka-12abcde3，请您按照实际情况进行授权。 `{ "statement": [`	当前主账号/其他主账号	否

```
        {
            "action": [
                "cam:GetRole"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ]
        },
        {
            "action": [
                "ckafka:ListRoute",
                "ckafka:AddRoute",

"ckafka:DescribeInstanceAttributes",
                "ckafka:AuthorizeToken",
                "ckafka:CreateToken",

"ckafka:DescribeTopicAttributes"
            ],
            "effect": "allow",
            "resource": [
                "qcs::ckafka:ap-
guangzhou:uin/100001234567:ckafkaId/ckafka-
12abcde3"
            ]//resource 配置为仅可以投递至广州地域的 CKafka：ckafka-12abcde3，请您按照实际情况进行授权
        }
    ],
    "version": "2.0"
}
```

1.2 新建角色，选择**腾讯云账户**角色载体，云账号类型选择**其他主账号**，然后输入 A 账号的 ID，例如100012345678，勾选**开启校验**并配置外部 ID，例如：**Hello123**。

1.3 配置角色策略，配置角色的访问策略，选择第一步配置好的访问策略 cross_shipper(示例)。

1.4 保存该角色，例如：uinA_writeCLS_to_CKafka。

2. 为角色配置载体。在 CAM 角色列表中找到 uinA_writeCLS_to_CKafka(示例)，单击该角色，选中下方的**角色载体 > 管理载体 > 添加产品服务 > 选中日志服务**，然后单击**更新**。

   可以看到当前角色的载体是两个：一个是 A 账号，另一个是 cls.cloud.tencent.com(CLS 日志服务)。

3. A 账号登录 CLS，填入**角色 ARN 和外部 ID**。

   该两项信息需 **B 账号来提供**：

   - B 账号在 CAM 角色列表中找到角色 uinA_writeCLS_to_CKafka(示例)，单击可查看该角色的 RoleArn，例如 qcs::cam::uin/100001112345:roleName/uinA_writeCLS_to_CKafka。
   - 在角色载体中可看到外部 ID，例如 Hello123。

注意：

- 填写角色 ARN，外部 ID 时，注意不要输入多余的空格，会导致权限校验失败。
- 跨账号投递会在 A 账号下，产生日志主题的读流量费。

CKafka 实例	与当前日志主题同地域的 CKafka Topic 作为投递目标。	列表选择	必填
投递数据格式	选项 **JSON**，以 JSON 的数据格式投递日志。	列表选择	必填
投递日志字段	请选择您需要投递的日志字段， • JSON 格式的转义/不转义的说明如下： ○ 转义，将 JSON 第一层节点的值转为 String，如果您的第一层节点的值是 Struct，在下游入库或者计算时，需要提前将该Struct 转为 String，可以选这个选项。 ○ 不转义，不对您的 JSON 结构和层级做修改，日志格式和采集侧保持一致。 **⚠ 注意：** 当 JSON 的第一层节点中包含有数值时，消费后会自动转为 **int**、**float**。 日志：{"a":123, "b":"123", "c":"-123", "d":"123.45", "e":{"e1":123,"f1":"123"}} 消费：{"a":123,"b":123,"c":-123,"d":123.45,"e":{"e1":123,"f1":"123"}} • 将__TAG__元信息平铺或者不平铺，说明如下： 示例：__TAG__元信息：{"__TAG__":{"fieldA":200, "fieldB":"text"}} ○ 平铺：{"__TAG__.fieldA":200,"__TAG__.fieldB":"text"} ○ 不平铺：{"__TAG__":{"fieldA":200, "fieldB":"text"}}	选择	必填
数据压缩格式	不压缩\SNAPPY\LZ4。	选择	
投递日志预览	预览您投递的日志数据。	-	-
高级选项	CKafka 分区生产规则：打开开关后，将依据您配置的字段值，计算 Hash 值，结果相同的，会写入 CKafka 的同一个分区。		

4. 单击**确定**，启动投递到 CKafka。

> **⚠ 注意：**
> 如需在投递至 CKafka 前对日志进行清洗加工过滤，请参见使用 数据加工 操作。

常见问题

提示没有读写 CKafka Topic 的权限，怎么办?

如果您直接使用 API 接口投递数据到 CKafka，可能会存在读写 CKafka Topic 的权限问题。因为，如果您在控制台使用该功能，系统会引导您完成相关授权；如果您直接调用 API 投递，则需要手动授权。具体的排查和解决方案请参见 投递任务角色授权 。

投递任务管理

最近更新时间：2025-06-27 11:06:31

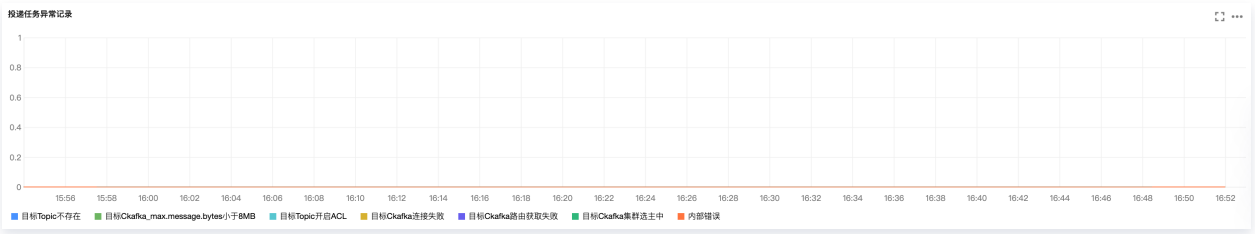
查看投递 CKafka 任务

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**投递任务**，进入投递任务管理页面。
3. 在投递任务管理页面，选择投递任务的地域、主题类型、日志集、日志主题，查看投递到 CKafka 的任务。
4. 如下图查看任务的详情和监控信息。

   您可查看投递流量、投递行数和投递延时信息：



   您可查看投递任务异常记录：



   - 投递流量：投递 CKafka 成功的日志流量（压缩后）。
   - 投递行数：投递 Ckafka 成功的日志行数。
   - 投递延时：Latency 指的是日志数据到达 CLS 之后，等待被投递任务处理的时间，5秒内都属于正常。该指标主要监控有无数据积压，日志是否被及时的投递到 CKafka。
   - 投递任务异常记录：展示任务异常的原因。例如目标 Topic 不存在、目标 CKafka 路由获取失败等。
   - 目的 Ckafka 磁盘满：是导致投递失败的原因之一。
   - 目的 Ckafka 写入限流：是导致投递失败的原因之一。

修改投递 CKafka 任务

1. 在投递任务管理页面中，选择投递任务的地域、主题类型、日志集、日志主题，选择**投递到 CKafka** 的页签，找到对应的投递任务，单击右侧的**编辑**，即可修改投递任务。
2. 您可以修改投递的目标 Topic、投递数据格式、投递日志字段、数据压缩格式等配置项。

关闭投递 Kafka 任务

1. 在投递任务管理界面中，选择投递任务的地域、主题类型、日志集、日志主题，选择投递到 CKafka 的页签，找到对应的投递任务，单击右侧的编辑。
2. 在弹出的窗口中，关闭投递到 CKafka 的开关，单击**确定**即可。

> ⚠ **注意：**

下次重启该任务，将从最新日志数据开始投递。



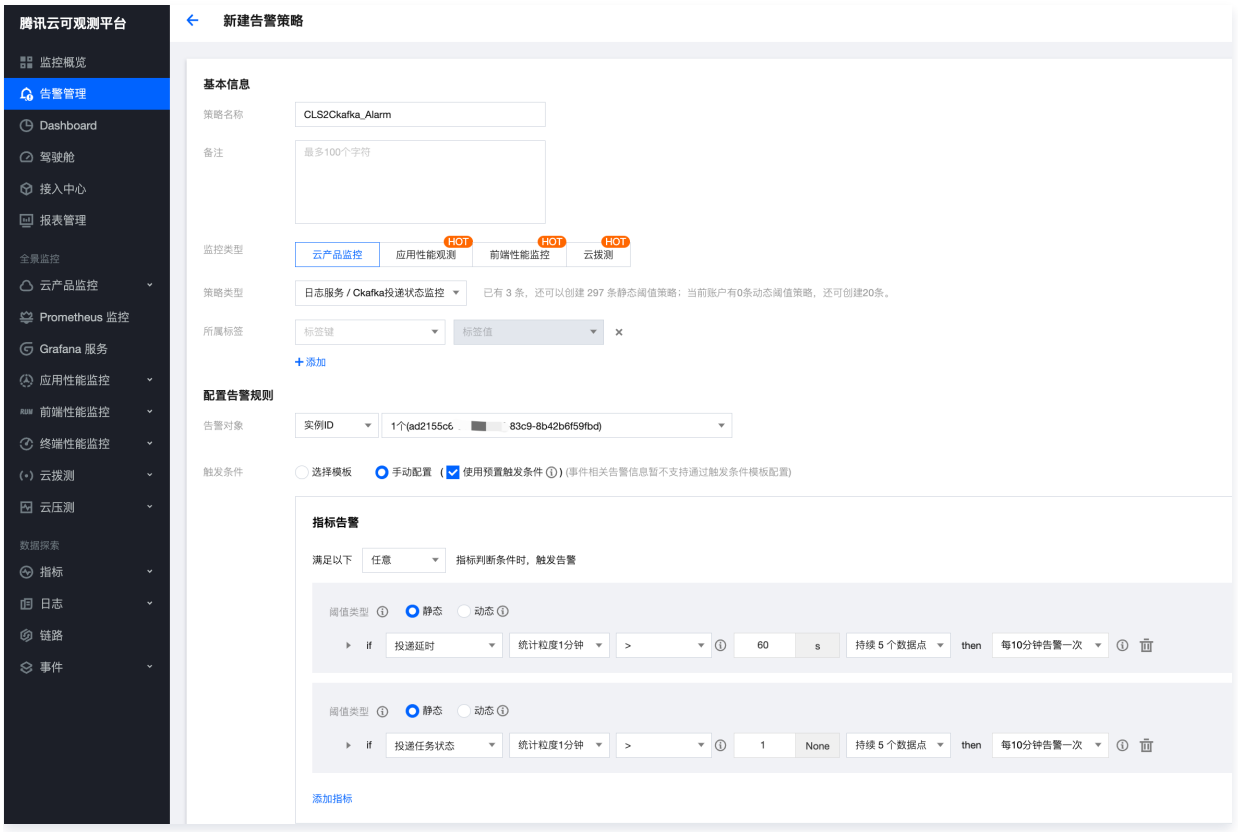
配置告警

最近更新时间：2024-10-24 20:59:22

配置 CLS 投递 Ckafka 任务的告警

1. 登录 腾讯云可观测平台；
2. 在左侧导航栏中，选择**告警管理**；
3. 输入策略名称，策略类型选择**日志服务/CKafka 投递状态监控**；
4. 告警对象选择您投递 Ckafka 的日志主题；
5. 图中展示的是预置触发条件，您可对其进行修改。

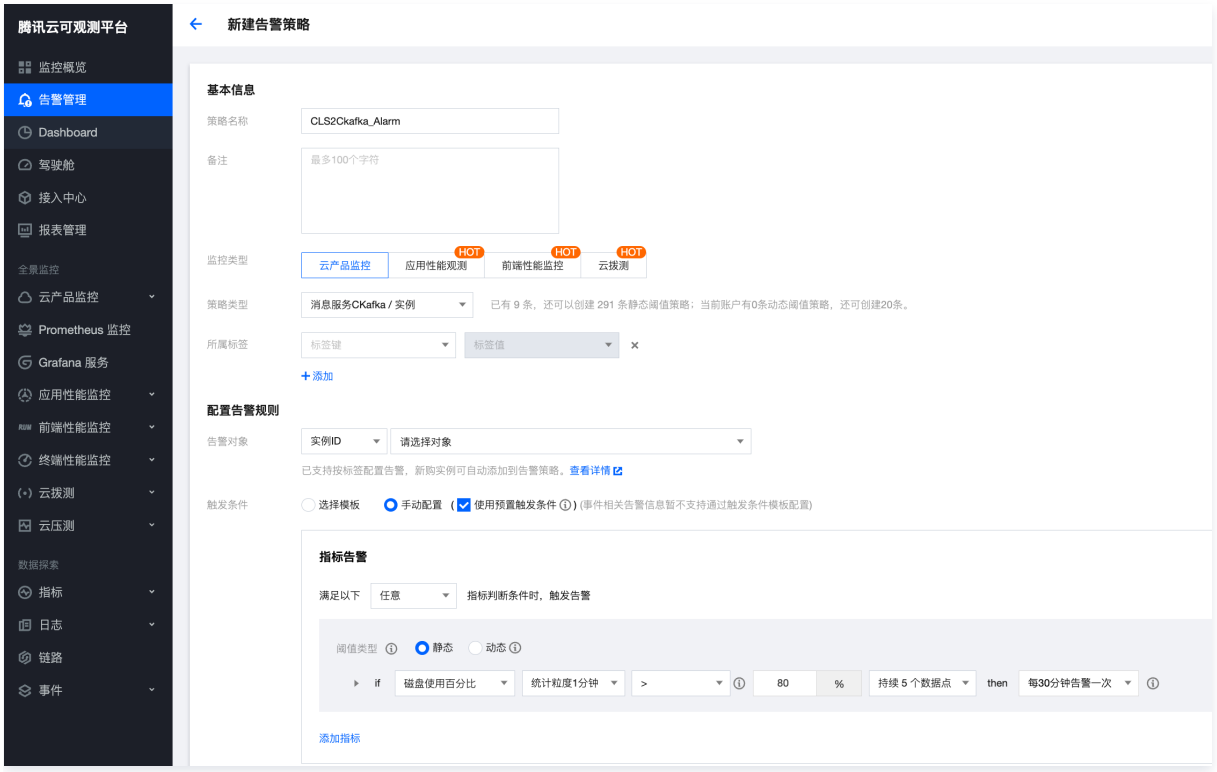
> **说明：**
> 投递任务状态=1时，表示任务正常运行，当任务异常（可在投递任务异常记录中查看）时，异常状态码从20001开始,因此使用"投递任务状态>1"对任务状态进行监控。



配置 Ckafka 实例的告警

在实际生产中，CKafka 实例的磁盘满也是导致投递任务失败的主要原因之一，因此，建议您同时配置 Ckafka 实例磁盘使用率的告警。

1. 输入策略名称，策略类型选择消息服务 Ckafka/实例；
2. 告警对象：选择您的 Ckafka 实例；
3. 触发条件：图中展示的是预置触发条件，您可对其进行修改。


腾讯云可观测平台
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
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
数据探索
指标
日志
链路
事件
新建告警策略
基本信息
策略名称
CLS2Ckafka_Alarm
备注
最多100个字符
监控类型
云产品监控
应用性能观测
前端性能监控
云拨测
HOT
策略类型
消息服务CKafka / 实例
已有 9 条，还可以创建 291 条静态阈值策略；当前账户有0条动态阈值策略，还可创建20条。
所属标签
标签键
标签值
添加
配置告警规则
告警对象
实例ID
请选择对象
已支持按标签配置告警，新购实例可自动添加到告警策略。查看详情
触发条件
选择模板
手动配置
使用预置触发条件
(事件相关告警信息暂不支持通过触发条件模板配置)
指标告警
满足以下
任意
指标判断条件时，触发告警
阈值类型
静态
动态
if
磁盘使用百分比
统计粒度1分钟
>
80
%
持续 5 个数据点
then
每30分钟告警一次
添加指标

投递到 DLC
新建投递 DLC 任务

最近更新时间：2025-04-09 19:07:43

日志采集到 CLS 后，支持将日志数据投递至腾讯云 DLC 中进行分析。

前提条件

- 开通日志服务，创建 日志集 与 日志主题，并成功采集到日志数据。
- 开通腾讯云 DLC 服务，并且在待投递日志主题的地域已创建数据库和数据表，请参见 SQL 编辑器。
- 子账号/协作者需要主账号授权，授权步骤参见 基于 CAM 管理权限，授权策略请参见 CLS 访问策略模板。
- 已授权给腾讯云 CLS 服务角色访问 DLC 的权限。大部分用户通过控制台操作时，系统会引导用户完成授权；小部分用户跨过控制台，直接调用 API，这部分客户需要手动去授权，详情请参见 投递任务角色授权。

操作步骤

1. 登录 日志服务控制台。
2. 在左侧导航栏中，选择**日志主题**。
3. 单击需要投递的日志主题 ID/名称，进入日志主题管理页面。
4. 单击**投递到 DLC** 页签，进入投递到 DLC 配置页面，单击**新建**。

基本配置项	解释说明	规则	是否必填
投递任务名称	配置投递任务的名称。	-	必填
服务日志	将投递任务运行的监控指标，写入免费的日志主题 cls_service_log 中。	-	否
投递模式	当前仅支持**批投递**。	-	否
投递文件大小	需要投递的原始日志文件的大小，和投递间隔时间配合使用，哪个条件先触发，就按照哪个规则去压缩文件，然后投递到 DLC。 例如：配置256M，15分钟，如果文件大小在5分钟就到了256MB，那么文件大小这个条件先触发投递任务。	5 - 256，单位：MB	否
投递间隔时间	指定间隔多长时间，触发一次投递，和投递文件大小配合使用，哪个条件先触发，就按照哪个规则去压缩文件，然后投递到 DLC。 例如：配置256MB，15分钟，如果文件大小在15分钟时仅为200MB，间隔时间这个条件先触发投递任务。	300 - 900，单位：s	否

5. 单击**下一步**，进入**数据表配置**。

数据表配置项	解释说明	规则	是否必填
数据目录	当前仅支持 DataLakeCatalog。	-	否
数据库	选择您的 DLC 数据库。	-	必填
数据表	选择您的 DLC 数据表。	-	必填

数据字段	• 日志字段名：将 CLS 日志中的字段，和对应的 DLC 字段做 Mapping，如下图中日志：仅支持填写 Json 中的第一层节点的 key 如 app_name，不支持 details.request_id 这种嵌套节点。
	<pre>{
	"__TIMESTAMP__":1742543131,
	"app_name": "cls",
	"category": "Filter",
	"latency": "11",
	"details": {
	"request_id": "123456dfg",</pre>
		-	必填

	<pre>"trace_id": "0610df2a4c5d9cdf"
	}
	}</pre>
	• 预览日志：点击该按钮,您可在页面右侧，查看日志样例（JSON 格式），方便您选择字段并填写至日志字段名中。
	• 数据表字段：系统从 DLC 中读取，此处不支持修改，请前往 DLC 进行修改。
	• 字段类型：DLC 字段的类型，此处不支持修改，请前往 DLC 进行修改。
	• 解析失败赋值：NULL/空/自定义值，如果用户原始字段无法解析为指定类型，会按解析失败字段尝试解析，如果解析失败字段也无法解析，会填充默认的零值。
	  ○ int/bigint：对应0
	  ○ float/double/decimal：对应0
	  ○ date：对应当天时间
	  ○ timestamp：对应当前时间戳
	• 开启映射：是否将该字段映射至 DLC 表，如果不需要，请关闭开关。		

分区字段	• 日志字段：日志的字段名称，用于 Mapping DLC 的分区字段，如果您的 DLC 表按照时间进行分区，那么建议您此处使用日志时间字段，即__TIMESTAMP__。 • 数据表字段：分区字段，系统从 DLC 中读取，此处不支持修改，请前往 DLC 进行修改。 • 字段类型：分区字段的类型，系统从 DLC 中读取，此处不支持修改，请前往 DLC 进行修改。	–	必填

6. 单击**提交**，完成新任务的配置。

日志消费
Kafka 协议消费
使用 Kafka 协议消费日志

最近更新时间：2025-06-27 15:38:52

使用 Kafka 协议消费功能，您可以将一个日志主题，当作一个 Kafka Topic 来消费。可将采集到 CLS 的日志数据，消费到下游的大数据组件或者数据仓库。比如 Flink、Flume、Logstash、Splunk 以及腾讯云 Oceanus。

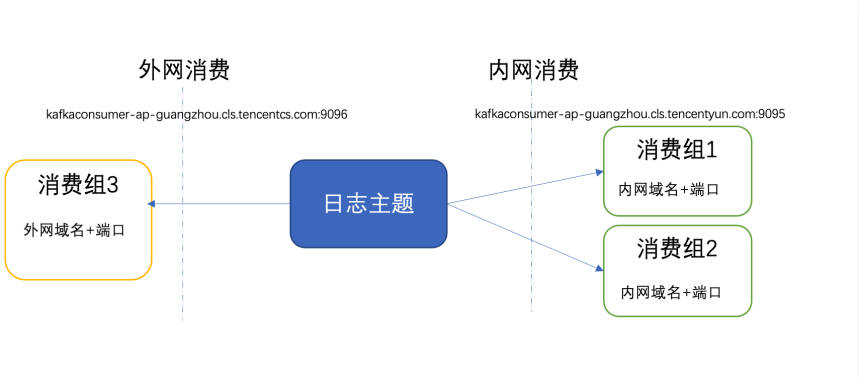
前提条件

- 已开通日志服务，创建 日志集 与 日志主题，并成功采集到日志数据。
- 确保当前操作账号拥有开通 Kafka 协议消费的权限，权限问题请参见 CLS 访问策略模板 。

> 注意：
> - 开启该功能后，默认将从最新数据开始消费。消息保留时间2小时，超过2小时还未消费的数据将不会被保存。例如您12:00打开 Kafka 协议消费，那么您将从12:00的数据开始消费，直到15:00，您可以消费13:00 – 15:00的数据，13:00之前的数据不能消费。
> - 新版本（公测中）支持消费历史数据，如需使用，请 提交申请 单独开通。

内网消费和外网消费说明

- **内网消费**：使用内网域名进行日志消费，流量费用为0.18元/GB。例如您的原始日志为100GB，消费时选择 Snappy 压缩，那么计量约为50GB，内网读流量费用为50GB * 0.18元，即9元。一般来说，如果您的消费端和日志主题在同一个 VPC 或者同一个地域，就可以使用内网消费。
- **外网消费**：使用外网域名进行日志消费，流量费用为0.8元/GB。例如您的原始日志为100GB，消费时选择 Snappy 压缩，那么计量约为50GB，外网读流量费用为50GB * 0.8元，即40元。一般来说，如果您的消费端和日志主题不在同一个 VPC，也不在同一个地域，需要使用外网消费。外网消费可以在配置项中关闭。



操作步骤

1. 登录日志服务控制台，选择左侧导航栏中的 日志主题。
2. 在**日志主题**页面，单击需要使用 Kafka 协议消费的日志主题 ID/名称，进入日志主题管理页面。
3. 在日志主题管理页面中，单击 **Kafka 协议消费**页签。
4. 单击右侧的**编辑**，将**当前状态**的开关按钮设置为打开状态后，编辑如下配置项，然后单击**确定**。

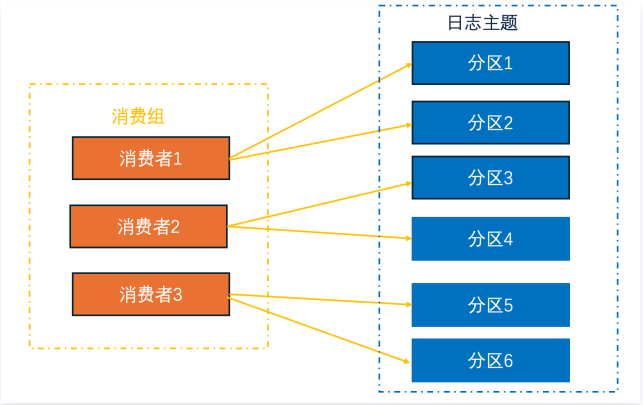
配置项	解释说明	规则
消费数据格式	• **JSON**，以 JSON 的数据格式消费日志。 • **原始内容**，以原文的格式消费日志。	选择
消费日志字段	请选择您需要消费的日志字段。 • JSON 格式的转义/不转义的说明如下: ○ 转义，将 JSON 第一层节点的值转为 String，如果您的第一层节点的值是 Struct，在下游入库或者计算时，需要提前将该Struct 转为 String，可以选这个选项。	选择

	○ 不转义，不对您的 JSON 结构和层级做修改，日志格式和采集侧保持一致。 **注意：** 当 JSON 的第一层节点中包含有数值时，消费后会自动转为 **int、float**。 日志：{"a":123, "b":"123", "c":"-123", "d":"123.45", "e":{"e1":123,"f1":"123"}} 消费：{"a":123,"b":123,"c":-123,"d":123.45,"e":{"e1":123,"f1":"123"}} • 将__TAG__元信息平铺或者不平铺，说明如下。 示例：__TAG__元信息：{"__TAG__":{"fieldA":200, "fieldB":"text"}} ○ 平铺：{"__TAG__.fieldA":200,"__TAG__.fieldB":"text"} ○ 不平铺：{"__TAG__":{"fieldA":200, "fieldB":"text"}}	
数据压缩格式	支持 SNAPPY\ LZ4\不压缩三种类型。	选择
外网消费	关闭后，您不可以从外网消费日志，仅可内网消费。	开关
消费日志预览	预览您消费的日志数据。	–
服务日志	消费的相关日志，用于您的消费监控图表，该数据由 CLS 免费提供。	开关

5. 控制台给出 Topic、Host+Port 的信息。您可以复制该信息，构造您的消费者（KafkaConsumer）。您也可以使用**自动生成消费者**小工具，生成一个可运行的消费客户端，如有其他业务逻辑请修改代码。



日志主题的分区较多时，建议您启动 多个消费者 进行消费，以免数据积压。您可在日志主题详情中查看分区个数。



消费示意图

消费者参数说明

参数	说明

用户认证方式	目前仅支持 SASL_PLAINTEXT。
hosts	• 内网消费：kafkaconsumer-${region}.cls.tencentyun.com:9095。 • 外网消费：kafkaconsumer-${region}.cls.tencentcs.com:9096，详细请参见 可用域名 - Kafka 消费日志 。
topic	消费主题 ID，请在 Kafka 协议消费的控制台复制。例如 XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX。
username	配置为 ${LogSetID}，即日志集 ID。例如：0f8e4b82-8adb-47b1-XXXX-XXXXXXXXXX ，请在 Kafka 协议消费的控制台复制。
password	配置为 `${SecretId}#${SecretKey}` 。例如：XXXXXXXXXXXXXX#YYYYYYYY，请登录 腾讯云访问管理 ，在左侧导航栏中单击**访问密钥**，API 密钥或者项目密钥均可使用。 如您有 子账号 需要使用该功能，建议使用 子账号密钥 。您为子账号授权时，建议将访问策略中的 action、resource 都配置为最小范围，详情可参见 Kafka 协议消费授权 。

消费者 Demo

- 消费 Demo-多语言SDK
- 消费 Demo-第三方软件
- 消费 Demo-流处理

任务监控和告警

最近更新时间：2025-01-14 15:11:12

操作步骤

1. 登录日志服务控制台，选择左侧导航栏中的 日志主题。
2. 在**日志主题**页面，单击需要使用 Kafka 协议消费的日志主题 ID/名称，进入日志主题管理页面。
3. 在日志主题管理页面中，单击 **Kafka 协议消费**页签。
4. 下拉页面至监控，监控分为主题、消费组两个维度。

下面分别从两个维度详细说明监控指标和告警的配置。

新版本

说明：
新版本（公测中）支持消费历史数据，如需使用，请 提交申请 单独开通。

消费主题维度

监控指标

- 消费日志（条）：消费日志的条数。
- 消费流量：该日志主题下的消费流量。



说明：
- 如果您同时开启了两个消费组，那么消费流量为二者之和，消费条数同理。
- 您可在图表上方选择监控数据的时间区间。

配置告警

说明：
使用告警，需要您打开**服务日志**的开关，并拥有 cls_service_log 的读写权限。

您可以单击图表右上角的告警图标，配置消费日志条数的告警，详情请参见 配置告警策略 。

消费组维度

监控指标

- **消费组颗粒度**：选择您要监控的消费组。
  消费延迟（秒）：消费组消费的整体延迟时间。
- **分区颗粒度**：选择消费组之后，进一步选择分区。
  消费延迟（秒）：单个分区消费的延迟时间。



说明：

- **最新日志时间戳**：到达 CLS 的最新日志对应的时间戳。
- **最近消费日志时间戳**：客户端消费到的最新日志对应的时间戳。

配置告警

您可以单击图表右上角的告警图标，配置消费延迟的告警，详情请参见 配置告警策略 。

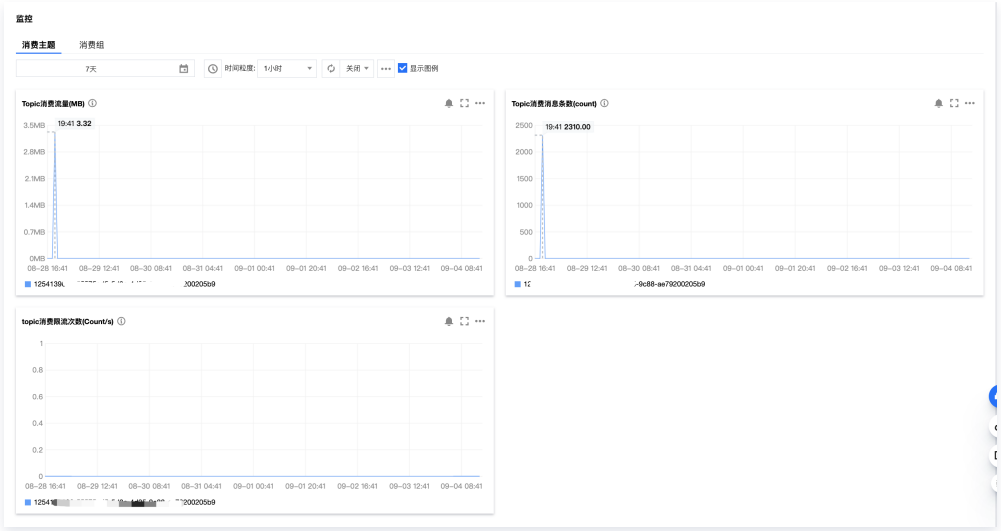
旧版本

消费主题维度

监控指标

- Topic 消费流量：消费日志的流量。
- Topic 消费消息条数：消费日志的条数。
- topic 消费限流次数：单分区流量超过5M/S时，可能触发限流。

说明：
如果您同时开启了两个消费组，那么消费流量为二者之和，消费条数同理。



配置告警

单击 **topic 消费限流次数**监控图中的 按钮，可以对 Topic 消费限流次数配置告警，详细配置信息请参见 新建告警策略 。

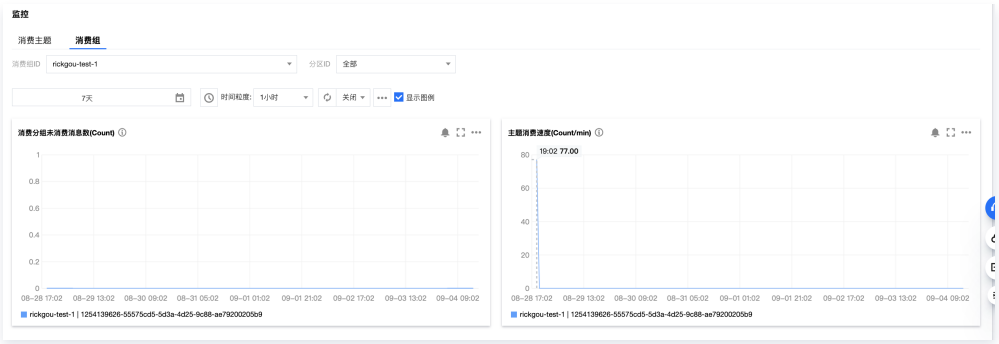
- 输入告警策略名称
- 选择日志主题 ID
- 配置告警规则
- 配置告警通知

消费组维度

监控指标

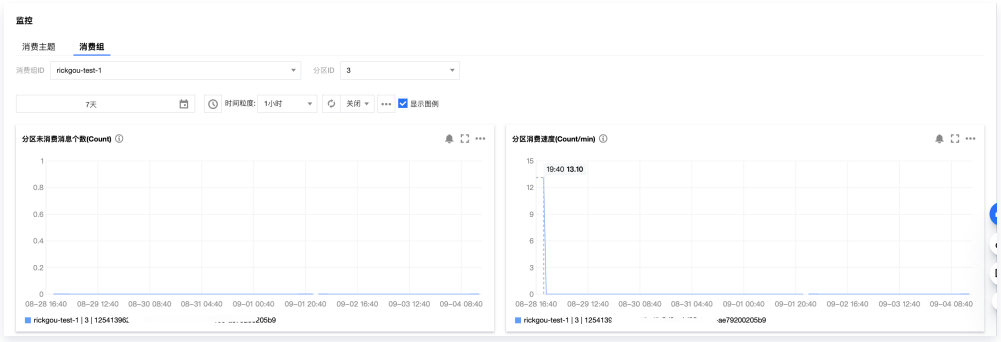
消费组颗粒度：

- 消费分组未消费消息数：指定消费组未消费的日志条数。
- 主题消费速度：指定消费组对日志主题的消费速度，单位为条/分钟。



分区颗粒度：选择消费组之后，进一步选择分区。

- 分区未消费消息个数：指定分区内未消费的日志条数。
- 分区消费速度：指定分区内日志消费速度，单位为条/分钟。



配置告警

- 单击**消费分组未消费消息数**监控图中的告警图标，跳转至可观测平台页面，您可对未消费消息数的阈值进行配置并告警，详细配置信息请参见 新建告警策略。
- 单击**分区未消费消息个数**监控图中的告警图标，跳转至可观测平台页面，您可对未消费消息数的阈值进行配置并告警，详细配置信息请参见 新建告警策略。

消费 Demo-多语言 SDK

最近更新时间：2025-06-27 11:06:31

本文介绍使用 Python、Java 消费 CLS 日志。

Python SDK

说明：
- 推荐您使用 Python 版本：3.9及以上。
- Python kafka 客户端：kafka-python、kafka-python-ng、confluent-kafka-python。
- 如果您配置了数据压缩格式：格式 SNAPPY，请确认安装了 python-snappy 包；格式 LZ4，确认安装了 lz4 包。

单个消费者

```
import uuid
from kafka import KafkaConsumer,TopicPartition,OffsetAndMetadata
consumer = KafkaConsumer(
# cls kafka 协议消费控制台给出的的主题名称，例如XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX，可在控制台复制
'您的消费主题',
group_id = '您的消费组名称',
auto_offset_reset='earliest',
# 服务地址+端口，外网端口9096，内网端口9095，例子是内网消费，请根据您的实际情况填写
bootstrap_servers = ['kafkaconsumer-${region}.cls.tencentyun.com:9095'],
security_protocol = "SASL_PLAINTEXT",
sasl_mechanism = 'PLAIN',
# 用户名是日志集合ID，例如ca5cXXXXdd2e-4ac0af12-92d4b677d2c6
sasl_plain_username = "${logsetID}",
# 密码是用户的 SecretId#SecretKey 组合的字符串，例如 AKID********************************#XXXXuXtymIXT0Lac注意不要丢失#。建议使用子账号密钥，主账号为子账号授权时，遵循最小权限原则，即子账号的访问策略中的 action、resource 都配置为最小范围，可以满足操作即可。
sasl_plain_password = "${SecretId}#${SecretKey}",
api_version = (0,10,1)
)
print('begin')
for message in consumer:
    print('begins')
    print ("Topic:[%s] Partition:[%d] Offset:[%d] Value:[%s]" % (message.topic, message.partition,
message.offset, message.value))
    print('end')
```

多个消费者

```
from kafka import KafkaConsumer
import threading

TOPIC_NAME = '您的消费主题'
GROUP_ID = '您的消费组名称'
# 服务地址+端口，外网端口9096，内网端口9095，例子是内网消费，请根据您的实际情况填写
BOOTSTRAP_SERVERS = ''kafkaconsumer-${region}.cls.tencentyun.com:9095''

def consume_messages(thread_id):
    # 创建 Kafka 消费者实例
    consumer = KafkaConsumer(
        TOPIC_NAME,
        group_id=GROUP_ID,
        bootstrap_servers=BOOTSTRAP_SERVERS,
        value_deserializer=lambda m: m.decode('utf-8'),
```

```
        auto_offset_reset='earliest',
        security_protocol = "SASL_PLAINTEXT",
        sasl_mechanism = 'PLAIN',
        sasl_plain_username = "${logsetID}"",
        sasl_plain_password = "${SecretId}#${SecretKey}",
        api_version = (2, 5, 1)
    )

    try:
        for message in consumer:
            print(f"Thread {thread_id}: partition = {message.partition}, offset = {message.offset}, value
= {message.value}")
    except KeyboardInterrupt:
        pass
    finally:
        # 关闭消费者
        consumer.close()

if __name__ == "__main__":
    # 启动3个消费者线程，这是个例子，请您根据实际情况配置
    num_consumers = 3
    threads = []
    for i in range(num_consumers):
        thread = threading.Thread(target=consume_messages, args=(i,))
        threads.append(thread)
        thread.start()

    # 等待所有线程结束
    for thread in threads:
        thread.join()
```

Java SDK

> **注意：**
> 下面的例子中的 Java 代码，jaas.config 的配置，`${SecretId}#${SecretKey}` 后有(;分号)，不要漏填，否则会报错。

```
<dependency>
<groupId>org.apache.kafka</groupId>
<artifactId>kafka-clients</artifactId>
<version>2.5.0</version>
</dependency>
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;

import java.time.Duration;
import java.util.Collections;
import java.util.Properties;

public class KafkaConsumerGroupTest {

    public static void consume() {
        Properties props = new Properties();
        String logset_id = "${logsetID}";
        // CLS 控制台 kafka 协议消费页面展示的主题名称
        String topic_id = "您的消费主题";

        String accessKeyID = System.getenv("${SecretId}");
        String accessKeySecret = System.getenv("${SecretKey}");
```

```
        String groupId = "您的消费组名称";

       // 服务地址+端口，外网端口9096，内网端口9095，例子是内网消费，请根据您的实际情况填写
        String hosts = "kafkaconsumer-${region}.cls.tencentyun.com:9095";
        props.put("bootstrap.servers", hosts);
        props.put("security.protocol", "SASL_PLAINTEXT");
        props.put("sasl.mechanism", "PLAIN");
        props.put("sasl.jaas.config",
       "org.apache.kafka.common.security.plain.PlainLoginModule required username=\"" +
        logset_id + "\" password=\"" + accessKeyID + "#" + accessKeySecret + "\";");

        // Kafka 消费者配置
        props.put("group.id", groupId);
        props.put("enable.auto.commit", "true");
        props.put("auto.commit.interval.ms", "5000");
        props.put("session.timeout.ms", "10000");
        props.put("auto.offset.reset", "earliest");
        props.put("max.poll.interval.ms", "120000");
        props.put("heartbeat.interval.ms", "3000");
        props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

        // 创建 Kafka 消费者实例
        KafkaConsumer<String,String> consumer =  new KafkaConsumer<String,String>(props);
        consumer.subscribe(Collections.singletonList(topic_id));
        while(true) {
            ConsumerRecords<String, String> records = consumer.poll(Duration.ofSeconds(10000));
            for (ConsumerRecord<String, String> record : records) {
                System.out.println("Received message: (" + record.key() + ", " + record.value() + ") at
offset " + record.offset());
            }
        }
    }
    public static void main(String[] args){
        consume();
    }
}
```

消费 Demo-第三方软件

最近更新时间：2025-06-27 15:38:52

本文介绍使用 Filebeat、Logstash、Flume 消费日志，以及通过 HTTP 方式将您消费的日志发送至 SIEM（Splunk、Devo）。

Filebeat 消费 CLS 日志

> **说明：**
> 推荐使用版本7.5.0及以上。

```
filebeat.inputs:
- type: kafka
  hosts:
    - kafkaconsumer-${region}.cls.tencentyun.com:9095
  topics: "您的消费主题"
  group_id: "您的消费组名称"
  username: "${logsetID}"
  password: "${SecretId}#${SecretKey}"
  sasl.mechanism: "PLAIN"
  processors:
    - decode_json_fields:
        fields: ["message"]
        target: ""
        overwrite_keys: true
output.file:
  path: /tmp
  filename: filebeat_data.log
  rotate_every_kb: 102400
  number_of_files: 7
```

Logstash 消费 CLS 日志

> **说明：**
> 推荐使用 Logstash 8.0及以上版本。

```
input {
    kafka {
        # cls kafka 协议消费控制台给出的的主题名称，例如 XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX，可在控制台复制
        topics => "您的消费主题"
        # 服务地址+端口，外网端口9096，内网端口9095，例子是内网消费，请根据您的实际情况填写
        bootstrap_servers => "kafkaconsumer-${region}.cls.tencentyun.com:9095"
        group_id => "您的消费组名称"
        security_protocol => "SASL_PLAINTEXT"
        sasl_mechanism => "PLAIN"
        # 用户名是日志集合 ID，例如 ca5cXXXXdd2e-4ac0af12-92d4b677d2c6
        # 密码是用户的 SecretId#SecretKey 组合的字符串，例如 AKID********************************#XXXXuXtymIXT0Lac，注意不要丢失#。建议使用子账号密钥，主账号为子账号授权时，遵循最小权限原则，即子账号的访问策略中的 action、resource 都配置为最小范围，可以满足操作即可。
        sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginModule required username='${logsetID}' password='${SecretId}#${SecretKey}';"
    }

}
```

```
output {
  stdout { codec => json }
}
```

Flume 消费 CLS 日志

若您需将日志数据消费至自建的 HDFS、Kafka 集群，则可以通过 Flume 组件来中转，具体操作参见如下示例。

> **说明：**
> 推荐使用 Flume1.9.0及以上。

开启日志的 Kafka 消费协议

参见 操作步骤 开启日志的 Kafka 消费协议，并获取消费的服务域名和 Topic。

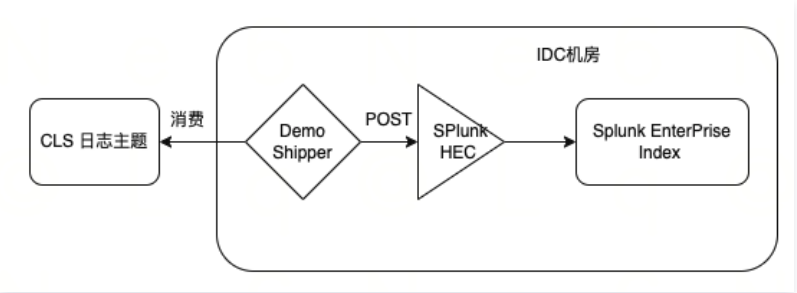
Flume 配置

```
a1.sources = source_kafka
a1.sinks = sink_local
a1.channels = channel1

# 配置Source
a1.sources.source_kafka.type = org.apache.flume.source.kafka.KafkaSource
a1.sources.source_kafka.batchSize = 10
a1.sources.source_kafka.batchDurationMillis = 200000
# 服务地址+端口，外网端口9096，内网端口9095，例子是内网消费，请根据您的实际情况填写
a1.sources.source_kafka.kafka.bootstrap.servers = kafkaconsumer-${region}.cls.tencentyun.com:9095
# cls kafka 协议消费控制台给出的的主题名称，例如 XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX，可在控制台复制
a1.sources.source_kafka.kafka.topics = 您的消费主题
# 请替换为您的消费组名称
a1.sources.source_kafka.kafka.consumer.group.id = 您的消费组名称
a1.sources.source_kafka.kafka.consumer.auto.offset.reset = earliest
a1.sources.source_kafka.kafka.consumer.security.protocol = SASL_PLAINTEXT
a1.sources.source_kafka.kafka.consumer.sasl.mechanism = PLAIN
# 用户名是日志集合 ID，例如 ca5cXXXXdd2e-4ac0af12-92d4b677d2c6
# 密码是用户的 SecretId#SecretKey 组合的字符串，例如 AKID********************************#XXXXuXtymIXT0Lac注意不要丢失#。建议使用子账号密钥，主账号为子账号授权时，遵循最小权限原则，即子账号的访问策略中的 action、resource 都配置为最小范围，可以满足操作即可，注意 jaas.config 最后有;分号，不填写会报错。
a1.sources.source_kafka.kafka.consumer.sasl.jaas.config =
org.apache.kafka.common.security.plain.PlainLoginModule required username="${logsetID}"
password="${SecretId}#${SecretKey}";

# 配置 sink
a1.sinks.sink_local.type = logger

a1.channels.channel1.type = memory
a1.channels.channel1.capacity = 1000
a1.channels.channel1.transactionCapacity = 100

# 将 source 和 sink 绑定到 channel
a1.sources.source_kafka.channels = channel1
a1.sinks.sink_local.channel = channel1
```

将日志消费至 SIEM（Splunk）

SIEM：安全信息与事件管理系统（Security Information and Event Management），例如 Splunk。

版权所有：腾讯云计算（北京）有限责任公司



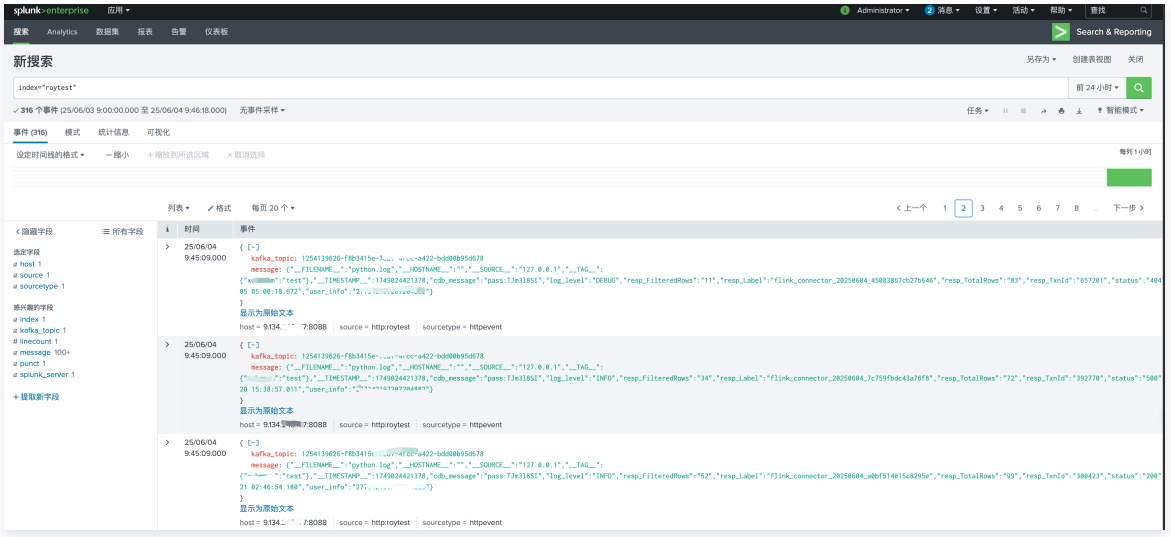
1. 请在您的 Splunk 配置 HEC （HTTP事件收集器），您在 Demo Shipper 代码中需要使用：
   - Splunk HEC endpoint，例如 http:#192.168.1.1:8088/services/collector。
   - HEC token，例如 1329e5cc-2c0d-XXXX-XXXX-332c753f54bd。
2. 部署 Demo shipper。该代码的作用：从 CLS 日志消费日志的同时，将日志通过 HTTP 的方式发送到 Splunk HEC。

```
import json
import time
import threading
import queue
from kafka import KafkaConsumer
import requests

# 日志消费参数配置
cls_consumer_config = {
    'bootstrap_servers': ['kafkaconsumer-ap-guangzhou.cls.tencentcs.com:9096'],
    'topic_name': '1254139626-XXXXXXXXX-XXXX-XXXX-bdd00b95d678', #消费主题 ID，请在 Kafka 协议消费的控制台复
制。
    'group_id': 'xhm_consumer_group',
    'auto_offset_reset': 'earliest',
    'security_protocol': 'SASL_PLAINTEXT',
    'sasl_mechanism': 'PLAIN',
    'sasl_plain_username': "e60e9e91-XXXX-XXXX-XXXXX-f19fb6f4e626",#配置为 ${LogSetID}，即日志集 ID。
    'sasl_plain_password': "AKIDnTG#SfehOq3",#配置为 ${SecretId}#${SecretKey},
    'api_version': (2, 5, 1)
}

# Splunk HEC 配置
SPLUNK_CONFIG = {
    'url': 'http:#192.168.1.1:8088/services/collector',#splunk HEC endpoint,请在Splunk侧获取
    'token': 'Splunk 1329e5cc-2c0d-XXXX-XXXX-332c753f54bd', #HEC token,请在Splunk侧获取
    'max_retries': 3,  # 失败重试次数
    'retry_delay': 1  # 重试延迟(秒)
}

# 创建线程安全的队列
log_queue = queue.Queue(maxsize=1000)  # 根据内存调整队列大小


def splunk_sender():
    """Splunk日志发送线程"""
    session = requests.Session()
    session.headers.update({
        'Authorization': SPLUNK_CONFIG['token'],
        'Content-Type': 'application/json'
    })

    while True:
        try:
            # 从队列获取日志
```

```
            log_data = log_queue.get()
            if log_data is None:  # 终止信号
                break

            # 构造Splunk事件
            event = {
                'event': {
                    'message': log_data.decode('utf-8'),
                    'kafka_topic': cls_consumer_config['topic_name']
                }
            }

            # 发送日志到Splunk（带重试机制）
            for attempt in range(SPLUNK_CONFIG['max_retries'] + 1):
                try:
                    response = session.post(
                        SPLUNK_CONFIG['url'],
                        data=json.dumps(event),
                        verify=False,  # 跳过SSL验证（-k参数）
                        timeout=5
                    )
                    response.raise_for_status()
                    print(f"Successfully sent to Splunk, status: {response.status_code}")
                    break  #成功则跳出重试循环
                except Exception as e:
                    if attempt < SPLUNK_CONFIG['max_retries']:
                        print(f"Attempt {attempt + 1} failed: {str(e)}, retrying...")
                        time.sleep(SPLUNK_CONFIG['retry_delay'])
                    else:
                        print(f"Failed to send to Splunk after {SPLUNK_CONFIG['max_retries']} attempts:
{str(e)}")
                        # 可以添加死信队列处理或本地存储逻辑

        except Exception as e:
            print(f"Error in splunk_sender: {str(e)}")
        finally:
            log_queue.task_done()


def consume_messages():
    """Kafka消费主线程"""
    consumer = KafkaConsumer(
        cls_consumer_config['topic_name'],
        bootstrap_servers=cls_consumer_config['bootstrap_servers'],
        group_id=cls_consumer_config['group_id'],
        auto_offset_reset=cls_consumer_config['auto_offset_reset'],
        security_protocol=cls_consumer_config['security_protocol'],
        sasl_mechanism=cls_consumer_config['sasl_mechanism'],
        sasl_plain_username=cls_consumer_config['sasl_plain_username'],
        sasl_plain_password=cls_consumer_config['sasl_plain_password'],
        api_version=cls_consumer_config['api_version']
    )

    print('Start consuming messages...')
    try:
        for message in consumer:
            print(
                f"Received message from Topic: {message.topic}, Partition: {message.partition}, Offset:
{message.offset}")
            # 将消息放入队列（阻塞直到有空位）
            log_queue.put(message.value)
```

```
    except Exception as e:
        print(f"Kafka consumer error: {str(e)}")
    finally:
        consumer.close()


if __name__ == "__main__":
    # 启动Splunk发送线程
    sender_thread = threading.Thread(target=splunk_sender, daemon=True)
    sender_thread.start()

    try:
        consume_messages()
    except KeyboardInterrupt:
        print("Shutting down...")
    finally:
        # 等待队列处理完毕
        log_queue.join()
        # 发送终止信号
        log_queue.put(None)
        sender_thread.join(timeout=10)
        print("All done.")
```

在 SIEM（Splunk）中查看日志：



将日志消费至 SIEM（ Devo）

- 通过 HTTP 方式：类似于Splunk、SIEM(Devo) 也提供了 HTTP endpoint，您可将消费出的日志数据通过 HTTP 协议发送至Devo，请参考 Devo 的文档 Sending data to the HTTP API endpoint 。
- 通过 Logstash，详情参考 Sending data from Logstash to Devo 。示例如下：

```
  input {
  kafka {
        # cls kafka 协议消费控制台给出的的主题名称，例如 XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX，可在控制台复制
        topics => "您的消费主题"
        # 服务地址+端口，外网端口9096，内网端口9095，例子是内网消费，请根据您的实际情况填写
        bootstrap_servers => "kafkaconsumer-${region}.cls.tencentyun.com:9095"
        group_id => "您的消费组名称"
        security_protocol => "SASL_PLAINTEXT"
        sasl_mechanism => "PLAIN"
        # 用户名是日志集合 ID，例如 ca5cXXXXdd2e-4ac0af12-92d4b677d2c6
```

```
        # 密码是用户的 SecretId#SecretKey 组合的字符串，例如
AKID********************************#XXXXuXtymIXT0Lac，注意不要丢失#。建议使用子账号密钥，主账号为子账号授权时，
遵循最小权限原则，即子账号的访问策略中的 action、resource 都配置为最小范围，可以满足操作即可。
        sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginModule required
username='${logsetID}' password='${SecretId}#${SecretKey}';"
    }
}
output {
    syslog {
        facility => "local7"
        host => "ecollector-eu.devo.io"
        port => "443"
        severity => "informational"
        appname => "my.app.oracle.test"
        protocol => "ssl-tcp"
        ssl_cert => "/Users/Bob/logstash/ca/domain.crt"
        ssl_key => "/Users/Bob/logstash/ca/domain.key"
        ssl_cacert => "/Users/Bob/logstash/ca/chain.crt"
        codec => line {
            format => "%{name},%{id}"
            }
        }
}
```

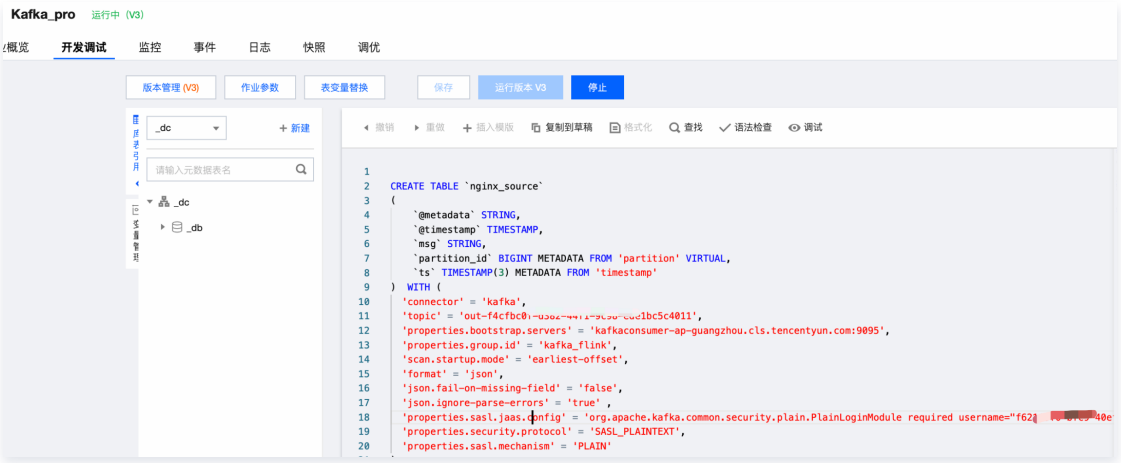
消费 Demo-流处理

最近更新时间：2025-06-27 15:38:52

本文介绍使用流处理计算框架 Flink、腾讯云 Oceanus 来消费日志。

腾讯云 Oceanus 消费 CLS 日志

1. 在 Oceanus 控制台 新建 SQL 作业 。如下图所示：



2. 编写 SQL 语句。

```
CREATE TABLE `nginx_source`
(   # 日志中字段
    `@metadata` STRING,
    `@timestamp` TIMESTAMP,
    `agent` STRING,
    `ecs` STRING,
    `host` STRING,
    `input` STRING,
    `log` STRING,
    `message` STRING,
    `partition_id` BIGINT METADATA FROM 'partition' VIRTUAL,    -- kafka 分区
    `ts` TIMESTAMP(3) METADATA FROM 'timestamp'
)  WITH (
  'connector' = 'kafka',
  # cls kafka 协议消费控制台给出的的主题名称，例如XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX，可在控制台复制
  'topic' = '您的消费主题',
  # 服务地址+端口，外网端口9096，内网端口9095，例子是内网消费，请根据您的实际情况填写
  'properties.bootstrap.servers' = 'kafkaconsumer-${region}.cls.tencentyun.com:9095',
    # 请替换为您的消费组名称
  'properties.group.id' = '您的消费组名称',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json',
  'json.fail-on-missing-field' = 'false',
  'json.ignore-parse-errors' = 'true' ,
  # 用户名是日志集合ID，例如 ca5cXXXXdd2e-4ac0af12-92d4b677d2c6
  # 密码是用户的 SecretId#SecretKey 组合的字符串，例如
AKID********************************#XXXXuXtymIXT0Lac，注意不要丢失#。建议使用子账号密钥，主账号为子账号授权时，
遵循最小权限原则，即子账号的访问策略中的 action、resource 都配置为最小范围，可以满足操作即可，注意 jaas.config 最后有;
分号，不填写会报错。
  'properties.sasl.jaas.config' = 'org.apache.kafka.common.security.plain.PlainLoginModule required
username="${logsetID}" password="${SecretId}#${SecretKey}";',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.sasl.mechanism' = 'PLAIN'
);
```

Flink 消费 CLS 日志

开启日志的 Kafka 消费协议

参见 操作步骤 开启日志的 Kafka 消费协议，并获取消费的服务域名和 Topic。

确认 flink-connector-kafka 依赖

确保 flink lib 中有 flink-connector-kafka 后，直接在 sql 中注册 Kafka 表即可使用。依赖如下：

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-kafka</artifactId>
  <version>1.14.4</version>
</dependency>
```

注册 flink 表

```
CREATE TABLE `nginx_source`
(
    #日志中的字段
    `@metadata` STRING,
    `@timestamp` TIMESTAMP,
    `agent` STRING,
    `ecs` STRING,
    `host` STRING,
    `input` STRING,
    `log` STRING,
    `message` STRING,
    # kafka 分区
    `partition_id` BIGINT METADATA FROM 'partition' VIRTUAL,
    `ts` TIMESTAMP(3) METADATA FROM 'timestamp'
)  WITH (
  'connector' = 'kafka',
  # cls kafka 协议消费控制台给出的的主题名称，例如XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX，可在控制台复制
  'topic' = '您的消费主题',
  # 服务地址+端口，外网端口9096，内网端口9095，例子是内网消费，请根据您的实际情况填写
  'properties.bootstrap.servers' = 'kafkaconsumer-${region}.cls.tencentyun.com:9095',
   # 请替换为您的消费组名称
  'properties.group.id' = '您的消费组名称',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json',
  'json.fail-on-missing-field' = 'false',
  'json.ignore-parse-errors' = 'true' ,
  # 用户名是日志集合ID，例如 ca5cXXXXdd2e-4ac0af12-92d4b677d2c6
  # 密码是用户的 SecretId#SecretKey 组合的字符串，例如 AKID********************************#XXXXuXtymIXT0Lac注意不要丢失#。建议使用子账号密钥，主账号为子账号授权时，遵循最小权限原则，即子账号的访问策略中的 action、resource 都配置为最小范围，可以满足操作即可，注意 jaas.config 最后有;分号，不填写会报错。
  'properties.sasl.jaas.config' = 'org.apache.kafka.common.security.plain.PlainLoginModule required username="${logsetID}" password="${SecretId}#${SecretKey}";',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.sasl.mechanism' = 'PLAIN'
);
```

> **注意：**
> flink 版本 sasl 认证配置对应包：
> - 1.16版本以下：org.apache.kafka.common.security.plain.PlainLoginModule。

- 1.16版本及以上：org.apache.flink.kafka.shaded.org.apache.kafka.common.security.plain.PlainLoginModule。

查询使用

执行成功后，即可查询使用。

```
select count(*) , host from nginx_source group by host;
```

自定义消费

最近更新时间：2025-06-27 11:06:31

前提条件

1. 开通日志服务，创建 日志集 与 日志主题，并成功采集到日志数据。
2. 子账号/协作者需要主账号授权，授权步骤参见 基于 CAM 管理权限，复制授权策略参见 CLS 访问策略模板 。

消费组消费流程

使用消费组消费数据时，服务端会管理消费组里的所有消费者的消费任务，根据主题分区和消费者的数量关系自动调整消费任务的均衡性，同时会记录每个主题分区的消费进度，保证不同消费者可以无重复消费数据。消费组消费的具体流程如下：

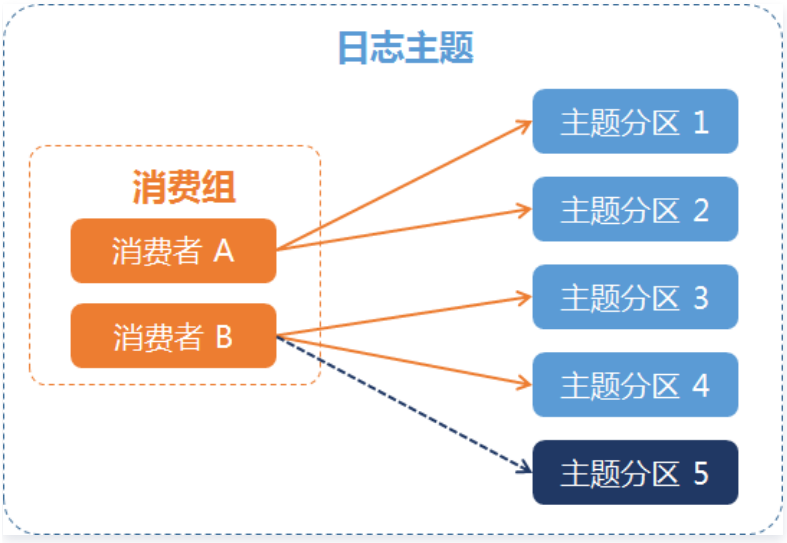
1. 创建消费组。
2. 每个消费者定期向服务端发送心跳。
3. 消费组根据主题分区负载情况自动分配主题分区给消费者。
4. 消费者根据所分配的分区列表，获取分区 offset 并消费数据。
5. 消费者周期性地更新分区的消费进度到消费组，便于下次消费组分配任务。
6. 重复步骤2 – 步骤6，直至消费结束。

负载均衡消费原理

消费组会根据活跃消费者和主题分区的数量动态调整每个消费者的消费任务，保证消费的均衡性。同时，消费者会保存每个主题分区的消费进度，保证故障恢复后可继续消费数据，避免重复消费。

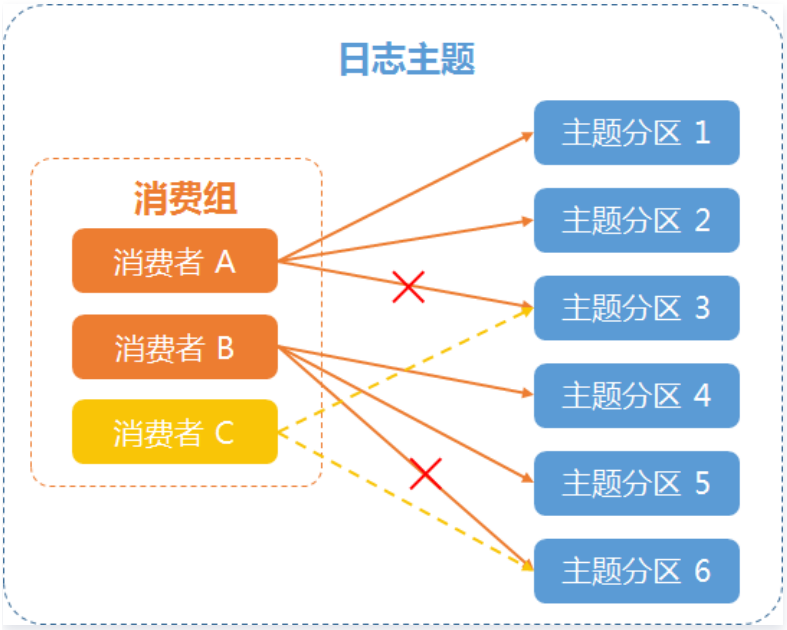
例一：主题分区发生变化

例如，某个日志主题有两个消费者，消费者 A 负责消费1、2号分区，消费者 B 负责消费3、4号分区，通过分裂操作新增主题分区5后，消费组会自动将5号分区分配给消费者 B 进行消费，如下图所示：



例二：消费者发生变化

例如，某个日志主题有两个消费者，消费者 A 负责消费1、2、3号分区，消费者 B 负责消费4、5、6号分区，为保证消费速度持平生成速度，新增一个消费者 C，消费组会重新进行均衡分配，将3、6号分区分配给新消费者 C 进行消费，如下图所示：



消费 Demo（Python）

> **说明：**
> Python Demo 完整示例请参见 tencentcloud-cls-sdk-python/tests，推荐使用3.5及以上 Python 版本进行数据消费。

Demo 的使用和说明如下：

1. 安装 SDK，具体可参见 tencentcloud-cls-sdk-python 。

```
pip install git+https://github.com/TencentCloud/tencentcloud-cls-sdk-python.git
```

2. 消费者处理消费的数据，用户的日志数据被保存到 log_group 结构体中。log_group 结构体如下：

```
log_group {
    source    //日志来源，一般为机器的 IP
    filename  //日志文件名
    logs {
        time //日志时间，unix 时间戳，微秒级别
        user_defined_log_kvs   //用户日志字段
    }
}
```

下面是 SampleConsumer 方法的实现：

```
class SampleConsumer(ConsumerProcessorBase):
    last_check_time = 0
    log_results = []
    lock = RLock()

    def initialize(self, topic_id):
        self.topic_id = topic_id

    # 处理消费的数据
    def process(self, log_groups, offset_tracker):
        for log_group in log_groups:
            for log in log_group.logs:
                # 处理单行数据
```

```
                item = dict()
                item['filename'] = log_group.filename
                item['source'] = log_group.source
                item['time'] = log.time
                for content in log.contents:
                    item[content.key] = content.value

                with SampleConsumer.lock:
                    # 数据汇总到SampleConsumer.log_results
                    SampleConsumer.log_results.append(item)

        # 每隔3s提交一次offset
        current_time = time.time()
        if current_time - self.last_check_time > 3:
            try:
                self.last_check_time = current_time
                offset_tracker.save_offset(True)
            except Exception:
                import traceback
                traceback.print_exc()
        else:
            try:
                offset_tracker.save_offset(False)
            except Exception:
                import traceback
                traceback.print_exc()

        return None

    # Worker退出时，会调用该函数，可以在此处执行清理工作
    def shutdown(self, offset_tracker):
        try:
            offset_tracker.save_offset(True)
        except Exception:
            import traceback
            traceback.print_exc()
```

3. 创建消费者并启动消费者线程，该消费者会从指定的主题中消费数据。

配置参数	说明	默认值	取值范围
endpoint	请求域名，API 上传日志标签页面的域名。	-	支持地域：ALL
access_key_id	用户的 Secret_id，请前往 CAM 查看。	-	-
access_key	用户的 Secret_key，请前往 CAM 查看。	-	-
region	主题所在地域，例如 ap-beijing、ap-guangzhou、ap-shanghai，详情请参见 地域和访问域名 。	-	支持地域：ALL
logset_id	日志集 ID，仅支持一个日志集。	-	-
topic_ids	日志主题 ID，多个主题请使用','隔开。	-	-
consumer_group_name	消费者组名称。	-	-
internal	• 内网：TRUE • 公网：FALSE 说明：	FALSE	TRUE/FALSE

	内网/外网读流量费用请参见 产品定价 。		
consumer_name	消费者名称。同一个消费者组内，消费者名称不可重复。	-	0-9、aA-zZ、'-'、'_'、'.'组成的字符串
heartbeat_interval	消费者心跳上报间隔，2个间隔没有上报心跳，会被认为是消费者下线。	20	0-30分钟
data_fetch_interval	消费者拉取数据间隔，不小于1秒。	2	-
offset_start_time	拉取数据的开始时间，字符串类型的 UNIX 时间戳，精度为秒，例如 "1711607794"，也可以直接可配置为"begin"、"end"。 • begin：日志主题生命周期内的最早数据 • end：日志主题生命周期内的最新数据	"end"	"begin"/"end"/UNIX 时间戳
max_fetch_log_group_size	消费者单次拉取数据大小，默认2M，最大10M。	2097152	2M - 10M
offset_end_time	拉取数据的结束时间，支持字符串类型的 UNIX 时间戳，精度为秒，例如"1711607794"。**不填写代表持续拉取。**	-	-

注意：
如果您指定了开始时间（例如 offset_start_time=1711607794）消费了一次，那么第二次再指定从这个时间消费时，需要修改消费组的名称，否则会从上次消费的 offset 开始消费。

```
def sample_consumer_group():
    # 日志服务接入点，请您根据实际情况填写
    endpoint = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_ENDPOINT', '')
    # 访问的地域
    region = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_REGION', '')
    # 用户的Secret_id
    access_key_id = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_ACCESSID', '')
    # 用户的Secret_key
    access_key = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_ACCESSKEY', '')
    # 消费的日志集ID
    logset_id = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_LOGSET_ID', '')
    # 消费的日志主题ID列表，多个主题用英文逗号分隔
    topic_ids = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_TOPICS', '').split(',')
    # 消费组名称，同一个日志集下的消费组名称唯一
    consumer_group = 'consumer-group-1'
    # 消费者名称
    consumer_name1 = "consumer-group-1-A"
    consumer_name2 = "consumer-group-1-B"

    assert endpoint and access_key_id and access_key and logset_id,
ValueError("endpoint/access_id/access_key and "
                                                    "logset_id cannot be
empty")
    # 创建访问云API接口的Client
    client = YunApiLogClient(access_key_id, access_key, region=region)
    SampleConsumer.log_results = []

    try:
        # 创建两个消费者配置
        option1 = LogHubConfig(endpoint, access_key_id, access_key, region, logset_id, topic_ids,
consumer_group,
                               consumer_name1, heartbeat_interval=3, data_fetch_interval=1,
```

```
                                offset_start_time='end', max_fetch_log_group_size=1048576)
        option2 = LogHubConfig(endpoint, access_key_id, access_key, region, logset_id, topic_ids,
consumer_group,
                                consumer_name2, heartbeat_interval=3, data_fetch_interval=1,
                                offset_start_time='end', max_fetch_log_group_size=1048576)

        # 创建消费者
        print("*** start to consume data...")
        client_worker1 = ConsumerWorker(SampleConsumer, consumer_option=option1)
        client_worker2 = ConsumerWorker(SampleConsumer, consumer_option=option2)

        # 启动消费者
        client_worker1.start()
        client_worker2.start()

        # 等待2分钟，或者获取到数据后继续往后执行
        sleep_until(120, lambda: len(SampleConsumer.log_results) > 0)

        # 关闭消费者
        print("*** stopping workers")
        client_worker1.shutdown()
        client_worker2.shutdown()

        # 打印汇总的日志数据
        print("*** get content:")
        for log in SampleConsumer.log_results:
            print(json.dumps(log))

        # 打印消费组信息：消费组的名称、消费的日志主题、消费者心跳超时时间
        print("*** consumer group status ***")
        ret = client.list_consumer_group(logset_id, topic_ids)
        ret.log_print()

        # 删除消费组
        print("*** delete consumer group")
        time.sleep(30)
        client.delete_consumer_group(logset_id, consumer_group)
    except Exception as e:
        import traceback
        traceback.print_exc()
        raise e


if __name__ == '__main__':
    sample_consumer_group()
```

指标主题投递

最近更新时间：2025-06-27 11:06:31

操作场景

指标（Metric）采集到 CLS 后，支持将其投递至开启了 Remote Write 的时序数据库。

前提条件

1. 已有 指标主题，并成功采集到指标数据。
2. 您的时序数据库已经**配置了 RemoteWrite 地址**。例如 Prometheus 时序数据库，可前往 Prometheus.yml 配置文件中查看 remote_write.url 配置项。
3. 子账号/协作者需要主账号授权，授权步骤参见 子账号授权，授权策略参见 CLS 访问策略模板 。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**指标主题**。
3. 单击需要投递的指标主题 ID/名称，进入指标主题管理页面。
4. 单击**指标投递**，进入投递配置页面，单击**新建**。

   配置项说明如下：

配置项	解释说明	限制	是否必填
任务名称	配置投递任务的名称	不超过128个字符	必填
服务日志	默认开启，将投递任务产生的日志投递至 cls_service_log（免费）	-	-
投递目标	• 腾讯云 Prometheus：投递至 腾讯云 Prometheus 实例 **说明：** 腾讯云 Prometheus 仅需选择实例即可，无需配置其余参数。 • 自建 Prometheus：投递至支持 Prometheus 协议的时序数据库，例如 Prometheus\Cortex\VictoriaMetrics\Thanos	提前配置好投递目标的 RemoteWrite	-
网络访问	• **内网**：通过内网访问投递目标，例如您的 Prometheus 部署在同地域的 CVM 云服务器上，那么选择内网 • **公网**：**暂不支持**	投递目标和指标主题在同一个地域	必填
所属网络（内网）	投递目标所属的 VPC 网络	-	必填
网络服务类型（内网）	• **CVM**：投递目标直接部署在腾讯云 CVM 云服务器 • **CLB**：投递目标的服务地址和端口通过腾讯云 CLB（负载均衡）转发	-	-
Remote Write 地址	**例如**：http://192.168.2.17:9090/api/v1/prom/write	-	必填
鉴权方式	通过 RemoteWrite 访问时序数据库的鉴权方式： • **BASIC AUTH**：需要输入用户名、密码 • **无鉴权**	-	必填
校验连通性	校验 CLS 和投递目标之间的网络是否联通 **说明：** 连通性校验通过之后，方可提交指标投递配置。	-	必填

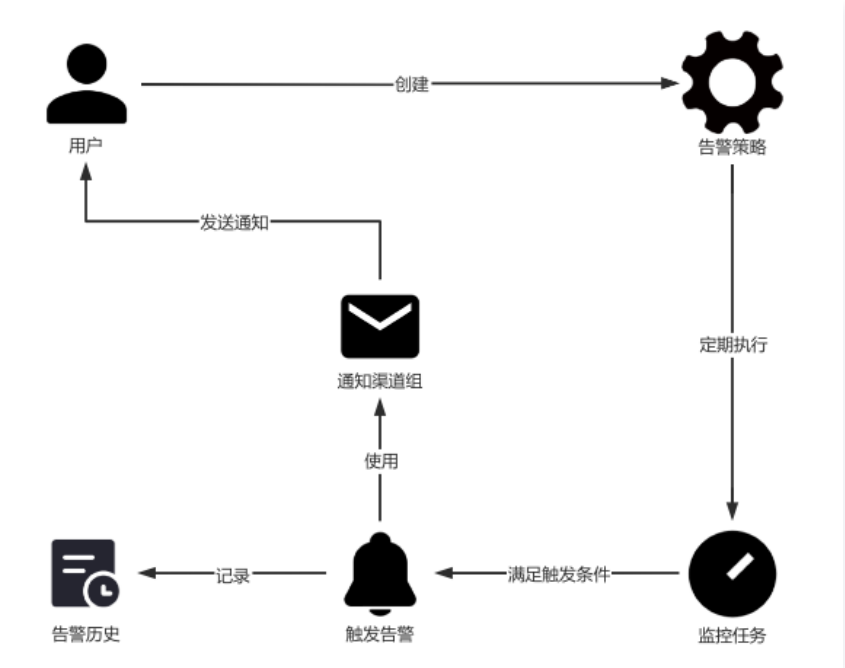
5. 单击**确定**，提交投递任务。

监控告警
监控告警概述

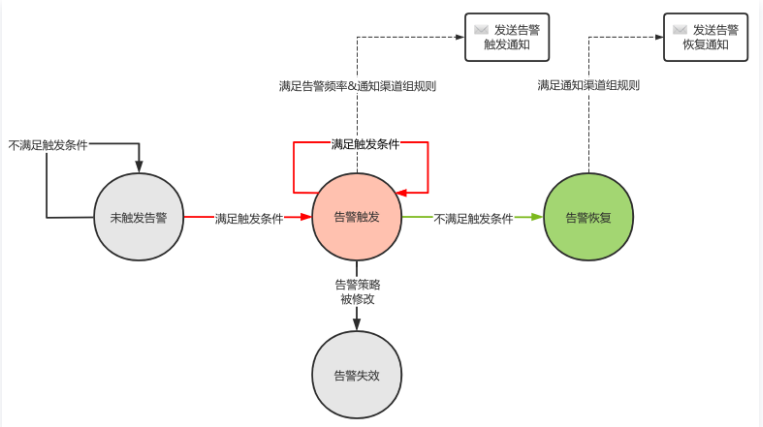
版权所有：腾讯云计算（北京）有限责任公司

最近更新时间：2024-07-11 16:43:42

日志服务支持对日志及指标主题设置告警策略，告警策略会周期性执行监控任务，对监控对象中的主题执行检索分析，当查询结果满足触发条件时发送告警通知，方便用户及时发现异常问题。



基本概念

名称		描述
告警策略		监控告警的配置单元，包括监控对象、执行周期、触发条件、告警频率和通知渠道等信息
监控对象		需要告警的日志或指标主题，告警策略会对该主题执行检索分析语句（例如统计错误日志条数），然后检查结果是否满足触发条件
监控任务	执行语句	对监控对象执行的检索分析语句，语法规则与检索分析一致
	触发条件	监控对象的检索分析结果满足指定条件时（例如错误日志数大于100条），则触发告警 检索分析结果不再满足执行条件时，则告警恢复
	执行周期	策略执行周期，支持固定频率（例如每隔5分钟）和固定时间（例如每天12:00）
告警	告警等级	不同的触发条件可触发不同等级的告警，包括提醒、警告和紧急三种等级
	告警状态	首次满足触发条件时告警为未恢复状态，不再满足触发条件时告警为已恢复状态，告警策略被编辑时未恢复的告警会被置为已失效状态 

	告警历史	可查看当前未恢复的告警及已恢复的历史告警记录
告警通知	告警频率	告警持续未恢复时，每隔多久发送一次告警通知，避免告警通知过于频繁
	通知渠道组	• 支持短信、微信、语音、邮件、企业微信、钉钉、飞书、自定义接口回调等告警通知方式 • 可针对告警设定不同的规则以分别使用不同的通知渠道，例如“紧急”告警使用电话进行通知，“警告”告警使用短信进行通知，“提醒”告警不通知 • 支持将告警策略执行过程以日志形式投递至日志主题中
	通知内容模板	支持自定义告警通知内容及格式
	集成配置	统一管理告警通知对接企业微信、钉钉、飞书机器人的地址及自定义接口回调地址

产品优势

- 告警策略执行语句语法规则与检索分析一致：
  - 日志主题支持 SQL，并符合 SQL-92规范，可使用灵活的 SQL 语法及400+ SQL 函数对日志进行统计，从日志中获取各种需要告警的指标。
  - 指标主题支持 PromQL，可进行灵活的指标运算及统计。
- 使用多维分析功能，可在告警通知中直接查看告警详细原因，提升问题处理效率。

费用说明

监控告警功能不产生任何费用。短信及语音通知渠道目前也供免费使用，后续如有收费计划将至少提前1个月通知您。

规格与限制

告警功能共享检索分析的规格与限制，请参见 规格与限制 。

版权所有：腾讯云计算（北京）有限责任公司

管理告警策略
配置告警策略

最近更新时间：2024-10-15 18:50:51

操作场景

基于日志或指标配置告警策略，在出现符合特定条件的日志或指标时发送告警，例如 error 日志数量过多、接口响应时间过长等。

前提条件

- 数据已上传到日志或指标主题。
- 日志主题已 配置索引，且不支持 低频存储 。

操作步骤

登录日志服务控制台，并进入 告警策略 管理页面，在左上角单击**新建**，配置如下主要信息。

1. 基本信息

- **告警策略名称**：字符长度为1至255个字符。
- **启用状态**：是否启用当前告警策略，告警策略不启用时不产生任何告警及通知。
- **标签**：为当前创建的告警策略设定标签，便于从不同维度对资源分类管理。

2. 监控对象

选择需要监控的日志或指标主题，支持两种选择方式：

- **所有执行语句使用相同的监控对象（默认）**：支持多选，最多可选择20个主题，不支持跨地域。每个主题单独判断是否满足触发条件，同时有多个主题满足触发条件时将同时产生多条告警。
- **在每个执行语句中单独选择监控对象**：可为每个执行语句单独设置主题，单选，不支持跨地域。每个执行语句的结果合并在一起判断是否满足触发条件。例如 `A 主题错误日志数 + B 主题错误日志数 > 10` 时触发一条告警。

3. 监控任务

- **执行语句**：作用于日志或指标主题的查询语句，

日志主题

执行语句：日志检索分析语句，详见 语法规则 。
- 例1：状态为 error 日志 `status:error`
- 例2：统计域名 domain:aaa.com 的平均响应时间 `domain:"aaa.com" | select avg(request_time) as Latency`

查询时间范围：最大时间范围为最近24小时

指标主题

执行语句：PromQL，详见 语法规则 。
- 例1：文件系统空间占用率大于 50% 时触发告警 `filesystem_usage_pct > 0.5`
- 例2：最近5分钟 CPU 负载最大值大于 1 时触发告警 `max_over_time(node_load1[5m]) > 1`

- **触发条件**：满足触发条件时进行告警，支持“交互模式”及“语句模式”两种配置方式。
  - 交互模式：
    - 依次选择告警等级、执行语句、变量名称和比较方式，然后设定告警阈值。
    - 支持设定多个告警等级，满足条件时触发对应等级的告警，同时满足多个条件时将触发多个告警。可根据告警级别在通知渠道组中设定不同的通知规则，例如“紧急”告警使用电话进行通知，“警告”告警使用短信进行通知，“提醒”告警不通知。
  - 语句模式：

- 告警等级：支持提醒、警告和紧急三种级别，满足条件时触发对应等级的告警，同时满足多个条件时将触发多个告警。
- 表达式：语法规则参见 触发条件表达式 。

- **分组触发**：是否需要使用分组触发功能。
  - 使用分组触发功能时，如果某一个执行语句有多条结果，将根据指定的分组字段对这多条结果进行分组，按组分别判断是否满足告警触发条件，满足触发条件的每个组分别单独触发告警。
  - 分组字段：从执行语句执行结果中选择需要作为分组依据的字段，至少选择1个字段。
  - 例如：
    - 执行语句：
      ```
      * | select avg(request_time) as request_time, domain group by domain order by request_time desc limit 5
      ```
    - 触发条件：`$1.request_time > 5`
    - 执行语句结果：

不使用分组触发

共触发一条告警：

domain	request_time	执行结果
aaa.com	12.56	满足触发条件，共触发一条告警
bbb.com	9.45	
ccc.com	7.23	
ddd.com	5.21	
eee.com	4.78	不满足触发条件，不触发告警

使用分组触发

以 `$1.domain` 作为分组字段，分为5组，组1～组4满足触发条件，各单独触发一条告警，共触发4条告警：

domain	分组结果	request_time	执行结果
aaa.com	组1	12.56	满足触发条件，单独触发一条告警
bbb.com	组2	9.45	满足触发条件，单独触发一条告警
ccc.com	组3	7.23	满足触发条件，单独触发一条告警
ddd.com	组4	5.21	满足触发条件，单独触发一条告警
eee.com	组5	4.78	不满足触发条件，不触发告警

> ⚠ **注意：**
> - 使用分组触发功能时，一次告警执行可能会有大量的执行结果满足触发条件，进而同时产生大量告警，造成告警风暴。请合理设定分组字段及触发条件。
> - 指定分组字段时，最多可将执行结果分为1000个组，超出范围的组将不触发告警。

- **执行周期**：监控任务的执行频率，提供如下两种周期配置方式：

周期配置方式	说明	示例
固定频率	按固定的时间间隔执行一次监控任务，支持1~1440分钟	每隔5分钟执行一次监控任务

固定时间	按固定的时间点执行一次监控任务，支持00:00~23:59	每天02:00点执行一次监控任务
Cron 表达式	按照 Cron 表达式 执行监控任务，最小粒度为分钟	`*/5 8-20 * * *` 每天的8点 - 20点，每5分钟执行一次监控任务

4. 多维分析

触发告警时，可通过多维分析对原始日志做进一步的分析，并将分析结果附加在告警通知中，以辅助定位告警原因。多维分析不会影响告警触发条件。

多维分析类型	说明
相关原始日志	获取符合执行语句检索条件的原始日志，可配置需要展示的日志字段、条数及展示形式。 例如 error 日志过多触发告警时，在告警中查看对应的详细日志。
字段 TOP5 及占比统计	针对触发告警时间范围内的全部日志，按指定字段分组统计日志条数，获取前5项字段值及其占比。 例如 error 日志过多触发告警时，日志对应的 Top5 URL、Top5 响应状态码等。
自定义检索分析	针对触发告警时间范围内的全部日志，执行自定义检索分析语句。 例1：通过 `* | select avg(timeCost) as time,URL group by URL` 获取每个接口请求耗时。 例2：通过 `status:>499` 获取错误日志。

说明

- “相关原始日志”和“字段 TOP5 及占比统计”支持自动关联指定的执行语句中的检索条件（不包含 SQL 中的过滤条件，例如 where），表示需要对符合条件的原始日志进行多维分析。
- 指标主题暂不支持多维分析。

5. 告警通知

- **告警频率：**
  - 持续周期：当持续满足触发条件达到一定次数（默认为1，有效值范围：1 - 10）以后才发送通知。
  - 间隔时间：上一次发送通知后，指定间隔时间内不再发送通知。例如，每15分钟告警一次，表示15分钟内最多只会收到一次告警通知。
- **通知渠道组：**
  通过关联通知渠道组，设置发送通知的方式及对象，支持短信、邮件、电话、微信、企业微信、钉钉、飞书、自定义接口回调（webhook）等通知方式。详情参见 管理通知渠道组 。
  如果选中的通知渠道组中包含自定义接口回调，则会显示“单独设置自定义接口回调参数”开关。未启用该功能时自定义接口回调将使用通知内容模板中的配置作为请求头（Request Header）和请求内容（Request Body）。启用该功能时可在告警策略中单独定义请求头和请求内容，其中可使用 告警通知变量 。
- **附加分类字段：**
  对告警进行分类。采用键值对的形式(key:value)进行配置，key 仅支持字母、数字及下划线，且需以字母开头。key 最多可配置50字符，value 最多可配置200字符。可在通知渠道组中根据该字段配置不同的通知规则，也可在附加通知内容及自定义接口回调中使用 `{{.Label}}` 引用该字段。
  例如设置分类字段为 service:payService（代表当前告警对应的服务名称为 payService）：
  - 在通知渠道组中配置通知规则，当 service 包含 payService 时将告警发送至对应的团队。
  - 在自定义接口回调配置中使用 `{{.Label}}` 或 `{{.Label.service}}` 获取该字段的值（分别为 `{"service":"payService"}` 、 `payService` ），第三方平台收到接口回调时即可识别出该告警对应的服务名称。
- **附加通知内容：**
  在告警通知中额外增加一部分文字，使告警通知内容更加丰富，可使用 告警通知变量 动态生成。

执行周期与查询时间范围详细说明

日志服务在执行告警策略时会自动调度告警策略的实际执行时间，避免较多的告警策略集中在同一时刻执行而出现缓慢情况，因此告警策略的实际执行时间会在指定的告警执行周期基础上进行秒级别的变化。
例如12:00:00同时创建了两个告警策略，均为每分钟执行一次，但实际执行时间并不完全一致：

- 第一个告警策略的实际执行时间为12:01:12、12:02:12、12:03:12、12:04:11、12:05:11...
- 第二个告警策略的实际执行时间为12:01:27、12:02:27、12:03:27、12:04:27、12:05:27...

同时为了保证在自动调度的情况下，告警不遗漏，告警查询时间范围会自动取整并额外增加30秒查询延迟（数据上报一般会有10秒左右延迟，严格按实际时间执行告警时，很可能因为最新的数据尚未上报而出现告警遗漏，因此额外增加30秒查询延迟）。
例如告警策略每分钟执行一次，每次查询最近一分钟的数据，实际执行时间及对应的查询时间范围如下：

执行时间	查询时间范围
12:01:12	11:59:30 ~ 12:00:30
12:02:12	12:00:30 ~ 12:01:30
12:03:12	12:01:30 ~ 12:02:30
12:04:11	12:02:30 ~ 12:03:30
12:05:11	12:03:30 ~ 12:04:30

Cron 表达式

告警策略执行周期支持最小粒度为分钟的 Cron 表达式，由五个空格分隔的时间字段组成，每个字段代表一个时间单位。结构如下：

```
* * * * *
```

各字段含义如下：

- 第1个字段：分钟（0 – 59）
- 第2个字段：小时（0 – 23）
- 第3个字段：日（1 – 31）
- 第4个字段：月（1 – 12）
- 第5个字段：星期（0 – 6）（星期天为0）

> **注意：**
> 在所有的地域中，Cron 表达式执行时区均为 UTC+8 时区。

表达式中可以使用一些特殊字符来表示更复杂的时间规则：

- `*`：表示匹配任意值。假如在分钟字段使用 `*`，即表示每分钟都将执行一次。
- `?`：表示无指定字段，只能用在日和星期两个字段。
- `-`：表示匹配范围。例如在分钟字段使用 `5-20`，表示在5分到20分钟之间，每分钟执行一次 。
- `/`：表示从起始时间开始执行，每隔固定时间执行一次。例如在分钟字段使用 `5/20`，表示从5分开始，每隔20分钟执行一次，也即25分、45分各执行一次。
- `,`：表示列出枚举值。例如：在分钟字段使用 `5,20`，表示在5分和20分各执行一次。

示例：

表达式	含义
0 12 * * *	每天中午12:00执行一次
15 10 * * *	每天上午10:15执行一次
* 14 * * *	每天的14:00 – 14:59时间段，每隔1分钟执行一次
0/5 14 * * *	每天的14:00 – 14:55时间段，每隔5分钟执行一次
0/5 14,18 * * *	每天的14:00 – 14:55，18:00 – 18:55，每隔5分钟执行一次
0-5 14 * * *	每天的14:00 – 14:05时间段，每隔1分钟执行一次
10,44 14 * 3 3	三月份每周三的下午14:10和14:44各执行一次
15 10 * * 1-5	每周周一至周五每天上午10:15执行一次

最佳实践

- 按时间段分别设置告警触发条件
- 使用同环比作为告警触发条件

- 按日志所属服务将告警发送到不同的团队

触发条件表达式

最近更新时间：2024-08-23 10:01:01

触发条件表达式用于判断是否触发告警。将执行语句结果作为触发表达式的输入变量，当表达式判断结果为真时触发告警。

语法说明

运算符	说明	示例

$N.__QUERYCOUNT__	执行语句结果条数，`N` 为对应的执行语句编号。
	针对不同使用场景，其具体含义如下：
	• 日志主题
	  ○ 执行语句不包含 SQL：匹配检索条件的原始日志条数
	  ○ 执行语句包含 SQL：SQL结果条数（注意：SQL 中未指定 limit 时，默认最大返回100条结果）
	• 指标主题：PromQL 查询结果条数
		$1.__QUERYCOUNT__

$N.keyname	引用执行语句的结果，`N` 为对应的执行语句编号，`keyname` 为**执行语句结果**中的字段名称。
	注意：
	• `keyname` 首字符必须是字母，可以包含字母、数字、下划线，如包含特殊字符：
	  ○ 日志主题执行语句中支持使用 AS 语法 对字段名称设置别名
	  ○ 使用方括号整体包裹，例如 `[$1.@value]`
	• 监控对象为指标主题时，除指标维度(label)外，系统预置如下变量：
	  ○ `$N.__name__`：指标的名称
	  ○ `$N.value`：指标的值
	  ○ `$N.time`：指标对应的时间，秒级 UNIX 时间戳
	  指标维度(label)的字段名称如果与上述变量冲突，会在原字段名称前自动添加@符号，例如 `[$1.@value]`
		$1.ErrCount
		$1.value

+	求和运算符	$1.ErrCount+$1.FatCount>10
-	减法运算符	$1.Count-$1.InfoCount>100
*	乘法运算符	$1.RequestMilSec*1000>10
/	除法运算符	$1.RequestSec/1000>0.01
%	取模运算符	$1.keyA%10==0
==	比较运算符：等于	$1.ErrCount==100 $1.level=="Error"
>	比较运算符：大于	$1.ErrCount>100
<	比较运算符：小于	$1.pv<100
>=	比较运算符：大于等于	$1.ErrCount>=100
<=	比较运算符：小于等于	$1.pv<=100
!=	比较运算符：不等于	$1.level!="Info"
()	括号，控制运算优先级	($1.a+$1.b)/$1.c>100
&&	逻辑运算符：与	$1.ErrCount>100 && $1.level=="Error"
||	逻辑运算符：或	$1.ErrCount>100 || $1.level=="Error"

- 需使用比较运算符对变量进行比较，且结果需为布尔值且为“真”时才会触发告警。例如 `$1.a` 值为80，`$1.b` 值为20时：
  - 表达式 `$1.a+$1.b`：结果为100，不是布尔值，不触发告警。
  - 表达式 `$1.a+$1.b>=100`：结果为“真”，触发告警。

- 一个告警策略中可以设置多个执行语句（最多3个），每个执行语句有编号（从1开始依次递增），例如 `$1.key1` 引用编号为 `1` 的执行语句中的 `key1` 字段， `$2.key2` 引用编号为 `2` 的执行语句中的 `key2` 字段。
- 监控对象为指标主题时，建议在执行语句中直接配置好告警条件，触发条件中直接使用 `$N.__QUERYCOUNT__ > 0` 即可（请将其中的 N 替换为执行语句编号），详见 示例3 。
- 当执行语句返回多个结果时，会根据返回结果依次计算1000次，判断其是否满足触发条件，当计算结果为真时停止。例如，表达式为 `$1.a+$2.b>100` ，若执行语句1返回 m 条结果，执行语句 2 返回 n 条结果，则会进行 m * n 次计算，计算过程中当结果满足 `$1.a+$2.b>100` 为真时停止或计算超过1000次停止。超过1000次仍未满足触发条件时，认为本次判断结果为假，不触发告警。

示例

示例1:【日志主题】当近5分钟出现 error 级别日志时触发告警

- 监控对象：日志主题
- 执行语句：

```
level:error
```

- 查询时间范围：近5分钟
- 触发条件：执行语句1的结果条数大于0

```
$1.__QUERYCOUNT__>0
```

示例2:【日志主题】当某 VIP 用户（uid:10001）的请求平均延时大于10s 时触发告警

- 监控对象：日志主题
- 执行语句：

```
uid:10001 | select avg(request_time) as latency
```

- 查询时间范围：近5分钟
- 触发条件：执行语句1的 latency 大于10

```
$1.latency>10
```

示例3:【指标主题】文件系统空间占用率大于 50% 时触发告警

- 监控对象：指标主题
- 执行语句：

```
filesystem_usage_pct > 0.5
```

- 触发条件：执行语句1的结果条数大于0

```
$1.__QUERYCOUNT__>0
```

`__QUERYCOUNT__` 为系统预置变量，表示执行语句的结果条数。执行语句中已经通过 `> 0.5` 对指标进行了过滤，所以只要执行语句的结果条数大于 0 就需触发告警。

版权所有：腾讯云计算（北京）有限责任公司

管理通知渠道组

最近更新时间：2025-04-15 15:22:11

操作场景

通知渠道组用于在告警策略产生告警时发送告警通知。您可在通知渠道组中设定告警通知的接收对象以及具体的通知方式，例如短信或邮件。

操作步骤

新增通知渠道组

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**监控告警 > 通知渠道组**，进入通知渠道组管理页面。
3. 单击**新建**，并在**新建通知渠道组**中，填写如下信息：

- 基本信息
  - 渠道组名称：自定义渠道组名称。
  - 查询数据链接：告警通知中查询详细数据使用的链接，一般情况下不用修改。通过 DataSight 使用 CLS 时建议使用 DataSight 域名，以便收到告警时在 DataSight 中查看告警详细数据。
- 通知规则：指定哪些告警需要发送告警通知以及具体的通知渠道。
  - 规则：
    - 支持按照告警等级、通知类型、告警策略、告警分类字段、通知时间及告警持续时间设定条件，多个条件之间可配置“且/或”关系。
    - 产生告警时将根据规则顺序由上至下依次判断是否满足，满足时将使用该规则对应的通知渠道发送通知。

      > **注意：**
      > 满足某条规则时，判断流程将结束，后续规则不再执行。

    - 每个通知渠道组最多可添加50个规则，每个规则下最多可添加10个通知渠道。“邮件、短信、微信、电话”类型下最多可勾选50个用户或用户组。
    - 例如“紧急”告警使用电话及短信进行通知，“警告”告警使用短信进行通知，“提醒”告警则不进行通知。



  - 通知渠道：发送告警通知使用的渠道
    - 邮件、短信、微信、电话
      - 接收对象：对指定用户或用户组（包含多个用户）进行通知触达。
      - 内容模板：选择告警通知 内容模板，在内容模板中可自定义告警通知内容及格式。
      - 通知时段：可接收告警通知的时间段，支持跨天，例如22:00:00～08:00:00代表每天晚上22点至次日早上8点发送通知。
      - 接收渠道：勾选具体需要的通知渠道。

        > **注意：**
        > - 接收对象需在 访问管理-用户列表 中添加，且配置手机号码、邮件、微信等信息时需进行验证。
        > - 电话告警存在发送频率限制，建议仅通过电话接收非常紧急的告警。具体频率限制如下：

- 同一号码30秒内最多接收1条电话告警，10分钟内最多接收3条电话告警，1天内最多接收50条电话告警。
- 同一号码22:00至次日08:00最多接收3条电话告警，如需去除该限制，请在 消息中心 中开启**夜间接收状态**开关。
- 以上限制不局限于日志服务产生的电话告警通知，腾讯云可观测平台及消息中心产生的电话通知会共享该限制。

使用电话接收告警时，建议您将以下电话号码添加至手机及运营商骚扰电话拦截白名单中，以免被拦截：

```
02155931000,02155931001,02155931002,02155931003,02155931004,02155931005,021559310
06,02155931007,02155931008,02155931009,02155931010,02155931011,02155931012,021559
31013,02155931014,02155931015,02155931016,02155931017,02155931018,02155931019,021
55931020,02155931021,02155931022,02155931023,02155931024,02155931025,02155931026,
02155931027,02155931028,02155931029,02155931030,02155931031,02155931032,021559310
33,02155931034,02155931035,02155931036,02155931037,02155931038,02155931039,021559
31040,02155931041,02155931042,02155931043,02155931044,02155931045,02155931046,021
55931047,02155931048,02155931049,02155931050,02155931051,02155931052,02155931053,
02155931054,02155931055,02155931056,02155931057,02155931058,02155931059,010213646
89,01021364720
```

- 企业微信、钉钉、飞书
  - Webhook 地址：企业微信、钉钉、飞书机器人 Webhook 地址，请在 集成配置 中提前创建好。
  - 提醒方式：企业微信及钉钉支持在群消息中@指定人员或@所有人员，飞书仅支持@所有人员。
  - 内容模板：选择告警通知 内容模板，在内容模板中可自定义告警通知内容及格式。
  - 通知时段：可接收告警通知的时间段，支持跨天，例如22:00:00～08:00:00代表每天晚上22点至次日早上8点发送通知。

> **注意：**
> 每个机器人发送的消息不能超过20条/分钟。若您的告警策略较多，建议创建多个机器人，分散绑定告警策略。避免多个告警策略在同一时间触发告警，导致您无法接收部分告警通知。

- 自定义接口回调
  - Webhook 地址：选择在 集成配置 中创建好的自定义接口回调地址，或选择“自定义”，在此直接输入接口回调地址。Webhook 地址需支持通过公网进行访问，日志服务无法回调内网地址。
  - 请求方法：Webhook 地址为“自定义”时需配置，支持 POST 及 PUT。
  - 内容模板：选择告警通知 内容模板，自定义回调没有固定的请求信息，需在通知内容模板中按需进行配置，包括请求头（Request Header）及请求内容（Request Body）。
  - 通知时段：可接收告警通知的时间段，支持跨天，例如22:00:00～08:00:00代表每天晚上22点至次日早上8点发送通知。

> **注意：**
> - 每个自定义回调接口发送的消息不能超过20条/分钟。若您的告警策略较多，建议多创建几个自定义回调接口，分散绑定告警策略。避免多个告警策略在同一时间触发告警，导致您无法接收部分告警通知。
> - 自定义回调接口 HTTP 返回状态码为2xx 时，认为回调成功。状态码为4xx 及5xx 时，回调失败，不支持重定向（状态码为3xx）。自定义回调出口 IP 会动态更新，如需固定 IP 请联系 技术支持。

- 告警升级：当告警长时间未认领或未恢复时，可自动升级至下一个通知环节，每个通知环节可包含多个通知渠道，详情请参见 告警认领与升级。
- 高级配置：
  - 统一设定自定义回调参数：当告警策略中单独设置自定义接口回调参数时，启用该开关后将使用 内容模板 覆盖告警策略中的配置，便于批量设定回调参数。
  - 免登录操作告警：开启后允许在告警通知中直接屏蔽/认领相关告警，无需登录腾讯云控制台。默认开启，详情请参见 告警屏蔽 和 告警认领与升级。
  - 投递日志：开启后将告警执行日志投递至日志主题中，该主题将按数据量产生费用。
    - 目标日志主题：需要将告警执行日志投递到哪个日志主题中，支持跨地域选择日志主题，建议使用标准存储日志主题。
    - 投递数据范围：
      - 全部日志：告警策略每次执行所产生的日志，包括未触发告警的情况。推荐使用该选项。
      - 仅告警触发及恢复日志：告警策略发送“触发”及“恢复”通知时所产生的日志。

> **说明：**
> 投递到日志主题中后，建议为该日志主题开启键值索引，并开启 自动配置 功能，以便于进行统计分析（例如使用 预置仪表盘 分析数据）。投递日志的字段说明详见 告警通知变量 。

复制通知渠道组

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**监控告警 > 通知渠道组**，进入通知渠道组管理页面。
3. 选择需要复制的通知渠道组，单击**复制**。

修改通知渠道组

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**监控告警 > 通知渠道组**，进入通知渠道组管理页面。
3. 选择需要修改的通知渠道组，单击**编辑**。

删除通知渠道组

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**监控告警 > 通知渠道组**，进入通知渠道组管理页面。
3. 选择需要删除的通知渠道组，单击**删除**。

版权所有：腾讯云计算（北京）有限责任公司

管理通知内容模板

最近更新时间：2025-03-18 18:18:02

操作场景

通知内容模板用于配置告警通知的具体内容及格式，便于您在收到告警通知时更清晰直观的查看告警信息。
内容模板由固定部分及自定义部分组成：

- 固定部分：固定内容，不支持编辑及删除；
- 自定义部分：自定义内容，可使用 告警通知变量 按需进行配置。

日志服务平台预置了中文和英文两个默认模板，并且已提前配置好常用的告警内容，可直接使用。默认模板不支持编辑及删除，由日志服务平台维护，具体内容会随产品升级发生变化，如需严格控制通知内容请单独新建通知内容模板。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**监控告警 > 通知内容模板**，进入通知内容模板管理页面。
3. 单击**新建**，并在“新建通知内容模板”中，填写如下信息：
   - 模板名称
   - 语言：支持中文及英文，将决定内容模板中“固定部分”使用的语言。
   - 通知内容：按渠道分别配置告警触发及恢复时的内容模板，可使用 告警通知变量 动态渲染通知内容。

   > **说明：**
   > - 自定义部分为空时平台将使用默认内容。
   > - 除自定义回调外，每类渠道的内容长度均有上限，在渲染后如果内容超出上限将自动截断。
   > - 企业微信、钉钉及飞书支持 Markdown 语法 。

4. 单击**保存**，完成通知内容模板配置。
5. 在通知渠道组中选择需要使用的内容模板，操作步骤详见 管理通知渠道组 。

Markdown 语法

企业微信、钉钉及飞书支持 Markdown 语法，具体支持的语法如下：

企业微信 Markdown 语法

> **注意：**
> 企业微信在通知渠道组内使用@提醒方式时，不支持 Markdown 语法。

标题

```
# 标题一
## 标题二
### 标题三
#### 标题四
##### 标题五
###### 标题六
```

> **注意：**
> - `#` 与文字中间要有空格。
> - 标题需另起一行，`#` 前方不能有其它字符。在告警策略“附加通知内容”首行使用时请在上方添加一个空行。

加粗

```
**bold**
```

链接

```
[这是一个链接](http://work.weixin.qq.com/api/doc)
```

行内代码段（暂不支持跨行）

```
`code`
```

引用

```
> 引用文字
```

字体颜色（只支持3种内置颜色）

```
<font color="info">绿色</font>
<font color="comment">灰色</font>
<font color="warning">橙红色</font>
```

钉钉 Markdown 语法

标题

```
# 一级标题
## 二级标题
### 三级标题
#### 四级标题
##### 五级标题
###### 六级标题
```

> **注意：**
> - `#` 与文字中间要有空格。
> - 标题需另起一行，`#` 前方不能有其它字符。在告警策略“附加通知内容”首行使用时请在上方添加一个空行。

引用

```
> A man who stands for nothing will fall for anything.
```

文字加粗、斜体

```
**bold**
*italic*
```

链接

```
[this is a link](http://example.com)
```

图片

```
![](http://example.com/pic.jpg)
```

无序列表

```
- item1
- item2
```

有序列表

```
1. item1
2. item2
```

飞书 Markdown 语法

加粗

```
**粗体**
__粗体__
```

斜体

```
*斜体*
```

删除线

```
~~删除线~~
```

字体颜色

```
<font color='green'>
  这是一个绿色文本
</font>
<font color='red'>
  这是一个红色文本
</font>
<font color='grey'>
  这是一个灰色文本
</font>
```

> **说明：**
> color 取值参考 颜色枚举值 。

文字链接

```
[开发文档](https://open.feishu.cn)
```

图片

```
![hover_text](image_key)
```

标签

```
<text_tag color='red'>标签文本</text_tag>
```

> **说明：**
> color 支持的枚举值范围包括：
> - neutral：中性色

- blue：蓝色
- turquoise：青绿色
- lime：酸橙色
- orange：橙色
- violet：紫罗兰色
- indigo：靛青色
- wathet：天蓝色
- green：绿色
- yellow：黄色
- red：红色
- purple：紫色
- carmine：洋红色

有序列表

```
1. 有序列表1
    1. 有序列表 1.1
2. 有序列表2
```

无序列表

```
- 无序列表1
    - 无序列表 1.1
- 无序列表2
```

告警通知变量

最近更新时间：2025-06-26 18:56:52

在告警策略中配置通知内容、自定义接口回调配置和多维分析时，可使用告警通知变量来自定义告警通知内容和多维分析语句，使得收到的告警通知更加清晰准确地描述告警原因。
告警变量还能够以日志的形式投递到日志主题中，使用方式请参见 管理通知渠道组 。

快速开始

配置告警策略 时，在附加通知内容中填写如下信息：

```
详细日志：
{{.QueryLog[0][0]}}
```

配置完成后如下图所示：



收到告警通知时，附加通知内容将自动替换为下面的值，代表触发告警时最近的一条详细日志。

```
详细日志：
{"content":{"body_bytes_sent":"33352","http_referer":"-","http_user_agent":"Mozilla/5.0 (Windows NT 6.2;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.17
Safari/537.36","remote_addr":"201.80.83.199","remote_user":"-
","request_method":"GET","request_uri":"/content/themes/test-
com/images/header_about.jpg","status":"404","time_local":"01/Nov/2018:01:16:31"},"fileName":"/root/testLog
/nginx.log","pkg_id":"285A243662909DE3-
70A","source":"172.17.0.2","time":1653831150008,"topicId":"a54de372-ffe0-49ae-a12e-c340bb2b03f2"}
```

通知变量

变量列表

变量	含义	变量值示例	投递日志	说明
{{.RecordId}}	告警通知 ID	76afb409-4648-xxxx-b67a-fe350129ceb4	否	每次告警通知均使用不同的 ID。
{{.RecordGroupId}}	告警分组 ID	52746a79-7681-495f-xxxx-2715593e625a	是	• 告警从触发到恢复，无论中间重复发送了多少次通知，对应的 ID 均相同。 • 告警策略启用分组触发功能时，同一时间可能产生多条告警，分别对应不同的分组及 ID。
{{.UIN}}	账号 ID	100007xxx827	是	-
{{.Nickname}}	账号昵称	xx 企业	是	-
{{.Region}}	地域	广州	是	-

		注意： 投递日志时为英文，例如 ap-guangzhou。		
{{.Alarm}}	告警策略名称	Nginx 错误日志过多	是	-
{{.AlarmID}}	告警策略 ID	notice-3abd7ad6-15b7-4168-xxxx-52e5b961a561	是	-
{{.Topic}}	日志主题名称	nginxLog	是	-
{{.TopicId}}	日志主题 ID	a54de372-ffe0-49ae-xxxx-c340bb2b03f2	是	-
{{.QueryParams}}	执行语句查询参数	[{ "TopicId": "36c6df96-a6ad-xxxx-99ec-3f0b66dea302", "TopicName": "CDN Demo 访问日志日志主题_10000753xxxx", "TopicType": "log", "Query": "* | select approx_percentile(request_time, 0.99) as p99", "grammarVersion": "cql", "StartTime": 1690956330000, "EndTime": 1690957230000, "QueryUrlPath": "/cls/search?grammarVersion=cql&queryBase64=KiB8IHNlbGVjdCBhcHByb3hfcGVyY2VudGlsZShyZXF1ZXN0X3RpbWUsIDAuOTkpIGFzIHA5OQ%3D%3D&region=ap-chongqing&time=2023-08-02T14%3A05%3A30.000%2C2023-08-02T14%3A20%3A30.000&topic_id=36c6df96-a6ad-xxxx-99ec-3f0b66dea302" }]	是	数组结构，{{.QueryParams[0]}}代表第1个执行语句，{{.QueryParams[1]}}代表第2个执行语句，以此类推。
{{.Condition}}	触发条件	$1.errorLogCount > 1	是	-
{{.ConditionGroup}}	告警分组触发时对应的分组信息	{"$1.AppName":"userManageService"}	是	告警策略启用分组触发功能时才具备该变量。
{{.Level}}	告警级别(英文)	Info	是	• Critical • Warn • Info
{{.Level_zh}}	告警级别(中文)	提醒	否	• 紧急 • 警告 • 提醒
{{.HappenThreshold}}	告警所需的触发条件持续满足次数	1	是	-

{{.AlertThreshold}}	告警间隔时间	15	是	单位：分钟
{{.Label}}	附加分类字段	{"service":"payService"}	是	
{{.StartTime}}	第一次告警触发时间	2022-05-28 18:56:37	是	时区：Asia/Shanghai
{{.StartTimeUnix}}	第一次告警触发时间戳	1653735397099	是	毫秒级 UNIX 时间戳
{{.NotifyTime}}	本次告警通知时间	2022-05-28 19:41:37	是	时区：Asia/Shanghai
{{.NotifyTimeUnix}}	本次告警通知时间戳	1653738097099	是	毫秒级 UNIX 时间戳
{{.NotifyType}}	告警通知类型	1	是	• 1：代表告警通知 • 2：代表恢复通知
{{.ConsecutiveAlertNums}}	连续告警次数	2	是	-
{{.Duration}}	告警持续时间	0	是	单位：分钟
{{.DetailUrl}}	告警详情页面链接	https://alarm.cls.tencentcs.com/MDv2xxJh	是	无需登录账号
{{.QueryUrl}}	第一个执行语句的检索分析链接	https://alarm.cls.tencentcs.com/T0pkxxMA	否	如需获取其它执行语句链接地址，请使用 `{{.QueryParams}}` 。

{{.SilentUrl}}	告警屏蔽链接	https://alarm.cls.tencentcs.com/K6hyxxyG	是	无需登录账号，需{{.CanSilent}}为 true 时方可使用，例如：
				`{{if .CanSilent}}`
				`{{.SilentUrl}}{{end}}`
				。

{{.CanSilent}}	是否支持通过{{.SilentUrl}}直接屏蔽告警	true	是	在通知渠道组中开启**免登录屏蔽告警**功能后该值为 true，否则为 false。
{{.Message}}	附加通知内容	-	是	特指告警策略配置中填写的“附加通知内容”，最大10KB，超出后将自动截断。
{{.TriggerParams}}	告警触发时参数	$1.errorLogCount=5;	是	执行语句有多条结果同时满足触发条件时，仅记录其中的第一条结果。 仅告警触发时具备该变量，告警恢复时无该变量。
{{.QueryResult}}	执行语句执行结果	详见 {{.QueryResult}} 说明	是	-
{{.TriggerResult}}	满足触发条件的执行语句结果	详见 {{.TriggerResult}} 说明	是	仅告警触发时具备该变量，告警恢复时无该变量。
{{.QueryLog}}	执行语句中检索条件匹配到的详细日志	详见 {{.QueryLog}} 说明	是	-
{{.AnalysisResult}}	多维分析结果	详见 {{.AnalysisResult}} 说明	是	仅告警触发时具备该变量，告警恢复时无该变量。
{{.AnalysisResultFormat}}	格式化后的多维分析结果（英文）	-	否	内容与{{.AnalysisResult}}一致，但将 json 格式化为文本形式，使用换行、表格等形式展示，提高内容易读性。
{{.AnalysisResultFormat_zh}}	格式化后的多维分析结果（中文）	-	否	
AlarmExecutionStatus	告警策略执行结果	TriggerNotice.SendSuccess	是	仅支持投递日志。

AlarmExecutionErrorDetail	告警策略执行异常详情	{"TriggerFailedReason":"Parse expression failed: Invalid token: '$'","notify_result_group":null}	是	仅支持投递日志。
Silent	当前告警通知是否被屏蔽	• 全部通知均被屏蔽：SilentAll • 部分通知被屏蔽：SilentPartial • 未被屏蔽：NotSilent	是	仅支持投递日志。

相关说明如下：

{{.QueryResult}}

- **含义**：执行语句执行结果（无论是否满足触发条件）。
- **说明**：数组结构，`{{.QueryResult[0]}}` 代表第1个执行语句的执行结果，`{{.QueryResult[1]}}` 代表第2个执行语句的执行结果，以此类推。每个执行语句最多1000条数据、最大10KB，最多10个执行语句，总计最大100KB，超出限制后将自动截断。
- **变量值示例**：
  假设告警策略存在两条执行语句：

```
第1个执行语句：status:>=400 | select count(*) as errorLogCount
第2个执行语句：status:>=400 | select count(*) as errorLogCount,request_uri group by request_uri order by count(*) desc
```

则变量值为：

```
[
  [{
      "errorLogCount": 7
  }],
  [{
      "errorLogCount": 3,
      "request_uri": "/apple-touch-icon-144x144.png"
  }, {
      "errorLogCount": 3,
      "request_uri": "/feed"
  }, {
      "errorLogCount": 1,
      "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html"
  }]
]
```

{{.TriggerResult}}

- **含义**：满足触发条件的执行语句执行结果。
- **说明**：数组结构，`{{.TriggerResult[0]}}` 代表第1个执行语句的执行结果，`{{.TriggerResult[1]}}` 代表第2个执行语句的执行结果，以此类推。每个执行语句最多1000条数据、最大10KB，最多10个执行语句，总计最大100KB，超出限制后将自动截断。
- **变量值示例**：
  在上文的 {{.QueryResult}} 变量值示例的基础上，若触发条件为 `$1.errorLogCount > 1` 。则变量值为：

```
[
  [{
      "errorLogCount": 7
  }],
  [{
      "errorLogCount": 3,
      "request_uri": "/apple-touch-icon-144x144.png"
  }, {
      "errorLogCount": 3,
```

```
        "request_uri": "/feed"
    }]
]
```

{{.QueryLog}}

- **含义：**执行语句中检索条件匹配到的详细日志（不包含 SQL 中的过滤条件）。
- **说明：**数组结构，`{{.QueryLog[0]}}` 代表第1个执行语句的详细日志，`{{.QueryLog[1]}}` 代表第2个执行语句的详细日志，以此类推。每个执行语句最多10条数据、最大50KB，最多10个执行语句，总计最大150KB，超出限制后将自动截断。
- **变量值示例：**

```
[
  [{
      "content": {
          "__TAG__": {
              "pod": "nginxPod",
              "cluster": "testCluster"
          },
          "body_bytes_sent": "32847",
          "http_referer": "-",
          "http_user_agent": "Opera/9.80 (Windows NT 6.1; U; en-US) Presto/2.7.62 Version/11.01",
          "remote_addr": "105.86.148.186",
          "remote_user": "-",
          "request_method": "GET",
          "request_uri": "/apple-touch-icon-144x144.png",
          "status": "404",
          "time_local": "01/Nov/2018:00:55:14"
      },
      "fileName": "/root/testLog/nginx.log",
      "pkg_id": "285A243662909DE3-5CD",
      "source": "172.17.0.2",
      "time": 1653739000013,
      "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
  }, {
      "content": {
          "__TAG__": {
              "pod": "nginxPod",
              "cluster": "testCluster"
          },
          "body_bytes_sent": "33496",
          "http_referer": "-",
          "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/27.0.1453.93 Safari/537.36",
          "remote_addr": "222.18.168.242",
          "remote_user": "-",
          "request_method": "GET",
          "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html",
          "status": "404",
          "time_local": "01/Nov/2018:00:54:37"
      },
      "fileName": "/root/testLog/nginx.log",
      "pkg_id": "285A243662909DE3-5C8",
      "source": "172.17.0.2",
      "time": 1653738975008,
      "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
  }]
]
```

{{.AnalysisResult}}

- **含义：**多维分析结果。
- **说明：**对象结构，第一层级对象分别对应每个多维分析的结果。一级对象的 key 为多维分析的名称，value 为该多维分析的结果。仅告警触发（即{{.NotifyType}}=1）时具备该变量，告警恢复（即{{.NotifyType}}=2）时无该变量。每个多维分析最多10KB，最多5个多维分析，总计最大50KB。超出限制后将自动截断
  - 相关原始日志：每个语句最多10条数据，以用户配置为准。
  - 字段 Top5及占比统计：每个语句最多5条数据
  - 自定义检索分析（SQL）：每个语句最多1000条数据
  - 自定义检索分析（检索）：每个语句最多10条数据
- **变量值示例：**

  假设告警策略存在3个多维分析：

```
名称：Top URL
类型：字段TOP5及占比统计
字段：request_uri

名称：错误日志URL分布
类型：自定义检索分析
分析语句：status:>=400 | select count(*) as errorLogCount,request_uri group by request_uri order by count(*) desc

名称：详细错误日志
类型：自定义检索分析
分析语句：status:>=400
```

  则变量值为：


```
{
  "Top URL": [{
      "count": 77,
      "ratio": 0.45294117647058824,
      "value": "/"
  }, {
      "count": 20,
      "ratio": 0.11764705882352941,
      "value": "/favicon.ico"
  }, {
      "count": 7,
      "ratio": 0.041176470588235294,
      "value": "/blog/feed"
  }, {
      "count": 5,
      "ratio": 0.029411764705882353,
      "value": "/test-tile-service"
  }, {
      "count": 3,
      "ratio": 0.01764705882352941,
      "value": "/android-chrome-192x192.png"
  }],
  "详细错误日志": [{
      "content": {
          "__TAG__": {
              "pod": "nginxPod",
              "cluster": "testCluster"
          },
          "body_bytes_sent": "32847",
          "http_referer": "-",
          "http_user_agent": "Opera/9.80 (Windows NT 6.1; U; en-US) Presto/2.7.62 Version/11.01",
```

```
            "remote_addr": "105.86.148.186",
            "remote_user": "-",
            "request_method": "GET",
            "request_uri": "/apple-touch-icon-144x144.png",
            "status": "404",
            "time_local": "01/Nov/2018:00:55:14"
        },
        "fileName": "/root/testLog/nginx.log",
        "pkg_id": "285A243662909DE3-5CD",
        "source": "172.17.0.2",
        "time": 1653739000013,
        "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
    }, {
        "content": {
            "__TAG__": {
                "pod": "nginxPod",
                "cluster": "testCluster"
            },
            "body_bytes_sent": "33496",
            "http_referer": "-",
            "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/27.0.1453.93 Safari/537.36",
            "remote_addr": "222.18.168.242",
            "remote_user": "-",
            "request_method": "GET",
            "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html",
            "status": "404",
            "time_local": "01/Nov/2018:00:54:37"
        },
        "fileName": "/root/testLog/nginx.log",
        "pkg_id": "285A243662909DE3-5C8",
        "source": "172.17.0.2",
        "time": 1653738975008,
        "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
    }],
    "错误日志URL分布": [{
        "errorLogCount": 3,
        "request_uri": "/apple-touch-icon-144x144.png"
    }, {
        "errorLogCount": 3,
        "request_uri": "/feed"
    }, {
        "errorLogCount": 1,
        "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html"
    }]
}
```


变量语法

变量语法类似 Go Template 语法，可对告警通知变量进行提取和格式化处理，以更加清晰地呈现在告警通知内容中。所有变量及变量语法均位于 `{{ }}` 中，外部的文本不会进行处理。

变量提取

语法格式

```
{{.variable[x]}} 或 {{index .variable x}}
{{.variable.childNodeName}} 或 {{index .variable "childNodeName"}}
```

语法说明

- 变量为数组时，使用 `{{.variable[x]}}` 按数组下标提取对应的数组元素，其中 x 为大于等于0的整数，等价于 `{{index .variable x}}`。
- 变量为对象时，使用 `{{.variable.childNodeKey}}` 按子级对象名称（key）提取对应的子级对象值（value），等价于 `{{index .variable "childNodeName"}}`。

> **注意：**
> - 子级对象名称包含空格、`$`、`-`等特殊符号或为纯数字时，请使用 `{{index .variable "childNodeName"}}` 形式的语法，例如 `{{index .AnalysisResult "Top URL"}}`、`{{index .ConditionGroup "$1.AppName"}}`、`{{index .variable "123"}}`。
> - 使用数组下标提取元素时，请勿超出数组长度。必要时可使用 len 函数通过 条件判断 语法确定是否超出数组长度。

使用示例

`{{.QueryResult}}` 变量值为：

```
[
    [{
        "errorLogCount": 7 //提取该值
    }],
    [{
        "errorLogCount": 3,
        "request_uri": "/apple-touch-icon-144x144.png"
    }, {
        "errorLogCount": 3,
        "request_uri": "/feed"
    }, {
        "errorLogCount": 1,
        "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html"
    }]
]
```

通过以下表达式获取第一个数组中的 errorLogCount 的值：

```
{{.QueryResult[0][0].errorLogCount}}
```

返回结果：

```
7
```

循环遍历

语法格式

```
{{range .variable}}
自定义内容{{.childNode1}}自定义内容{{.childNode2}}...
{{end}}
```

或

```
{{range $key,$value := .variable}}
自定义内容{{$key}}自定义内容{{$value}}...
{{end}}
```

语法说明

变量为数组或包含多个子级对象的对象时，可使用该语法将其中的每个元素/对象按指定格式展示出来。

> **注意：**

> 在 `range` 语法内部，默认获取的是 `.variable` 内部的元素/对象，不能直接使用 `{{.UIN}}` 的形式获取外部变量，如需获取请在 `range` 语法外部提前创建独立的变量，例如：
>
> ```
> {{$UIN := .UIN}}  //创建变量
> {{range .variable}}
> 自定义内容{{.childNode1}}自定义内容{{.childNode2}}...
> {{$UIN}}  //使用变量
> {{end}}
> ```

使用示例

`{{.QueryResult}}` 变量值为：

```
[
    [{
        "errorLogCount": 7
    }],
    [{
        "errorLogCount": 3,
        "request_uri": "/apple-touch-icon-144x144.png"
    }, {
        "errorLogCount": 3,
        "request_uri": "/feed"
    }, {
        "errorLogCount": 1,
        "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html"
    }]
]
```

通过以下表达式分别展示第二个数组中每个 request_uri 对应的 errorLogCount：

```
{{range .QueryResult[1]}}
* {{.request_uri}}错误日志数：{{.errorLogCount}}
{{end}}
```

返回结果：

```
* /apple-touch-icon-144x144.png错误日志数：3

* /feed错误日志数：3

* /opt/node_apps/test-v5/app/themes/basic/public/static/404.html错误日志数：1
```

条件判断

语法格式

```
{{if boolen}}
xxx
{{end}}
```

或

```
{{if boolen}}
xxx
{{else}}
```

```
xxx
{{end}}
```

或

```
{{if boolen}}
xxx
{{else if boolen}}
xxx
{{end}}
```

语法说明

根据条件判断结果分别执行对应的表达式，条件判断支持使用 and or not 进行逻辑运算，支持比较大小值。

```
eq arg1 arg2 :  arg1 == arg2时为true
ne arg1 arg2 :  arg1 != arg2时为true
lt arg1 arg2 :  arg1 < arg2时为true
le arg1 arg2 :  arg1 <= arg2时为true
gt arg1 arg2 :  arg1 > arg2时为true
ge arg1 arg2 :  arg1 >= arg2时为true
```

使用示例

`{{.QueryResult}}` 变量值为：

```
[
    [{
        "errorLogCount": 7
    }],
    [{
        "errorLogCount": 3,
        "request_uri": "/apple-touch-icon-144x144.png"
    }, {
        "errorLogCount": 3,
        "request_uri": "/feed"
    }, {
        "errorLogCount": 1,
        "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html"
    }]
]
```

通过以下表达式分别展示第二个数组中 errorLogCount 大于等于2且小于等于100的 request_uri 及对应的 errorLogCount：

```
{{range .QueryResult[1]}}
{{if and (ge .errorLogCount 2) (le .errorLogCount 100)}}
* {{.request_uri}}错误日志数：{{.errorLogCount}}
{{end}}
{{end}}
```

返回结果：

```
* /apple-touch-icon-144x144.png错误日志数：3
```

```
* /feed错误日志数：3
```

使用 if 还可以判断字段值是否存在，字段值为空白字符串或不存在时，相当于 false。例如：

```
{{if .QueryLog[0][0].apple}}
apple exist, value is : {{.QueryLog[0][0].apple}}
{{else}}
apple is not exist
{{end}}
```

移除空白符号

语法格式

```
{{- xxx}} 或 {{xxx -}}
```

语法说明

变量语法在执行过程中会保留语法本身带来的空格、缩进和换行等空白符号，例如循环迭代和条件判断使用示例中的返回结果中包含较多空白行，影响展示效果，此时可在 `{{ }}` 的头部或尾部使用 `-` 移除其前面或后面的空白符号。

使用示例

沿用条件判断中的示例，表达式修改为：

```
{{- range .QueryResult[1]}}
{{- if and (ge .errorLogCount 2) (le .errorLogCount 100)}}
* {{.request_uri}}错误日志数：{{.errorLogCount}}
{{- end}}
{{- end}}
```

返回结果：

```
* /apple-touch-icon-144x144.png错误日志数：3
* /feed错误日志数：3
```

变量函数

字符串函数

函数		说明	示例
转义特殊字符	{{escape .variable}}	将 " 等特殊符号进行转义，以便于将变量放在 json 中，保持 json 格式正确。	变量值：`{{.ExecuteQuery[0]}}` 的变量值为 `status:>=400 | select count(*) as "错误日志数"` 函数示例：`{{escape .ExecuteQuery[0]}}` 返回结果：`status:>=400 | select count(*) as \"错误日志数\"`
	{{escape_markdown .variable}}	将 * # 等特殊符号进行转义，以便于将变量放在 markdown 中，保持 markdown 格式正确。 适用于**企业微信**机器人通知渠道。	
	{{escape_markdown_html .variable}}	将 * # 等特殊符号进行转义，以便于将变量放在 markdown 中，保持 markdown 格式正确。 适用于**钉钉及飞书**机器人通知渠道。	
截取字符串	{{substr .variable start}}	按指定起始位置和长度（可选）对字符串进行截取，起始位置及长度均按照字节数计算，一般英文字母及数字占1个字节，中文占3 - 4个字节。	变量值：`{{.QueryLog[0][0].fileName}}` 的变量值为 `/root/testLog/nginx.log` 函数示例：`{{substr .QueryLog[0][0].fileName 6 7 }}` 返回结果：
	{{substr .variable start length}}		

			testLog

	{{extract .variable startStringRegex}}	按指定起始字符和结束字符（可选）对字符串进行截取，起始字符和结束字符通过正则表达式进行指定。	变量值：
			`{{.QueryLog[0][0].fileName}}` 的变量值为 `/root/testLog/nginx.log`
			函数示例：
			`{{extract .QueryLog[0][0].fileName "/root/" "/nginx"}}`
			返回结果：
			`testLog`

	{{extract .variable startStringRegex endStringRegex}}		

	{{trim .variable}}	删除原始字符串开头和末尾的空字符。	函数示例：
			`{{trim "  Hello, World!  "}}`
			返回结果：
			`Hello, World!`

	{{trimPrefix prefix .variable}}	删除原始字符串指定前缀。	函数示例：
			`{{trimPrefix "hello" "hello, world!"}}`
			返回结果：
			`, world!`

	{{trimSuffix suffix .variable}}	删除原始字符串指定后缀。	函数示例：
			`{{trimSuffix "world!" "hello, world!"}}`
			返回结果：
			`hello,`

大小写转换	{{lower .variable}}	将原始字符串中所有字符转换为小写形式。	函数示例：
			`{{lower "HELLO WORLD"}}`
			返回结果：
			`hello world`

	{{upper .variable}}	将原始字符串中所有字符转换为大写形式。	函数示例：
			`{{upper "hello world"}}`
			返回结果：
			`HELLO WORLD`

	{{title .variable}}	将原始字符串进行标题化处理，即每个单词的首字母大写，其余字母小写。	函数示例：
			`{{title "hello world"}}`
			返回结果：
			`Hello World`

字符串替换	{{replace(oldstr,newstr,originstr)}}	将指定字符串 `originstr` 中的某个子字符串 `oldstr` 替换为另一个子字符串 `newstr` 。	函数示例：
			`{{replace "foo" "bar" "foo bar foo"}}`
			返回结果：
			`bar bar bar`

	{{regexReplaceAll(regex, originstr,newstr)}}	将指定字符串 `originstr` 中匹配正则表达式 `regex` 的字符串替换为指定字符串 `newstr` 。	函数示例：
			`{{regexReplaceAll "h" "hello world" "H"}}`
			返回结果：
			`Hello world`

字符串匹配	{{containstr .variable searchstring}}	变量值中是否包含指定的字符串，判断结果可用于条件判断语法中。	变量值：
			`{{.QueryLog[0][0].fileName}}` 的变量值为 `/root/testLog/nginx.log`
			函数示例：
			`{{if containstr .QueryLog[0][0].fileName "test"}}`测试日志`{{else}}`非测试日志`{{end}}`
			返回结果：
			测试日志

	{{regexMatch(regex,string)}	判断指定字符串和正则表达式是否匹配。匹配则返回 true，否则返回 false。	函数示例：
			`{{regexMatch "^hello" "hello world"}}`
			返回结果：
			`true`

字符串拼接	{{concat .variable1 .variable2 ...}}	将指定的多个变量或字符串拼接在一起。	函数示例：
			`{{concat .Region .Alarm}}`
			返回结果：
			`广州Nginx 错误日志过多`

字符串分割	{{splitList delimiter string}}	将字符串 `string` 按照分隔符 `delimiter` 切分为数组。	函数示例：
			`{{splitList "," "apple,banana,orange"}}`
			返回结果：
			`["apple" "banana" "orange"]`

字符串排序	{{sortAlpha stringList}}	将字符串列表按字母顺序。
		它不会原地排序，而是返回一个排序后的列表副本，符合列表不可变性的原则。
			函数示例：
			`{{sortAlpha (splitList "," "banana,apple,orange")}}`
			返回结果：
			`["apple","banana","orange"]`

统计长度或元素个数	{{len .variable}}	获取数组、map 数据结构的元素个数，也可获取字符串的长度，不支持获取数值的长度。	变量值：
			`{{.QueryResult}}` 变量值为 `[[{"number":19778,"text":"hello"}],[]]`
			函数示例：
			`{{len .QueryResult}}`
			`{{len .QueryResult[0]}}`
			`{{len .QueryResult[0][0]}}`
			`{{len .QueryResult[0][0].text}}`
			返回结果：
			`2`
			`1`
			`2`
			`5`

数学计算函数

函数		说明	示例

加法	{{add .variable1 .variable2 ...}}	整数(int64)加法运算	函数示例：
			`{{ add 5 3 }}`
			返回结果：
			`8`

	{{addf .variable1 .variable2 ...}}	浮点数(float64)加法运算	

减法	{{sub .variable1 .variable2}}	整数(int64)减法运算	函数示例：
			`{{ sub 10 3 }}`
			返回结果：
			`7`

	{{subf .variable1 .variable2}}	浮点数(float64)减法运算	
除法	{{div .variable1 .variable2}}	整数(int64)除法运算	函数示例： `{{ div 10 2 }}` 返回结果：
	{{divf .variable1 .variable2}}	浮点数(float64)除法运算	

取模	{{mod .variable1 .variable2}}	整数(int64)取模运算	函数示例：
			`{{ mod 10 3 }}`
			返回结果：
			`1`

乘法	{{mul .variable1 .variable2}}	整数(int64)乘法运算	函数示例：
			`{{ mul 4 5 }}`
			返回结果：
			`20`

	{{mulf .variable1 .variable2}}	浮点数(float64)乘法运算	

最小值	{{min .variable1 .variable2 ...}}	整数(int64)中获取最小值	函数示例：
			`{{ min 3 7 }}`
			返回结果：
			`3`

	{{minf .variable1 .variable2 ...}}	浮点数(float64)中获取最小值	

最大值	{{max .variable1 .variable2 ...}}	整数(int64)中获取最大值	函数示例：
			`{{ max 3 7 }}`
			返回结果：
			`7`

	{{maxf .variable1 .variable2 ...}}	浮点数(float64)中获取最大值	

取整	{{ceil .variable}}	向上取整	函数示例：
			`{{ ceil 5.3 }}`
			返回结果：
			`6`

	{{floor .variable}}	向下取整	

四舍五入	{{round .variable decimal}}	对数值 `variable` 进行四舍五入，保留指定小数位 `decimal`	函数示例：
			`{{ round 5.6748 2 }}`
			返回结果：
			`5.67`

类型转换函数

函数	说明	示例

{{atoi .variable}}	将字符串转换为整数	函数示例：
		`{{ atoi "123" }}`
		`{{ float64 12 }}`
		`{{ int "42" }}`
		`{{ int64 1234 }}`
		`{{ toString 123 }}`
		返回结果：
		`123`
		`12`
		`42`
		`1234`
		`123`

{{float64 .variable}}	转换为 float64类型	
{{int .variable}}	转换为与系统宽度匹配的 int 类型	
{{int64 .variable}}	转换为 int64 类型	
{{toString .variable}}	转换为字符串	

时间函数

函数		说明	示例
UNIX 时间戳转换	{{fromUnixTime .variable}}	将 UNIX 时间戳（支持毫秒及秒级时间戳）转换为可读的日期时间。其中时区可选，默认为 Asia/Shanghai。	变量值： `{{.QueryLog[0][0].time}}` 的变量值为 `1653893435008` 函数示例：

			{{fromUnixTime .QueryLog[0][0].time}} {{fromUnixTime .QueryLog[0][0].time "Asia/Shanghai"}} {{fromUnixTime .QueryLog[0][0].time "Asia/Tokyo"}} 返回结果： 2022-05-30 14:50:35.008 +0800 CST 2022-05-30 14:50:35.008 +0800 CST 2022-05-30 15:50:35.008 +0900 JST
	{{fromUnixTime .variable "timezone"}}		
时间格式化	{{date "2006-01-02 15:04:05" .variable}}	将指定时间戳转换为指定格式的字符串，第一个参数表示时间格式，以 `2006-01-02 15:04:05` 为基础，可调整格式。	函数示例： {{date "2006-01-02" (div .NotifyTimeUnix 1000)}} {{dateInZone "2006-01-02 15:04:05" (div .NotifyTimeUnix 1000) "Asia/Shanghai"}} 返回结果： 2025-06-10 2025-06-10 21:55:04 说明： .NotifyTimeUnix 为毫秒时间戳，使用 div 除以1000 后转换为秒时间戳
	{{dateInZone "2006-01-02 15:04:05" .variable timezone}}	与 date 函数功能类似，可指定时区。	

	{{duration .variable}}	将持续时间(秒)转换为人眼易读的格式。	函数示例：
			`{{duration (mul .Duration 60)}}`
			返回结果：
			`4h23m0s`
			说明：
			.Duration 为告警持续时间，单位分钟，使用 mul 乘以60后转换为秒

JSON格式化函数

函数	说明	示例

{{toPrettyJson .variable}}	将指定参数转换为格式化的JSON字符串，便于展示。	函数示例：
		`{{toPrettyJson .TriggerResult}}`
		返回结果：
		[
		[{
		"errorLogCount": 7

```
    }],
    [{
        "errorLogCount": 3,
        "request_uri": "/apple-touch-icon-
144x144.png"
    }, {
        "errorLogCount": 3,
        "request_uri": "/feed"
    }]
]
```


编解码/加密函数

函数	说明	示例

{{base64_encode .variable}}
{{base64_decode .variable}}
{{base64url_encode .variable}}
{{base64url_decode .variable}}
{{url_encode .variable}}
{{url_decode .variable}}
	对指定的变量或字符串进行编码（encode）或解码（decode），其中 base64url 在编解码过程中不会去除或补充尾部的"="。	函数示例：
		`{{base64_encode "test测试"}}`
		`{{base64_decode "dGVzdOa1i+ivlQ=="}}`
		`{{base64url_encode "test测试"}}`
		`{{base64url_decode "dGVzdOa1i-ivlQ=="}}`
		`{{url_encode "https://console.cloud.tencent.com:80/cls?region=ap-chongqing"}}`
		`{{url_decode "https%3A%2F%2Fconsole.cloud.tencent.com%3A80%2Fcls%3Fregion%3Dap-chongqing"}}`
		返回结果：
		`dGVzdOa1i+ivlQ==`
		`test测试`
		`dGVzdOa1i-ivlQ==`
		`test测试`
		`https%3A%2F%2Fconsole.cloud.tencent.com%3A80%2Fcls%3Fregion%3Dap-chongqing`
		`https://console.cloud.tencent.com:80/cls?region=ap-chongqing`

{{md5 .variable}}
{{md5 .variable | base64_encode}}
{{md5 .variable | base64url_encode}}
{{sha1 .variable}}
{{sha1 .variable | base64_encode}}
{{sha1 .variable | base64url_encode}}
{{sha256 .variable}}
{{sha256 .variable | base64_encode}}
{{sha256 .variable | base64url_encode}}
{{sha512 .variable}}
	对指定的变量或字符串按照特定的加密算法进行加密，默认返回结果编码格式为16进制字符串，可按需修改为 base64 或 base64url 编码格式。	函数示例：
		`{{md5 "test"}}`
		`{{md5 "test" | base64_encode}}`
		`{{md5 "test" | base64url_encode}}`
		`{{sha1 "test"}}`
		`{{sha1 "test" | base64_encode}}`
		`{{sha1 "test" | base64url_encode}}`
		`{{sha256 "test"}}`
		`{{sha256 "test" | base64_encode}}`
		`{{sha256 "test" | base64url_encode}}`
		`{{sha512 "test"}}`
		`{{sha512 "test" | base64_encode}}`
		`{{sha512 "test" | base64url_encode}}`

{{sha512 .variable | base64_encode}} {{sha512 .variable | base64url_encode}}		返回结果： 098F6BCD4621D373CADE4E832627B4F6 CY9rzUYh03PK3k6DJie09g== CY9rzUYh03PK3k6DJie09g== A94A8FE5CCB19BA61C4C0873D391E987982FBBD3 qUqP5cyxm6YcTAhz05Hph5gvu9M= qUqP5cyxm6YcTAhz05Hph5gvu9M= 9F86D081884C7D659A2FEAA0C55AD015A3BF4F1B2B0B822C D15D6C15B0F00A08 n4bQgYhMfWWaL+qgxVrQFaO/TxsrC4Is0V1sFbDwCgg= n4bQgYhMfWWaL-qgxVrQFaO_TxsrC4Is0V1sFbDwCgg= EE26B0DD4AF7E749AA1A8EE3C10AE9923F618980772E473F 8819A5D4940E0DB27AC185F8A0E1D5F84F88BC887FD67B14 3732C304CC5FA9AD8E6F57F50028A8FF 7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc/iBml1JQODbJ6wYX4 oOHV+E+IvIh/1nsUNzLDBMxfqa2Ob1f1ACio/w== 7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc_iBml1JQODbJ6wYX4 oOHV-E-IvIh_1nsUNzLDBMxfqa2Ob1f1ACio_w==
{{hmac_md5 .variable "Secretkey"}} {{hmac_md5 .variable "Secretkey" | base64_encode}} {{hmac_md5 .variable "Secretkey" | base64url_encode}} {{hmac_sha1 .variable "Secretkey"}} {{hmac_sha1 .variable "Secretkey" | base64_encode}} {{hmac_sha1 .variable "Secretkey" | base64url_encode}} {{hmac_sha256 .variable "Secretkey"}} {{hmac_sha256 .variable "Secretkey" | base64_encode}} {{hmac_sha256 .variable "Secretkey" | base64url_encode}} {{hmac_sha512 .variable "Secretkey"}} {{hmac_sha512 .variable "Secretkey" | base64_encode}} {{hmac_sha512 .variable "Secretkey" | base64url_encode}}	对指定的变量或字符串按照特定的加密算法进行加密，默认返回结果编码格式为16进制字符串，可按需修改为base64或base64url 编码格式。其中 Secretkey 为 hmac 加密算法中的密钥，可按需进行修改。	函数示例： { { hmac_md5 "test" "Secretkey" } } { { hmac_md5 "test" "Secretkey" | base64_encode } } { { hmac_md5 "test" "Secretkey" | base64url_encode } } { { hmac_sha1 "test" "Secretkey" } } { { hmac_sha1 "test" "Secretkey" | base64_encode } } { { hmac_sha1 "test" "Secretkey" | base64url_encode } } { { hmac_sha256 "test" "Secretkey" } } { { hmac_sha256 "test" "Secretkey" | base64_encode } } { { hmac_sha256 "test" "Secretkey" | base64url_encode } } { { hmac_sha512 "test" "Secretkey" } } { { hmac_sha512 "test" "Secretkey" | base64_encode } } { { hmac_sha512 "test" "Secretkey" | base64url_encode } } 返回结果： E7B946D930658699AA668601E33E87CE 57lG2TBlhpmqZoYB4z6Hzg== 57lG2TBlhpmqZoYB4z6Hzg== 2AB64F124D932F5033EAC7AF392AC5CC4D52F503 KrZPEk2TL1Az6sevOSrFzE1S9QM= KrZPEk2TL1Az6sevOSrFzE1S9QM= FC49EBC05209B1359773D87C216BA85BCE0163FDE459EA37 AB603EC9D8445D23 /EnrwFIJsTWXc9h8IWuoW84BY/3kWeo3q2A+ydhEXSM=

		_EnrwFIJsTWXc9h8IWuoW84BY_3kWeo3q2A-ydhEXSM= D18DF3D943F74769A8B66E43D7EF03639BB6B8B8A2EBC997 6170DC58EEE58BE98478F3183E4B5AA3481DE12026AAE384 3F8213B39D639EAC6EE93734EA667BC5 0Y3z2UP3R2motm5D1+8DY5u2uLii68mXYXDcWO7li+mEePMY Pktao0gd4SAmquOEP4ITs51jnqxu6Tc06mZ7xQ== 0Y3z2UP3R2motm5D1-8DY5u2uLii68mXYXDcWO7li- mEePMYPktao0gd4SAmquOEP4ITs51jnqxu6Tc06mZ7xQ==

参考案例

案例1：在告警通知中展示最近一条详细日志

需求场景

将符合执行语句检索条件的最近一条详细日志添加至告警通知中，以 key:value 的形式展示，每行一个 key，不包含 CLS 预置字段和元数据字段。

通知内容配置

```
{{range $key,$value := .QueryLog[0][0].content}}
{{if not (containstr $key "__TAG__")}}
{{- $key}}:{{$value}}
{{- end}}
{{- end}}
```

其中 `.QueryLog[0][0]` 代表符合告警策略第一条执行语句检索条件的最近一条详细日志，其值为：

```
{
    "content": {
        "__TAG__": {
            "a": "b12fgfe",
            "c": "fgerhcdhgj"
        },
        "body_bytes_sent": "33704",
        "http_referer": "-",
        "http_user_agent": "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.3319.102 Safari/537.36",
        "remote_addr": "247.0.249.191",
        "remote_user": "-",
        "request_method": "GET",
        "request_uri": "/products/hadoop)",
        "status": "404",
        "time_local": "01/Nov/2018:07:54:08"
    },
    "fileName": "/root/testLog/nginx.log",
    "pkg_id": "285A243662909DE3-210B",
    "source": "172.17.0.2",
    "time": 1653908859008,
    "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
}
```

告警通知内容

```
remote_addr:247.0.249.191
time_local:01/Nov/2018:07:54:08
http_user_agent:Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.3319.102
Safari/537.36
remote_user:-
```

```
http_referer:-
body_bytes_sent:33704
request_method:GET
request_uri:/products/hadoop)
status:404
```

案例2：在告警通知中展示多个执行语句对应的详细日志

需求场景

告警策略具备多个执行语句时，将每个语句中对应的详细日志添加至告警通知中。

通知内容配置

```
{{- range .QueryLog}}
{{- range .}}
{{.content}}
{{- end}}
{{- end}}
```

其中 `.QueryLog` 代表符合告警策略所有执行语句检索条件的详细日志（最多不超过10条），其值为：

```
{
    "QueryLog": [
        [{
            "content": {
                "__TAG__": {
                    "pod": "nginxPod",
                    "cluster": "testCluster"
                },
                "body_bytes_sent": "32847",
                "http_referer": "-",
                "http_user_agent": "Opera/9.80 (Windows NT 6.1; U; en-US) Presto/2.7.62 Version/11.01",
                "remote_addr": "105.86.148.186",
                "remote_user": "-",
                "request_method": "GET",
                "request_uri": "/apple-touch-icon-144x144.png",
                "status": "404",
                "time_local": "01/Nov/2018:00:55:14"
            },
            "fileName": "/root/testLog/nginx.log",
            "pkg_id": "285A243662909DE3-5CD",
            "source": "172.17.0.2",
            "time": 1653739000013,
            "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
        }, {
            "content": {
                "__TAG__": {
                    "pod": "nginxPod",
                    "cluster": "testCluster"
                },
                "body_bytes_sent": "33496",
                "http_referer": "-",
                "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/27.0.1453.93 Safari/537.36",
                "remote_addr": "222.18.168.242",
                "remote_user": "-",
                "request_method": "GET",
                "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html",
                "status": "404",
```

```
                "time_local": "01/Nov/2018:00:54:37"
            },
            "fileName": "/root/testLog/nginx.log",
            "pkg_id": "285A243662909DE3-5C8",
            "source": "172.17.0.2",
            "time": 1653738975008,
            "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
        }],
        [{
            "content": {
                "__TAG__": {
                    "pod": "nginxPod",
                    "cluster": "testCluster"
                },
                "body_bytes_sent": "32847",
                "http_referer": "-",
                "http_user_agent": "Opera/9.80 (Windows NT 6.1; U; en-US) Presto/2.7.62 Version/11.01",
                "remote_addr": "105.86.148.186",
                "remote_user": "-",
                "request_method": "GET",
                "request_uri": "/apple-touch-icon-144x144.png",
                "status": "404",
                "time_local": "01/Nov/2018:00:55:14"
            },
            "fileName": "/root/testLog/nginx.log",
            "pkg_id": "285A243662909DE3-5CD",
            "source": "172.17.0.2",
            "time": 1653739000013,
            "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
        }, {
            "content": {
                "__TAG__": {
                    "pod": "nginxPod",
                    "cluster": "testCluster"
                },
                "body_bytes_sent": "33496",
                "http_referer": "-",
                "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/27.0.1453.93 Safari/537.36",
                "remote_addr": "222.18.168.242",
                "remote_user": "-",
                "request_method": "GET",
                "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html",
                "status": "404",
                "time_local": "01/Nov/2018:00:54:37"
            },
            "fileName": "/root/testLog/nginx.log",
            "pkg_id": "285A243662909DE3-5C8",
            "source": "172.17.0.2",
            "time": 1653738975008,
            "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
        }]
    ]
}
```


告警通知内容


```
{"__TAG__":{"pod":"nginxPod","cluster":"testCluster"},"body_bytes_sent":"32847","http_referer":"-
","http_user_agent":"Opera/9.80 (Windows NT 6.1; U; en-US) Presto/2.7.62
Version/11.01","remote_addr":"105.86.148.186","remote_user":"-
```


版权所有：腾讯云计算（北京）有限责任公司

```
","request_method":"GET","request_uri":"/apple-touch-icon-
144x144.png","status":"404","time_local":"01/Nov/2018:00:55:14"},
{"__TAG__":{"pod":"nginxPod","cluster":"testCluster"},"body_bytes_sent":"33496","http_referer":"-
","http_user_agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/27.0.1453.93 Safari/537.36","remote_addr":"222.18.168.242","remote_user":"-
","request_method":"GET","request_uri":"/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html","status":"404","time_local":"01/Nov/2018:00:54:37"},
{"__TAG__":{"pod":"nginxPod","cluster":"testCluster"},"body_bytes_sent":"32847","http_referer":"-
","http_user_agent":"Opera/9.80 (Windows NT 6.1; U; en-US) Presto/2.7.62
Version/11.01","remote_addr":"105.86.148.186","remote_user":"-
","request_method":"GET","request_uri":"/apple-touch-icon-
144x144.png","status":"404","time_local":"01/Nov/2018:00:55:14"},
{"__TAG__":{"pod":"nginxPod","cluster":"testCluster"},"body_bytes_sent":"33496","http_referer":"-
","http_user_agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/27.0.1453.93 Safari/537.36","remote_addr":"222.18.168.242","remote_user":"-
","request_method":"GET","request_uri":"/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html","status":"404","time_local":"01/Nov/2018:00:54:37"}
```


案例3：在告警通知中展示执行语句执行结果

需求场景

将执行语句执行结果中符合触发条件的部分添加至告警通知中，以列表的形式展示。

告警策略执行语句为：

`status:>=400 | select count(*) as errorLogCount,request_uri group by request_uri order by count(*) desc`

触发条件为： `$1.errorLogCount > 10`

通知内容配置

```
{{range .QueryResult[0]}}
{{- if gt .errorLogCount 10}}
{{.request_uri}}错误日志数：{{.errorLogCount}}
{{- end}}
{{- end}}
```

其中 `.QueryResult[0]` **代表告警策略第一条执行语句的执行结果，其值为：**


```
[{
    "errorLogCount": 161,
    "request_uri": "/apple-touch-icon-144x144.png"
}, {
    "errorLogCount": 86,
    "request_uri": "/opt/node_apps/test-v5/app/themes/basic/public/static/404.html"
}, {
    "errorLogCount": 33,
    "request_uri": "/feed"
}, {
    "errorLogCount": 26,
    "request_uri": "/wp-login.php"
}, {
    "errorLogCount": 10,
    "request_uri": "/safari-pinned-tab.svg"
}, {
    "errorLogCount": 7,
    "request_uri": "/mstile-144x144.png"
}, {
    "errorLogCount": 4,
    "request_uri": "/atom.xml"
}, {
    "errorLogCount": 3,
```

```
    "request_uri": "/content/plugins/prettify-gc-syntax-highlighter/launch.js?ver=3.5.2?ver=3.5.2"
}]
```

告警通知内容

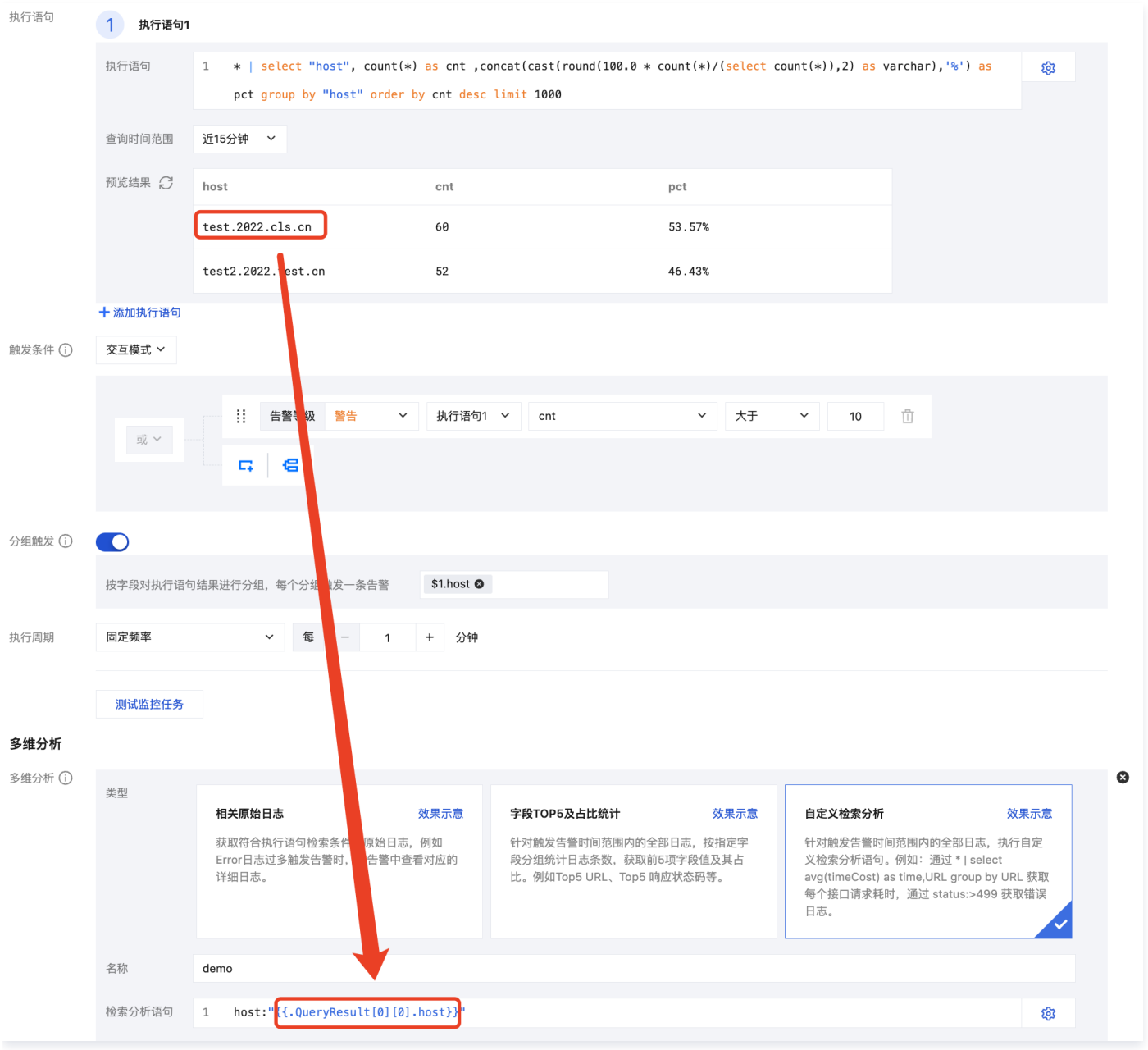
```
/apple-touch-icon-144x144.png错误日志数：161
/opt/node_apps/elastic-v5/app/themes/basic/public/static/404.html错误日志数：86
/feed错误日志数：33
/wp-login.php错误日志数：26
```

案例4：在多维分析中使用执行语句执行结果

需求场景

告警策略中配置多维分析时，需将执行语句的结果中的某个字段的值作为多维分析的过滤条件。

告警策略配置



其中 `{{.QueryResult[0][0].host}}` 代表第一条执行语句执行结果第一行中的 `host` 。

> **注意：**
> 变量值为字符串时，请使用引号包裹以符合检索分析语句规则。检索条件中使用双引号包裹，例如 `"{{.QueryResult[0][0].host}}"`，SQL 中需使用单引号包裹，例如 `'{{.QueryResult[0][0].host}}'` 。

管理集成配置

最近更新时间：2025-05-23 14:28:44

操作场景

通过集成配置可以将企业微信、钉钉及飞书机器人集成至日志服务 CLS，并使用这些机器人接收告警通知。也可以将自定义接口集成至日志服务 CLS，在触发告警时将自动调用该接口，实现与第三方系统的集成，例如触发告警时在工单系统自动创建工单。

操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**监控告警** > **集成配置**，进入集成配置管理页面。
3. 单击**新建**，并在“新建集成配置”中，填写如下信息：
   - 名称
   - 类型：支持企业微信、钉钉、飞书及自定义接口回调。
   - Webhook：企业微信、钉钉、飞书机器人地址或自定义接口地址。
     - 企业微信机器人配置方式：详情请查看 如何设置群机器人 。
     - 钉钉机器人配置方式：详情请查看 自定义机器人的创建和安装 。其中安全配置建议使用“加签”，如需使用“自定义关键词”，请使用 内容模板 中固定的关键词，不要使用变量。
     - 飞书机器人配置方式：详情请查看 如何在群组中使用机器人 。其中安全设置建议使用“签名校验”，如需使用“自定义关键词”，请使用 内容模板 中固定的关键词，不要使用变量。
   - 私钥：钉钉、飞书机器人可配置。
   - 请求方式：自定义接口回调可配置，支持 POST 及 PUT。自定义接口回调的请求信息（请求头及请求内容）需在 通知内容模板 中配置。
4. 单击**确定**，保存配置。
5. 在通知渠道组中选择需要使用的企业微信、钉钉、飞书或自定义接口回调，操作步骤详见 管理通知渠道组 。

版权所有：腾讯云计算（北京）有限责任公司

查看告警历史

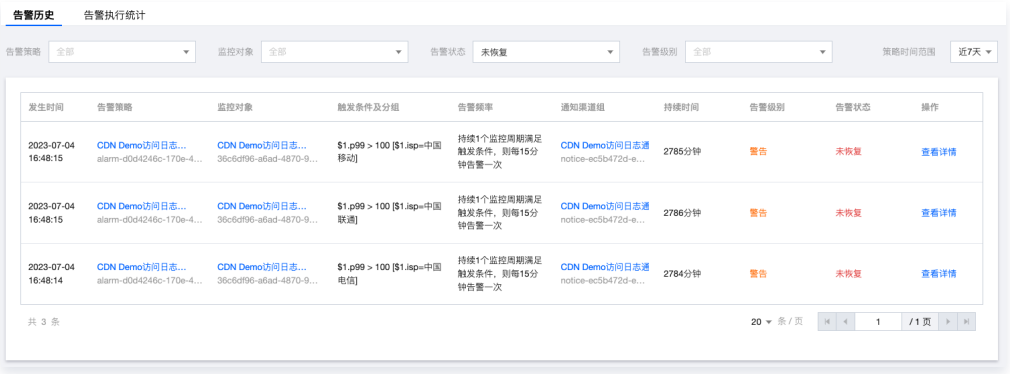
最近更新时间：2024-07-26 16:24:31

操作场景

日志服务提供近30天的告警历史信息查看，本文将为您介绍如何在日志服务控制台查看告警历史。

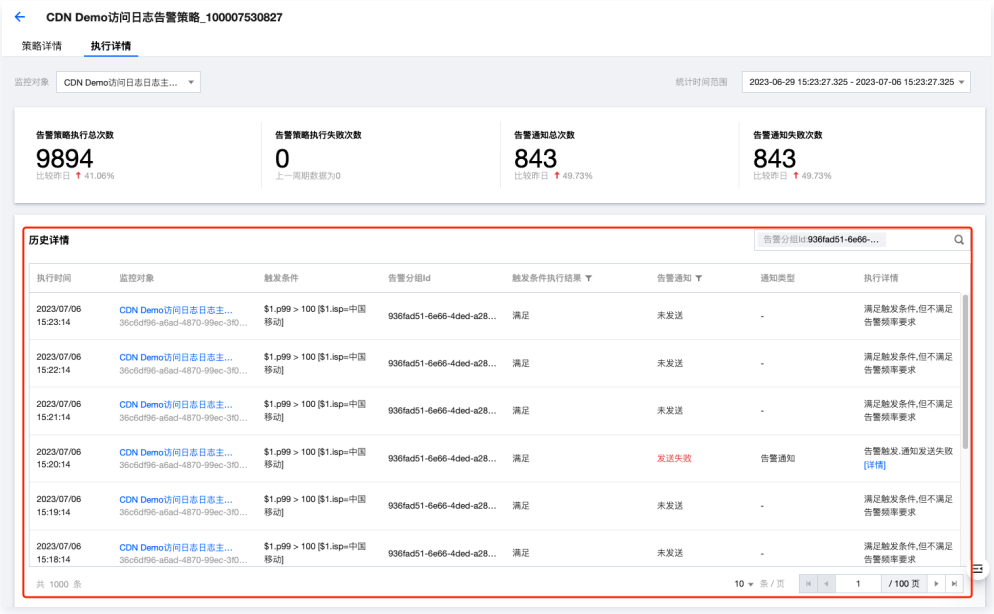
操作步骤

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**监控告警 > 告警历史**，进入告警历史查看页面。默认显示今天未恢复的告警，可在列表顶部下拉框中选择不同的告警状态、告警级别和时间范围。



新产生的告警处于“未恢复”状态，当监控指标不再满足触发条件时状态将变为“已恢复”，告警策略被删除或发生变化时，相关的告警状态将变为“已失效”。

3. 单击列表最右侧的**查看详情**，可查看本次告警的详细执行过程。



4. **执行详情**说明如下：

执行详情	AlarmExecutionStatus	说明
执行语句结果不满足触发条件	QueryResultUnmatch	-
满足触发条件,但不满足告警频率要求	AlertIntervalThresholdUnreach	可在 告警策略 中配置
满足触发条件,但不满足告警持续周期要求	HappenThresholdUnreach	可在 告警策略 中配置
告警触发.通知发送成功	MatchedTriggerNotice.SendSuccess	-

告警触发.通知被屏蔽	MatchedTriggerNotice.NotifySilenced	详见 告警屏蔽
告警触发.部分通知发送失败	MatchedTriggerNotice.SendPartFail	单击**详情**可查看详细失败原因
告警触发.通知发送失败	MatchedTriggerNotice.SendFail	单击**详情**可查看详细失败原因
告警触发.不符合通知渠道组通知时段要求	MatchedTriggerNotice.TimeUnmatched	可在 通知渠道组 中配置
告警触发.不符合通知渠道组类型要求	MatchedTriggerNotice.TypeUnmatched	可在 通知渠道组 中配置
告警触发.不符合任何通知规则	MatchedTriggerNotice.RuleUnmatched	可在 通知渠道组 中配置
告警恢复.通知发送成功	MatchedRecoverNotice.SendSuccess	–
告警恢复.通知被屏蔽	MatchedRecoverNotice.NotifySilenced	详见 告警屏蔽
告警恢复.部分通知发送失败	MatchedRecoverNotice.SendPartFail	单击**详情**可查看详细失败原因
告警恢复.通知发送失败	MatchedRecoverNotice.SendFail	单击**详情**可查看详细失败原因
告警恢复.不符合通知渠道组通知时段要求	MatchedRecoverNotice.TimeUnmatched	可在 通知渠道组 中配置
告警恢复.不符合通知渠道组类型要求	MatchedRecoverNotice.TypeUnmatched	可在 通知渠道组 中配置
告警恢复.不符合任何通知规则	MatchedRecoverNotice.RuleUnmatched	可在 通知渠道组 中配置
告警恢复（触发条件计算次数超出上限）	EvalTimesLimited	触发条件表达式 执行过程中，当语句返回多个结果时，会根据返回结果依次计算1000次，判断其是否满足触发条件。超过1000次仍未满足触发条件时，认为本次判断结果为假，告警恢复。
告警策略被修改，未恢复告警自动失效	AlarmConfigModified	告警策略被修改或删除时，未恢复的告警将会被设定为“已失效”状态
执行语句执行错误	QueryError	检查 告警策略 中的执行语句是否正确
触发条件表达式语法错误	ConditionSyntaxError	检查告警策略中的 触发条件表达式 是否正确
触发条件表达式执行错误(触发条件计算结果非布尔值)	ConditionValueTypeError	检查告警策略中的 触发条件表达式 是否正确
触发条件表达式执行错误	ConditionEvaluateError	检查告警策略中的 触发条件表达式 是否正确

告警屏蔽

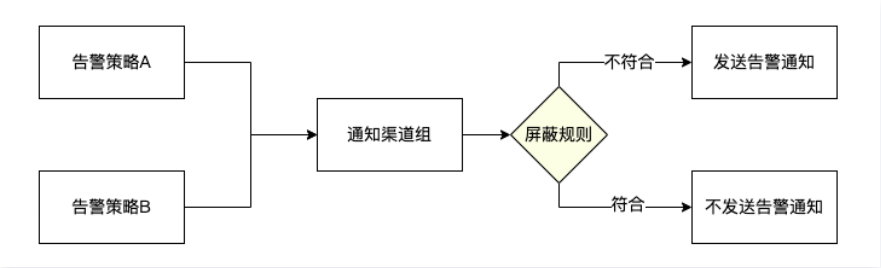
最近更新时间：2024-10-29 09:51:32

操作场景

系统出现异常时往往会持续产生告警，对于一些已经在处理的问题可以将告警屏蔽一段时间，避免被干扰。此外，计划性的系统变更也有可能产生一些预期内的告警，可提前屏蔽此类告警。
日志服务提供两种告警屏蔽方式：

- 临时性屏蔽告警：收到告警通知时，直接屏蔽当前告警一段时间（15分钟~12小时），操作便捷，适合临时屏蔽已经在处理的告警。
- 计划性屏蔽告警：通过控制台提前创建告警屏蔽规则，可批量操作，适合在系统变更前提前屏蔽部分预期内的告警。

告警屏蔽规则隶属于 管理通知渠道组，多个告警策略使用相同的通知渠道组时，可在该通知渠道组中批量屏蔽多个告警策略。



操作步骤

临时性屏蔽告警

> **注意：**
> - 暂不支持微信通知渠道。
> - 需在 管理通知渠道组 中开启**免登录操作告警**功能，允许在告警通知中直接屏蔽/认领相关告警，无需登录腾讯云控制台。

1. 短信、邮件、企业微信、钉钉等文字渠道：
   1.1 通过短信、邮件、企业微信、钉钉等渠道收到告警通知。
   1.2 在通知中点击**屏蔽告警**或**详细报告**（再在底部点击**屏蔽告警**），进入屏蔽告警配置页面，查看或填写如下信息：
      - 通知渠道组：当前告警策略使用的通知渠道组，可能为多个，不可修改。
      - 屏蔽时间范围：支持15分钟、30分钟、1小时、6小时、1天，单选。
      - 屏蔽规则：自动生成，仅屏蔽当前告警对应的告警策略、告警等级、监控对象及告警分组（需开启 分组触发 功能），不可修改。
      - 操作者：本次操作人员名称，必填。
      - 屏蔽原因：屏蔽该告警的原因，必填。
2. 电话渠道：
   2.1 接听电话。
   2.2 在接听过程中**按键2**屏蔽该告警30分钟。

计划性屏蔽告警

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**监控告警 > 通知渠道组**，进入通知渠道组管理页面。
3. 在列表中单击需要配置屏蔽规则的**通知渠道组名称/ID**，进入通知渠道组详情页面。
4. 在顶部单击**告警屏蔽规则**标签，进入告警屏蔽规则管理页面。
5. 单击**添加规则**，在弹出的对话框中填写如下信息：
   - 屏蔽时间范围：需要屏蔽哪段时间的告警。
   - 屏蔽规则：
      - 所有通知：屏蔽使用该通知渠道组的所有告警。
      - 同时满足以下规则的通知：仅屏蔽满足自定义规则的告警，例如屏蔽所有“提醒”级别的告警，支持的规则详见 规则详情 。
   - 屏蔽原因：屏蔽该告警的原因，必填。

查看已创建的屏蔽规则

无论哪种方式创建的屏蔽规则，均可在控制台中查看并管理。

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**监控告警 > 通知渠道组**，进入通知渠道组管理页面。
3. 在列表中单击需要配置屏蔽规则的**通知渠道组名称/ID**，进入通知渠道组详情页面。
4. 在顶部单击**告警屏蔽规则**标签，进入告警屏蔽规则管理页面。
5. 不同状态的规则支持如下操作：
   - 生效中：支持复制、过期，点击过期后该规则不再生效。
   - 已失效：支持复制、查看，不支持手动删除。系统最大保存1000条已失效的屏蔽规则，超出后自动删除旧规则。
   - 暂未生效：支持编辑、复制及删除。

规则详情

通过控制台或 API 创建屏蔽规则时，支持按照如下字段及比较方式定义屏蔽规则：

字段	比较方式	比较值	API 参数示例（Rule）
告警等级 `Level`	• 属于 `In` • 不属于 `NotIn`	• 提醒 `1` • 警告 `0` • 紧急 `2` （支持多个）	示例含义：告警等级不属于提醒 `{ "Type": "Condition", "Value": "Level", "Children": [{ "Value": "NotIn", "Type": "Compare" }, { "Value": "[1]", "Type": "Value" }] }`
告警策略 `AlarmID`	• 属于 `In` • 不属于 `NotIn`	告警策略（支持多个）	示例含义：告警策略属于 `alarm-4ddebe88-xxxx-4d8c-acce-6a8613e24cbf` 、`alarm-12a93b68-xxxx-4a42-bcf3-843690fc0793` `{ "Type": "Condition", "Value": "AlarmID", "Children": [{ "Value": "In", "Type": "Compare" }, { "Value": "[\"alarm-4ddebe88-xxxx-4d8c-acce-6a8613e24cbf\",\"alarm-12a93b68-xxxx-4a42-bcf3-843690fc0793\"]", "Type": "Value" }] }`
告警策略名称 `AlarmName`	• 正则匹配 `=~` • 正则不匹配 `!=~`	告警策略名称（正则表达式）	示例含义：告警策略名称正则不匹配 `test` `{ "Type": "Condition", "Value": "AlarmName", "Children": [{ "Value": "!=~",`

<table>
<tr><td></td><td></td><td></td><td>

```
        "Type": "Compare"
    }, {
        "Value": "test",
        "Type": "Value"
    }]
}
```

</td></tr>
<tr><td rowspan="2">告警分类字段 <code>Label</code>（需指定具体的分类字段名称）</td><td>

- 属于 <code>In</code>
- 不属于 <code>NotIn</code>

</td><td>分类字段值（支持多个）</td><td>

示例含义：告警分类字段 <code>key1</code> 属于 <code>value1</code> 、 <code>value2</code>

```
{
    "Type": "Condition",
    "Value": "Label",
    "Children": [{
        "Value": "key1",
        "Type": "Key"
    }, {
        "Value": "In",
        "Type": "Compare"
    }, {
        "Value": "[\"value1\",\"value2\"]",
        "Type": "Value"
    }]
}
```

</td></tr>
<tr><td>

- 正则匹配 <code>=~</code>
- 正则不匹配 <code>!=~</code>

</td><td>分类字段值（正则表达式）</td><td>

示例含义：告警分类字段 <code>key2</code> 正则匹配 <code>value3</code>

```
{
    "Type": "Condition",
    "Value": "Label",
    "Children": [{
        "Value": "key2",
        "Type": "Key"
    }, {
        "Value": "=~",
        "Type": "Compare"
    }, {
        "Value": "value3",
        "Type": "Value"
    }]
}
```

</td></tr>
<tr><td>分组触发字段 <code>Group</code>（需指定具体的分组字段名称）</td><td>

- 属于 <code>In</code>
- 不属于 <code>NotIn</code>

</td><td>分类字段值（支持多个）</td><td>

示例含义：分组触发字段 <code>$1.key1</code> 属于 <code>value1</code> 、 <code>value2</code>

```
{
    "Type": "Condition",
    "Value": "Group",
    "Children": [{
        "Value": "$1.key1",
        "Type": "Key"
    }, {
        "Value": "In",
        "Type": "Compare"
    }, {
        "Value": "[\"value1\",\"value2\"]",
        "Type": "Value"
    }]
```

</td></tr>
</table>

版权所有：腾讯云计算（北京）有限责任公司

			}
			示例含义：分组触发字段 `$1.key2` 正则不匹配 `value3`
			
			{
			"Type": "Condition",
			"Value": "Group",
			"Children": [{
			"Value": "$1.key2",
			"Type": "Key"
			}, {
			"Value": "!=~",
			"Type": "Compare"
			}, {
			"Value": "value3",
			"Type": "Value"
			}]
			}

	• 正则匹配 `=~` • 正则不匹配 `!=~`	分类字段值（正则表达式）	

监控对象 `Monitor Object`	• 属于 `In`
	• 不属于 `NotIn`
		日志主题及指标主题（支持多个）	示例含义：监控对象属于日志主题 `aa6b76f1-9040-47bc-xxxx-cdd67000c0ce` 、指标主题 `554588f7-3481-4c15-xxxx-80363b3a1c8e`
			说明：BizType=0代表日志主题；BizType=1代表指标主题
			
			{
			"Type": "Condition",
			"Value": "MonitorObject",
			"Children": [{
			"Value": "In",
			"Type": "Compare"
			}, {
			"Value": "[{\"TopicId\":\"aa6b76f1-9040-47bc-xxxx-cdd67000c0ce\",\"BizType\":0},{\"TopicId\":\"554588f7-3481-4c15-xxxx-80363b3a1c8e\",\"BizType\":1}]",
			"Type": "Value"
			}]
			}

例如以下屏蔽规则代表屏蔽 demo 告警策略中的提醒级别告警：



对应的 API 参数 Rule 为：

```
{
    "Value": "AND", //同时满足以下规则，必须为AND
    "Type": "Operation",
    "Children": [{
        "Type": "Condition", //第一条规则
        "Value": "Level", //告警等级
        "Children": [{
```

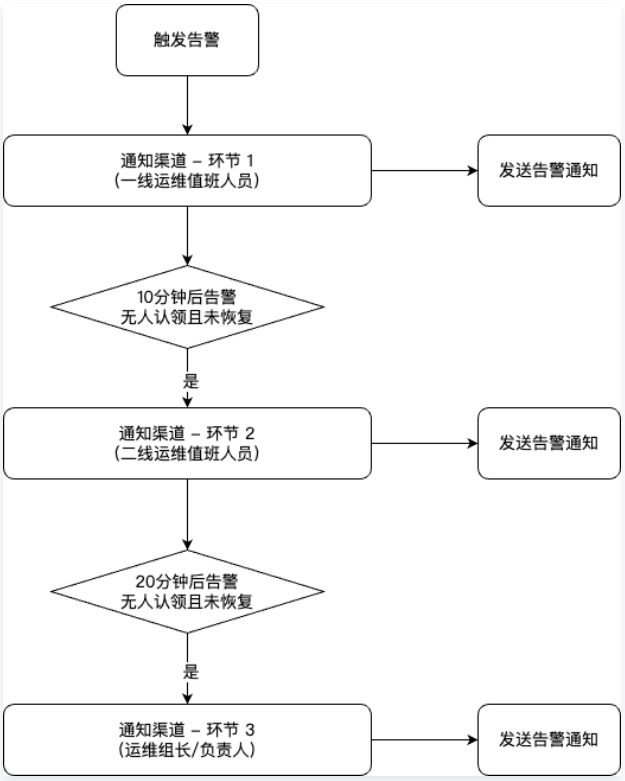
```
            "Value": "In", //属于
            "Type": "Compare"
        }, {
            "Value": "[1]", //提醒
            "Type": "Value"
        }]
    }, {
        "Type": "Condition", //第二条规则
        "Value": "AlarmID", //告警策略
        "Children": [{
            "Value": "In", //属于
            "Type": "Compare"
        }, {
            "Value": "[\"alarm-57105ec6-xxxx-xxxx-xxxx-892f3b8d143a\"]", //demo告警策略对应的ID
            "Type": "Value"
        }]
    }]
}
```

告警认领与升级

最近更新时间：2024-10-28 18:06:52

操作场景

系统出现异常时，接收到告警通知的人员可在告警详情中**认领**该告警，表示其已收到该通知并开始处理该告警，认领后的告警将不再重复发送通知。
认领还可以与告警升级结合使用，在 通知渠道组 中配置升级规则后，可按顺序依次为通知渠道组中的环节发送告警通知，当上一个环节超过指定时间仍未认领告警时再向下一个环节发送通知，例如下图。



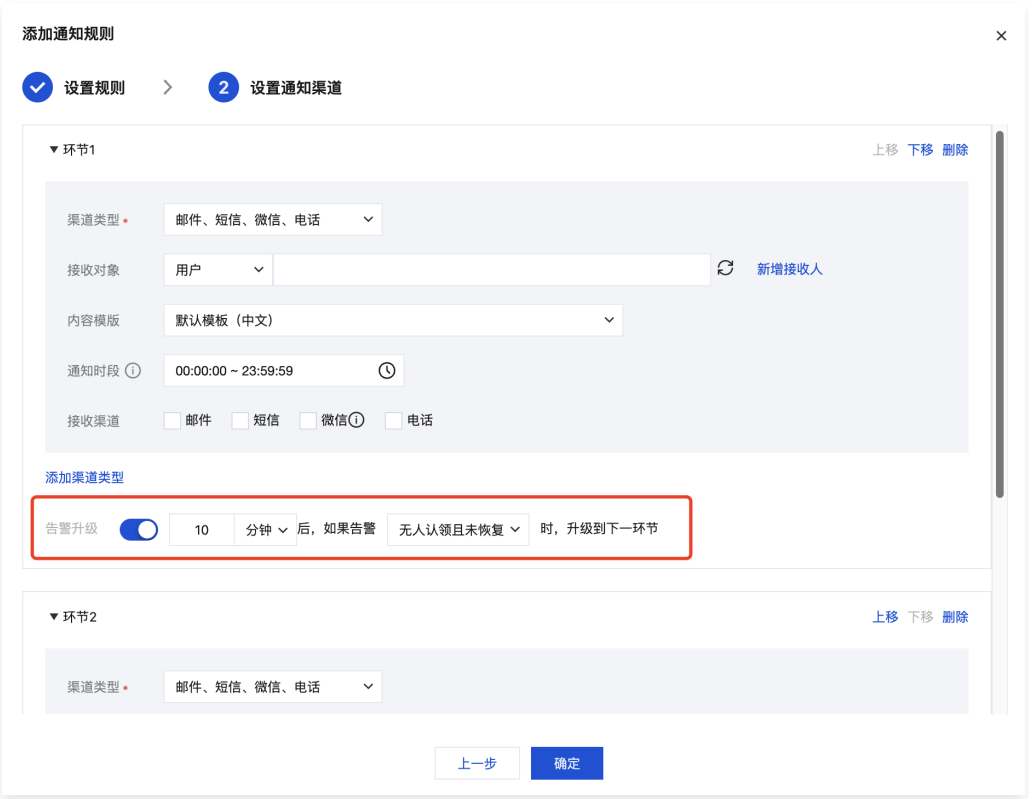
前提条件

- 在 通知渠道组 高级配置中，开启**免登录操作告警**开关，允许在告警通知中直接屏蔽/认领相关告警，无需登录腾讯云控制台。
- 暂不支持微信通知渠道。

操作步骤

配置告警升级规则

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**监控告警 > 通知渠道组**，进入通知渠道组管理页面。
3. 选择需要配置升级规则的通知渠道组，单击**编辑**。
4. 在**通知规则**中添加规则后，可在每个通知渠道环节的底部设置是否需要告警升级，最多可配置5个环节。



认领告警

- 短信、邮件、企业微信、钉钉等文字渠道：
  - 收到告警通知。
  - 在通知中点击**认领告警**或**详细报告**（再在底部点击**认领告警**），进入认领告警配置页面，填写操作人员名称。
- 电话渠道：
  - 接听电话。
  - 在接听过程中**按键1**认领该告警。

云产品中心
云产品中心概述

最近更新时间：2025-02-08 18:13:52

云产品中心是基于日志服务构建的统一云产品可观测平台，通过对云产品日志、指标、和使用情况等多维度数据的深度关联分析，提供全面的云产品用量分析、性能监控、安全审计、数据保护、异常检测和访问分析等服务。帮助您快速构建云产品的可观测性，提升云资源的使用效率和安全性，助力企业在数字化转型过程中实现更高的业务价值。

功能特性

- 开箱即用云产品运维分析仪表盘。
- 统一云产品日志检索分析入口。
- 批量开启云产品日志采集。

支持云产品

以下云产品支持通过云产品日志中心管理云产品日志，其他云产品接入请参见 采集云产品日志 。

- CLB 云产品中心
- CDN 云产品中心
- COS 云产品中心
- SCF 云产品中心
- TKE 云产品中心

版权所有：腾讯云计算（北京）有限责任公司

CLB 云产品中心

最近更新时间：2025-02-08 18:13:52

CLB 云产品中心提供负载均衡 CLB 访问日志的一站式采集与检索分析，同时基于访问日志，提供开箱即用的访问分析仪表盘。助力您快速构建 CLB 访问分析的可观测性。

功能特性

- 开箱即用 CLB 实例访问分析仪表盘。
- 统一 CLB 实例访问日志检索分析入口。
- 批量开启 CLB 实例访问日志采集。

前提条件

已在 CLB 控制台 创建负载均衡实例。

操作步骤

开启负载均衡访问日志采集

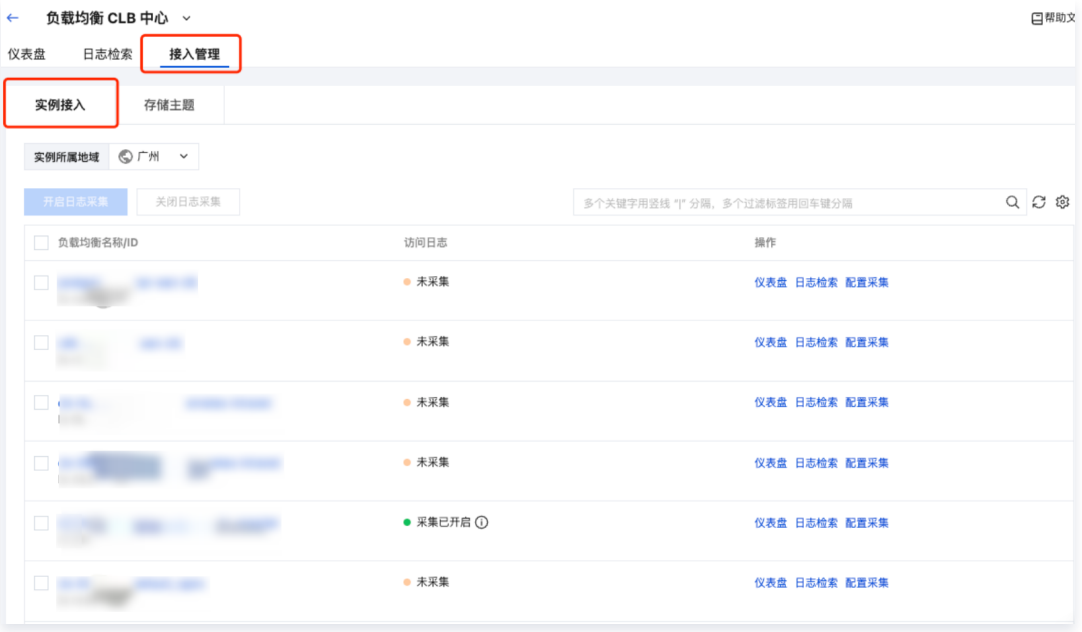
步骤1：进入 CLB 云产品中心

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**云产品中心**，进入云产品中心页面。
3. 在**云产品中心**中，选择并单击**负载均衡 CLB 中心**。



步骤2：开启负载均衡访问日志采集

1. 在**负载均衡 CLB 中心**中，选择**接入管理 > 实例接入**。



2. 选中目标负载均衡实例所在地域，在实例列表中，找到并勾选一个或多个目标负载均衡实例。



3. 单击**开启日志采集**。



4. 在开启日志采集弹窗中，选择或创建目标 日志主题，并单击**确认**，即可开启目标负载均衡实例的访问日志采集。

> **说明：**
> 仅支持自定义目标存储日志主题， 日志集默认为 CLB 负载均衡专用日志集，日志集名称为 clb_logset。



查看负载均衡访问分析仪表盘

1. 完成开启负载均衡访问日志采集后，在顶部 Tab 栏中，选择**仪表盘**。



2. 在 CLB 实例下拉列表中，选中已开启访问日志投递的目标负载均衡实例。

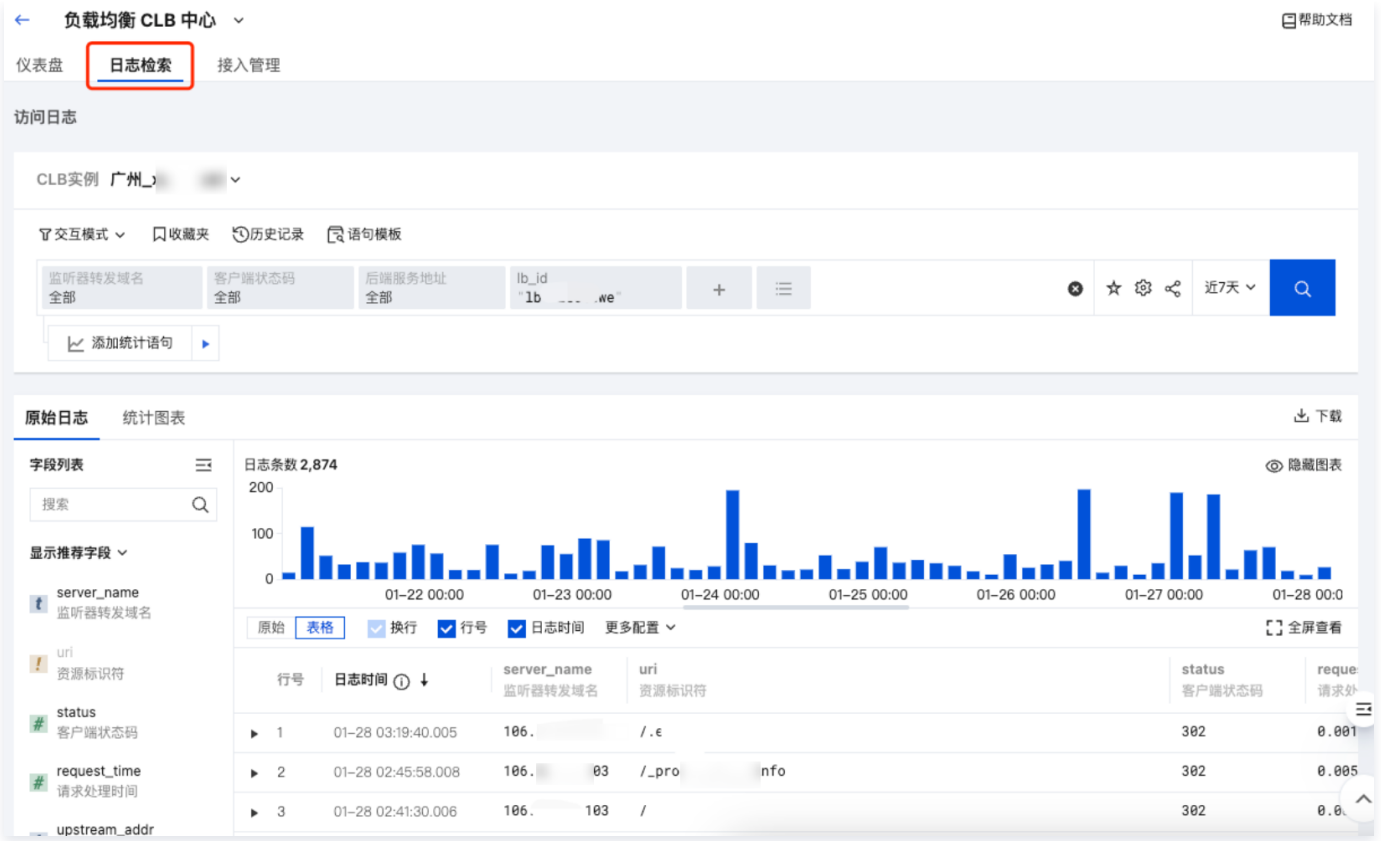


3. 选中实例后，即可查看与选中实例相关的 CLB 负载均衡访问分析大盘。

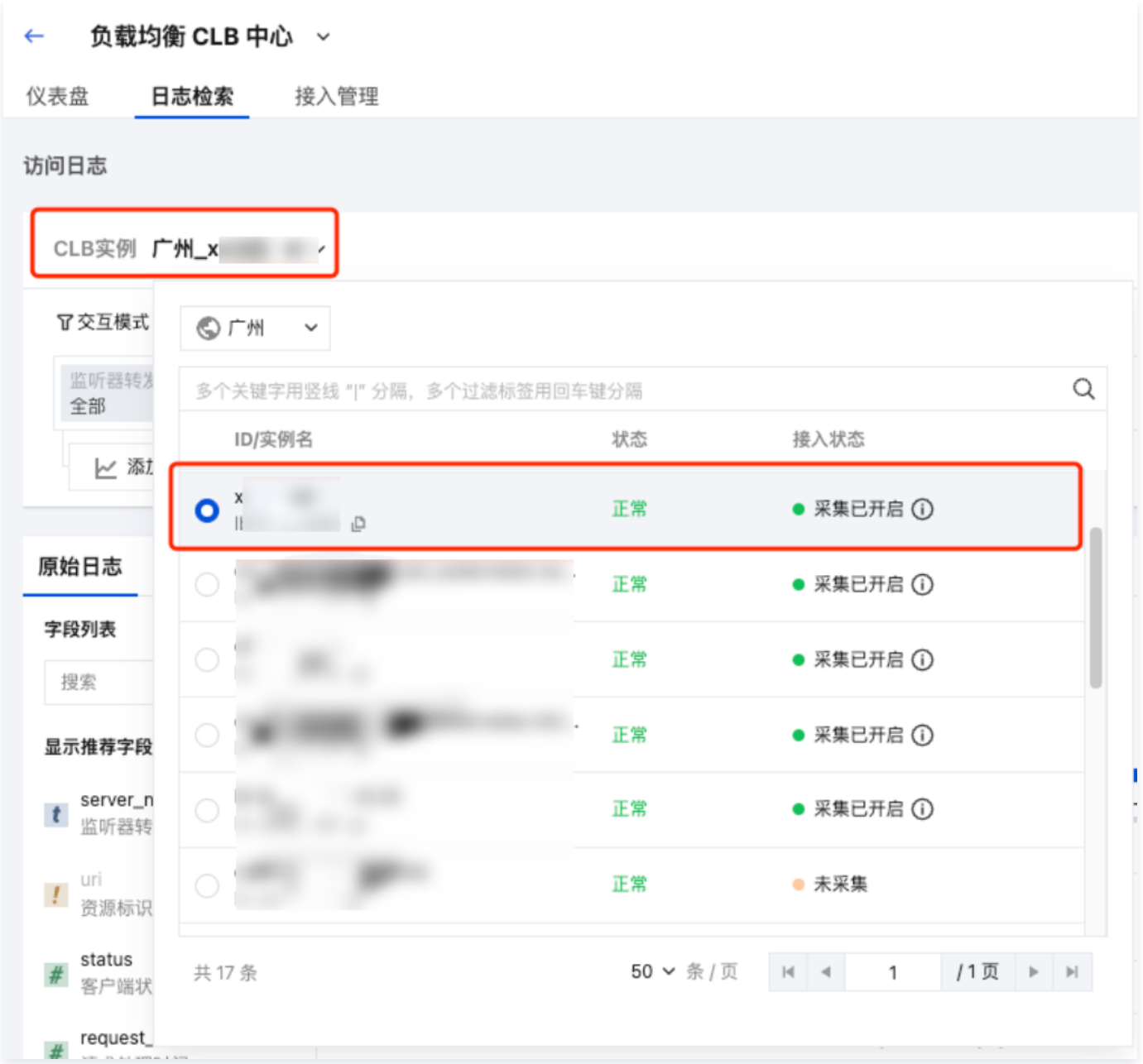


检索分析负载均衡访问日志

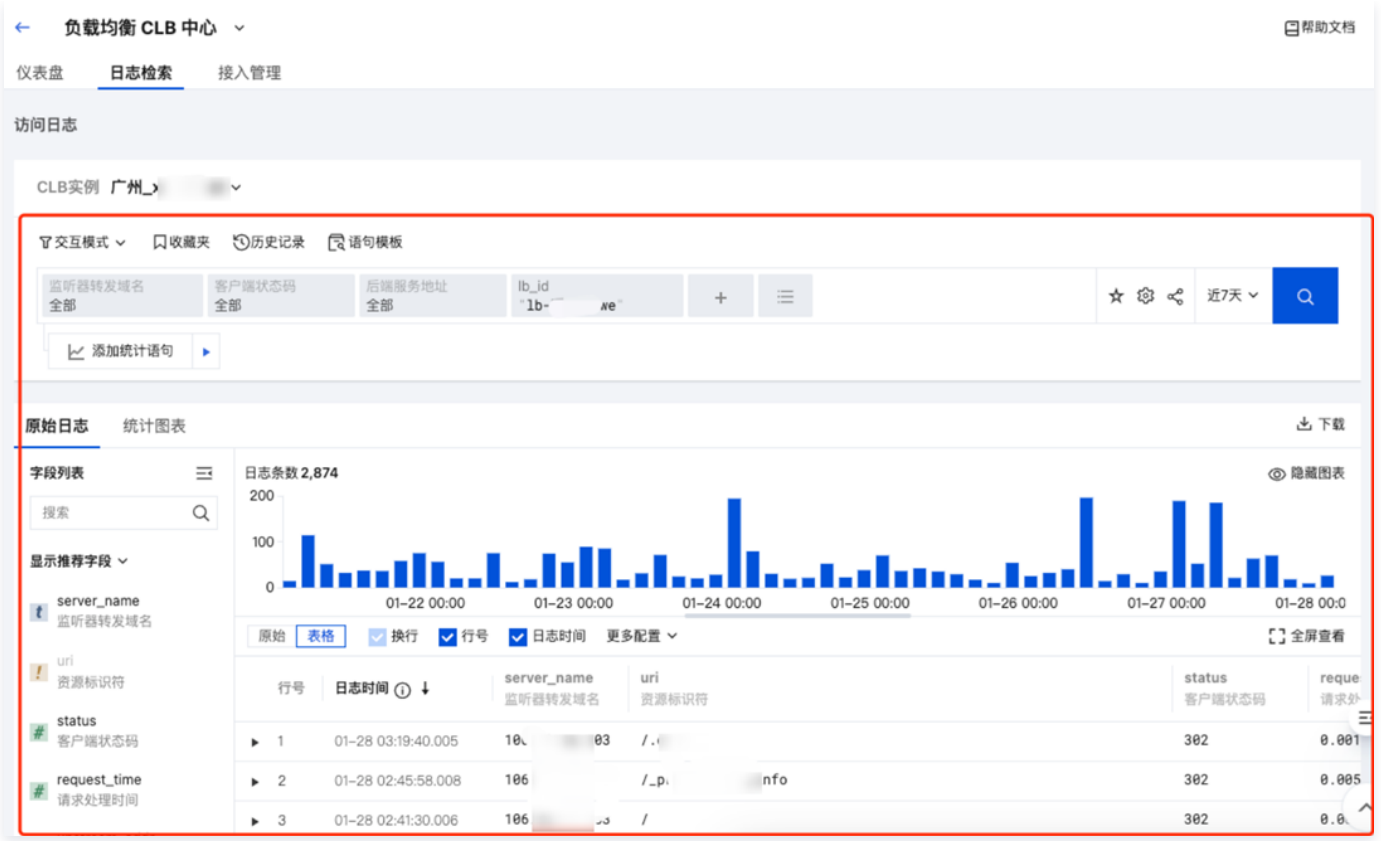
1. 完成开启负载均衡访问日志采集后，在顶部 Tab 栏中，选择**日志检索**。



2. 在 CLB 实例下拉列表中，选择已开启访问日志投递的目标负载均衡实例。

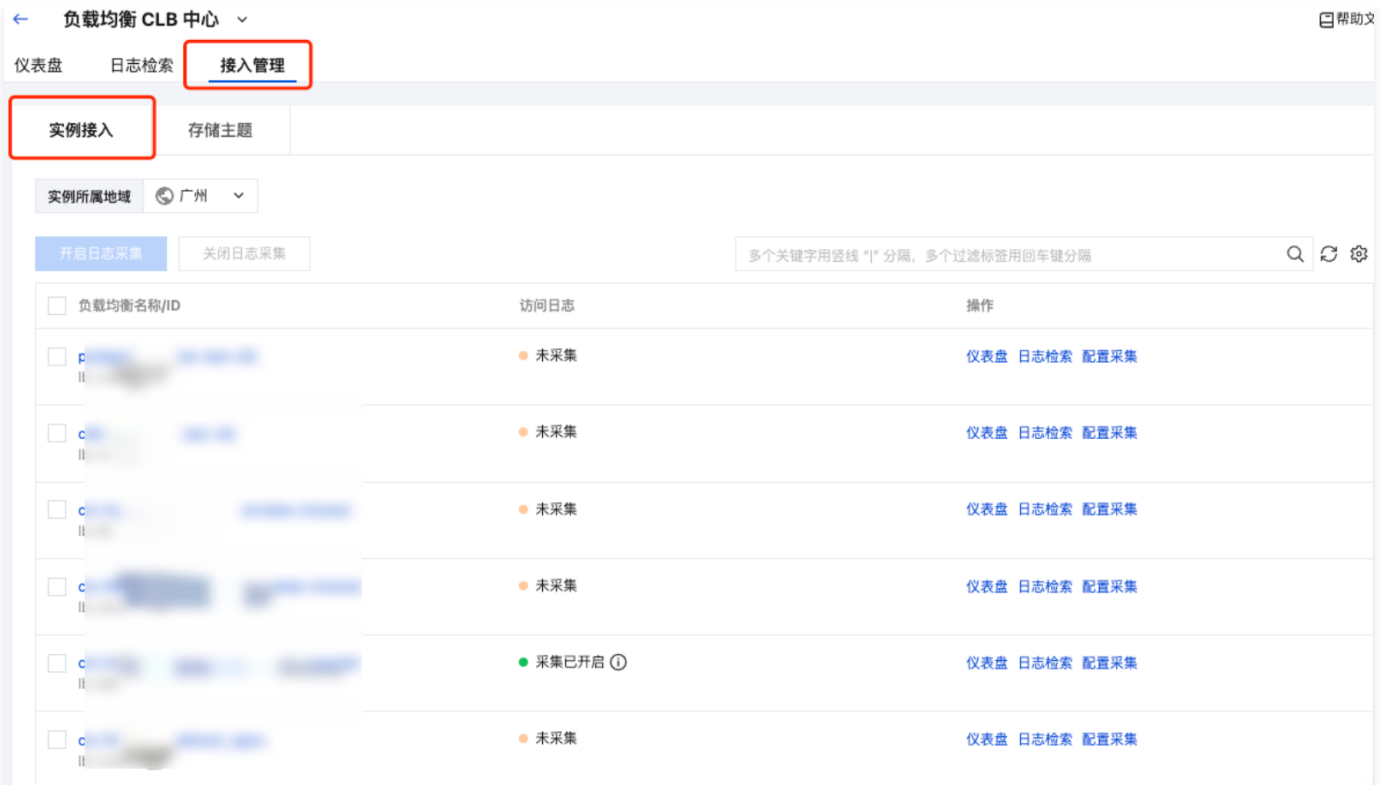


3. 选中实例后，即可检索分析与选中实例相关的 CLB 负载均衡访问日志。



关闭负载均衡访问日志采集

1. 在**负载均衡 CLB 中心**中，选择**接入管理 > 实例接入**。



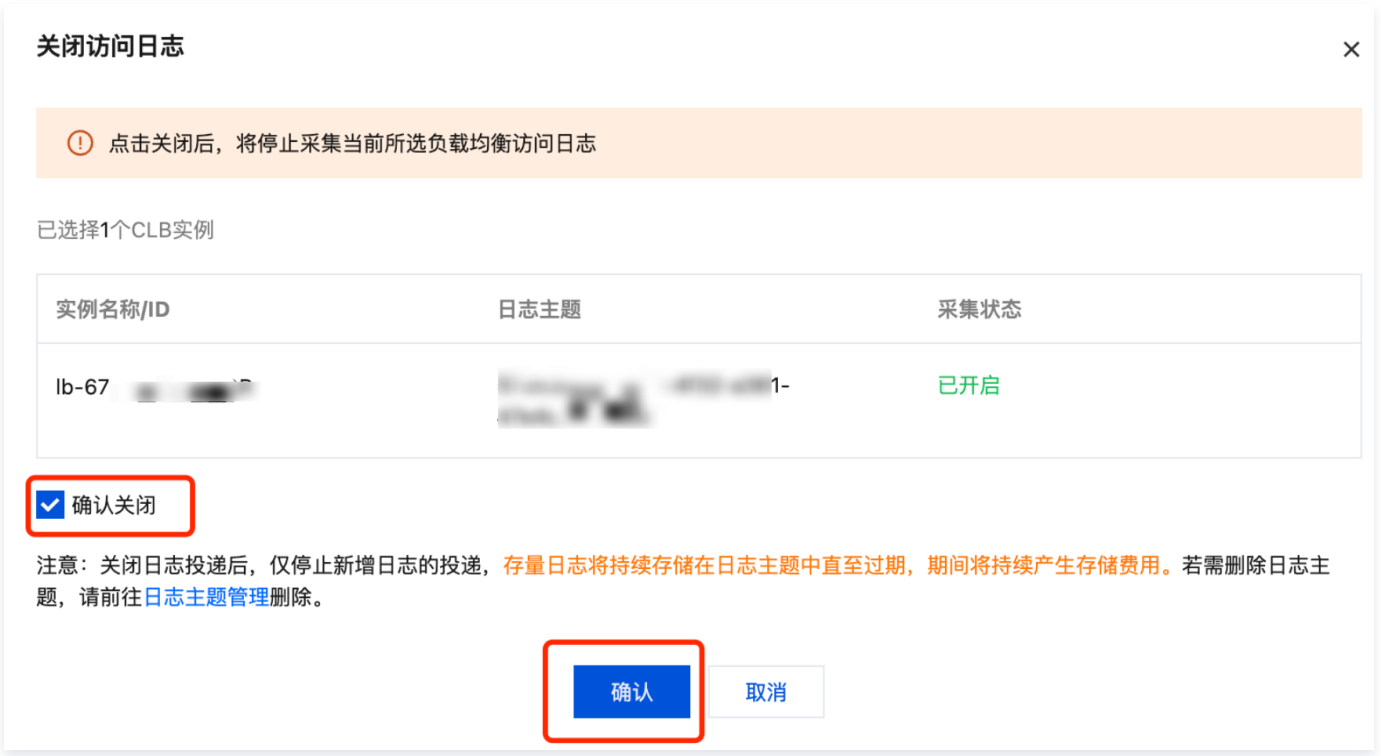
2. 在 CLB 实例列表中选中需要关闭日志采集的实例，并单击**关闭日志采集**。

> ⚠ **注意：**

> 关闭日志采集仅代表不再采集新增日志，存量已采集日志将持续存储与日志主题中直至过期，期间将持续产生 存储费用 。若不希望持续存储，请选择**接入管理** > **存储主题**，并在其中找到目标主题并删除。



3. 在关闭采集弹窗中，需再次**确认关闭**，然后单击**确认**即可关闭选中负载均衡的日志采集。



CDN 云产品中心

最近更新时间：2025-02-24 20:28:12

CDN 云产品中心提供内容分发网络 CDN 访问日志的一站式采集与检索分析，同时基于访问日志，提供开箱即用的访问分析仪表盘。助力快速构建 CDN 访问日志分析的可观测性。

功能特性

- 开箱即用 CDN 域名访问分析仪表盘。
- 统一 CDN 域名访问日志检索分析入口。
- 批量开启 CDN 域名访问日志采集。

前提条件

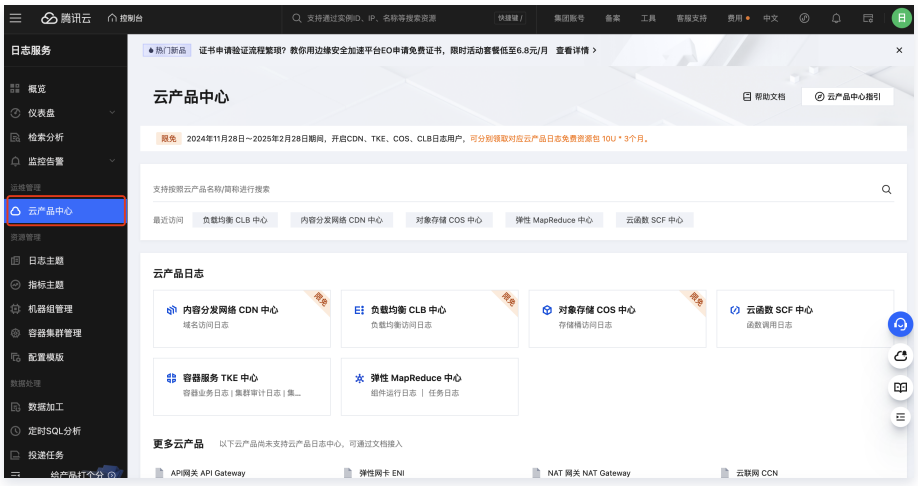
已 创建 CDN 域名 。

操作步骤

开启域名访问日志采集

步骤1：进入 CDN 云产品中心

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**云产品中心**，进入云产品中心页面。

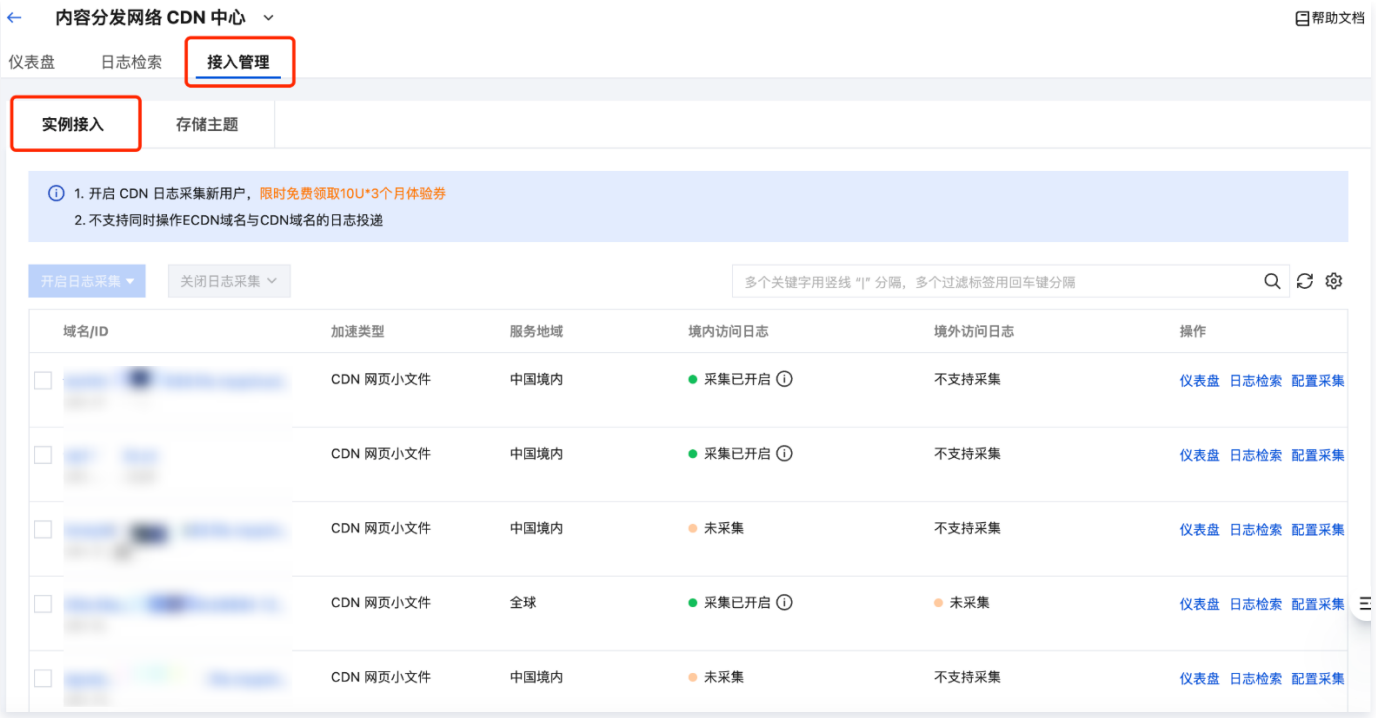


3. 在**云产品日志**中，选择并单击**内容分发网络 CDN 中心**。



步骤2：开启域名访问日志采集

1. 在**内容分发网络 CDN 中心**中，选择**接入管理 > 实例接入**。



2. 在 CDN 域名列表中，找到并勾选一个或多个目标 CDN 域名。



3. 单击**开启日志采集**，在下拉选项中选择开启**境内访问日志**或**境外访问日志**。

> **注意：**
> - ECDN 域名与境内域名不支持开启境外访问日志采集。
> - 境外域名不支持开启境内访问日志采集。



4. 在开启日志采集弹窗中，选择或创建目标 日志主题，并单击**确认**，即可开启目标 CDN 域名的境内或境外访问日志采集。

> **说明：**
> 仅支持自定义目标存储日志主题， 日志集默认为 CDN 专用日志集，日志集名称为 cloud_cdn_logset。



查看域名分析仪表盘

1. 完成开启 CDN 域名访问日志采集后，在顶部 Tab 栏中，选择**仪表盘**。



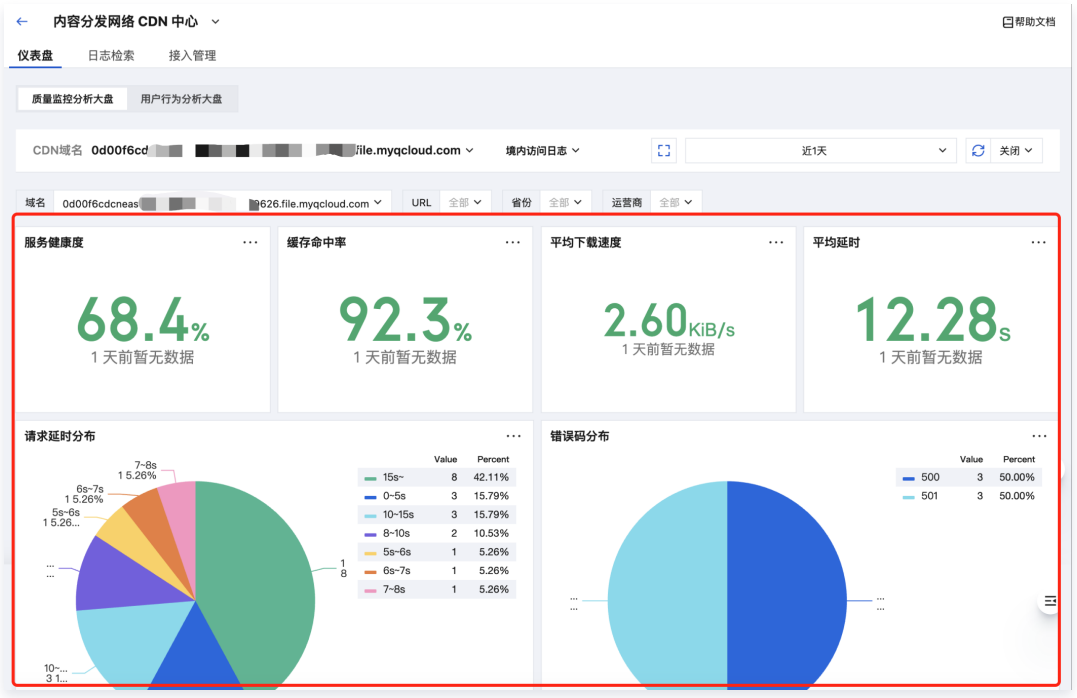
2. 在顶部二级 Tab 中，选中目标仪表盘。



3. 在 CDN 域名下拉列表中，选中已开访问日志投递的目标 CDN 域名，并选择目标需要分析的地域， 即**境内访问日志**或**境外访问日志**。

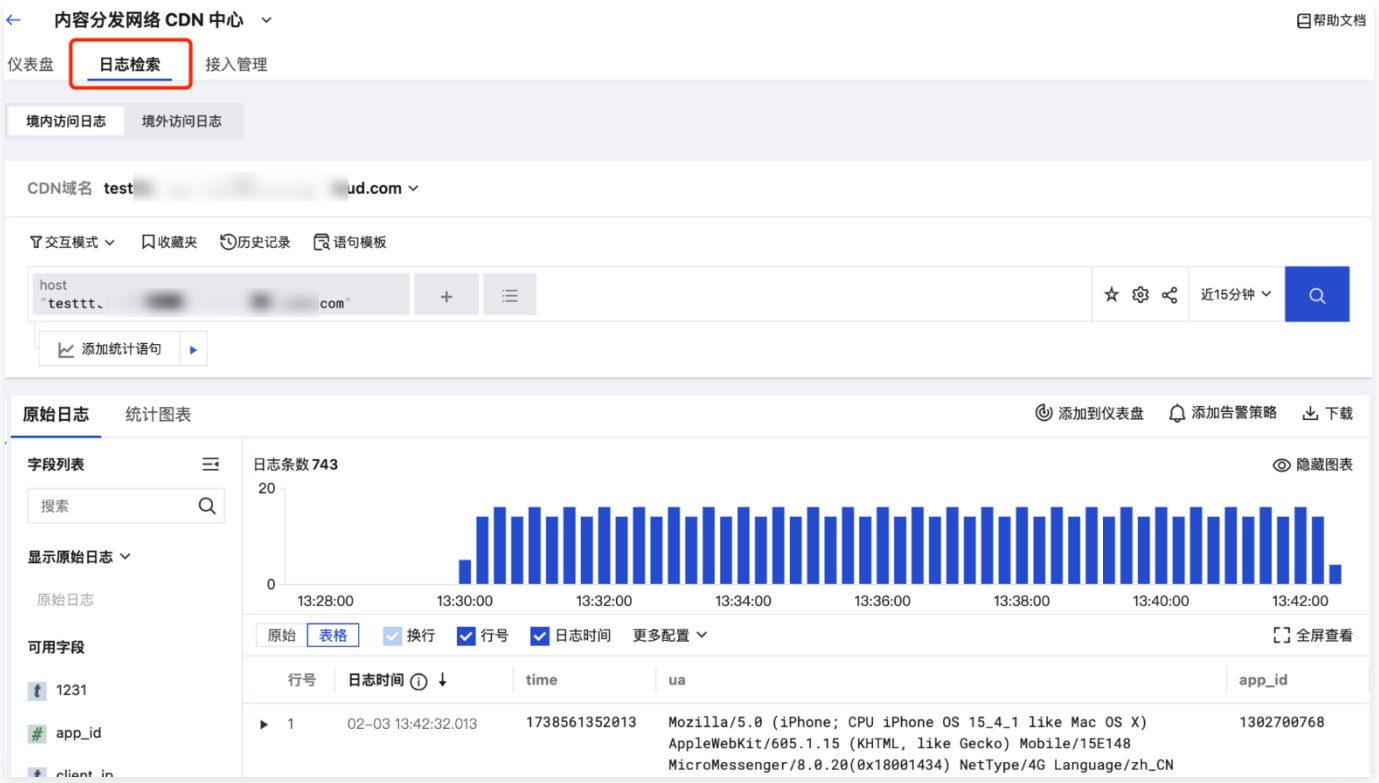


4. 选中域名后，即可查看与选中域名相关的访问分析仪表盘。



检索分析负载均衡访问日志

1. 完成开启 CDN 域名访问日志采集后，在顶部 Tab 栏中，选择**日志检索**。



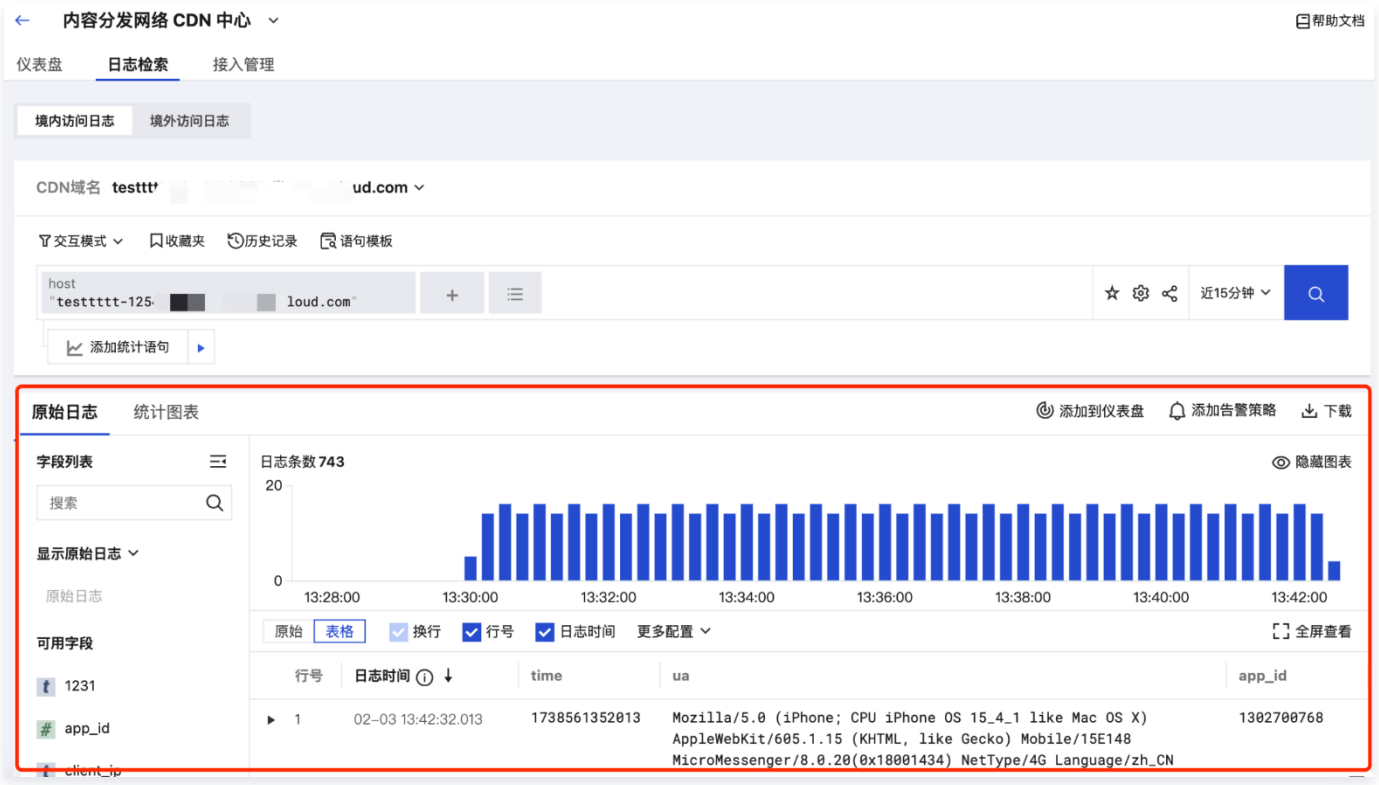
2. 选择目标需要分析的地域，即**境内访问日志**或**境外访问日志**。



3. 在 CDN 域名下拉列表中，选中已开启访问日志投递的目标 CDN 域名。



4. 选中实例后，即可检索分析与选中 CDN 域名相关的境内或境外访问日志。



关闭域名访问日志采集

1. 在**内容分发网络 CDN 中心**中，选择**接入管理 > 实例接入**。



2. 在列表中选中需要关闭日志采集的实例，并单击**关闭日志采集**，在下拉选项中选择关闭境内或境外域名的访问日志采集。

> **注意：**
> 关闭日志采集仅代表不再采集新增日志，存量已采集日志将持续存储与日志主题中直至过期，期间将持续产生 存储费用 。若不希望持续存储，请 选择**接入管理** > **存储主题**，在其中找到目标主题并删除。



3. 在关闭采集弹窗中，需再次**确认关闭**，然后单击**确认**即可关闭选中域名的日志采集。



COS 云产品中心

最近更新时间：2025-02-24 20:28:12

COS 云产品中心提供对象存储 COS 访问日志的一站式采集与检索分析，同时基于访问日志，提供开箱即用的访问分析仪表盘。助力快速构建 COS 访问日志分析的可观测性。

功能特性

- 开箱即用存储桶访问分析仪表盘。
- 统一 COS 存储桶访问日志检索分析入口。
- 批量开启 COS 访问日志日志采集。

前提条件

已 创建 COS 存储桶 。

操作步骤

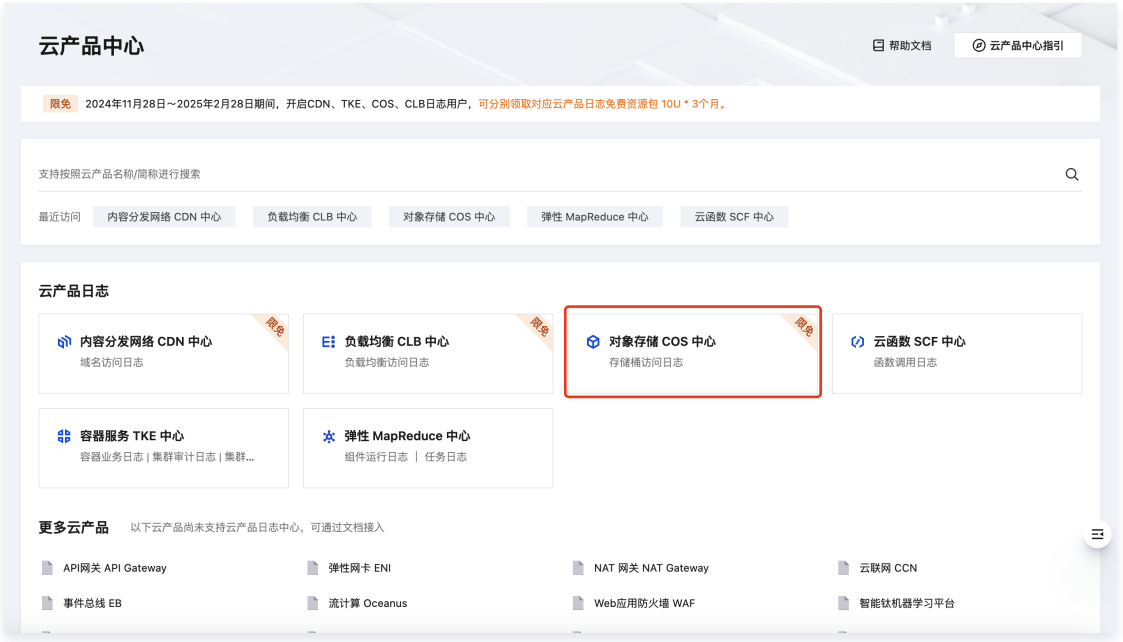
开启存储桶访问日志采集

步骤1：进入 COS 云产品中心

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**云产品中心**，进入云产品中心页面。



3. 在**云产品日志**中，选择并单击**对象存储 COS 中心**。



步骤2：开启对象存储访问日志采集

1. 在**对象存储 COS 中心**中，选择**接入管理** > **实例接入**。



2. 选中目标存储桶所在地域，在存储桶列表中，找到并勾选一个或多个目标存储桶实例。



3. 单击**开启日志采集**。



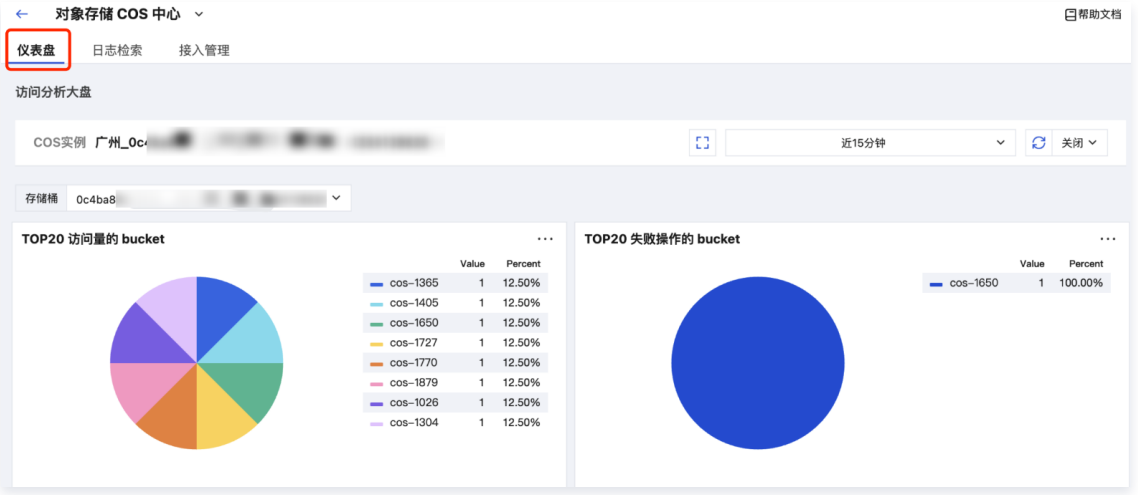
4. 在开启日志采集弹窗中，单击**确认**，即可开启目标存储桶实例的访问日志采集。

> **说明：**
> 目标日志主题与日志集默认为 COS 对象存储专用日志主题与日志集，日志主题名称为 cos-log-store，日志集名称为 cos-log-ap-${存储桶所在地域}-${UIN}。

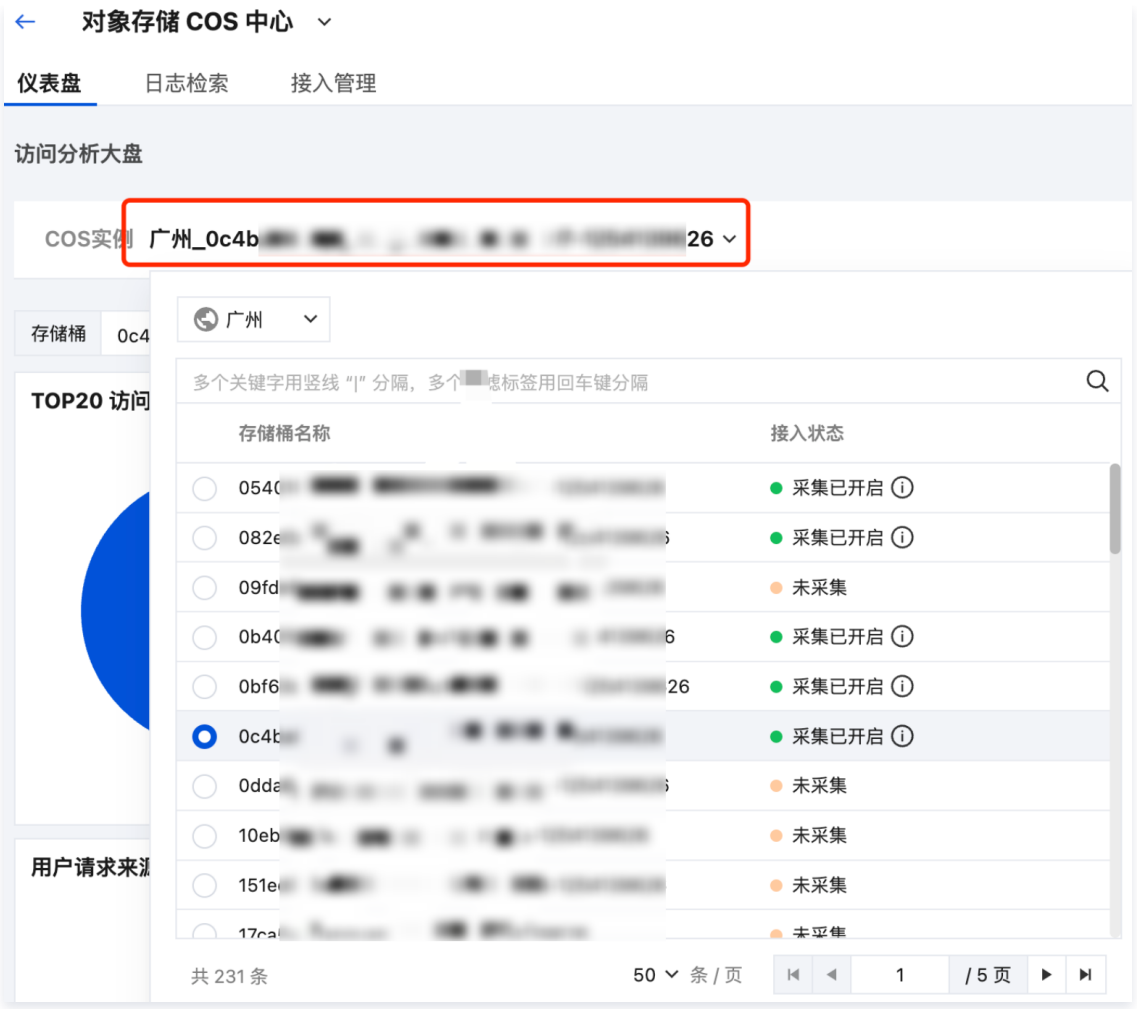


查看存储桶访问分析仪表盘

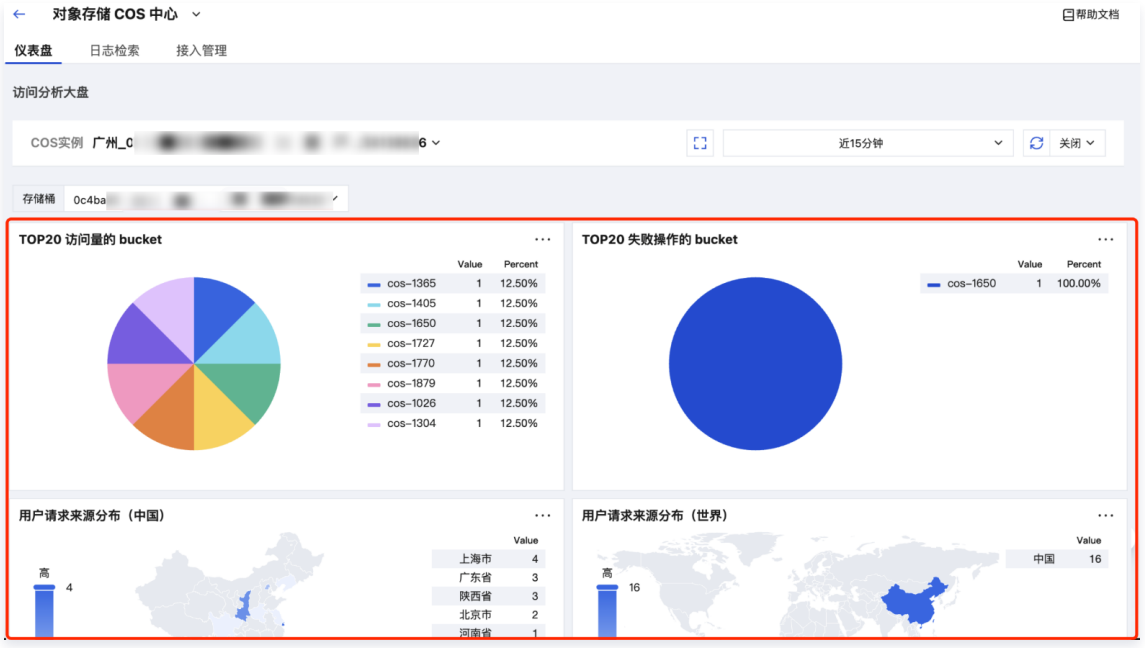
1. 完成开启存储桶访问日志采集后，在顶部 Tab 栏中，选择**仪表盘**。



2. 在存储桶下拉列表中，选中已开启访问日志投递的目标存储桶。

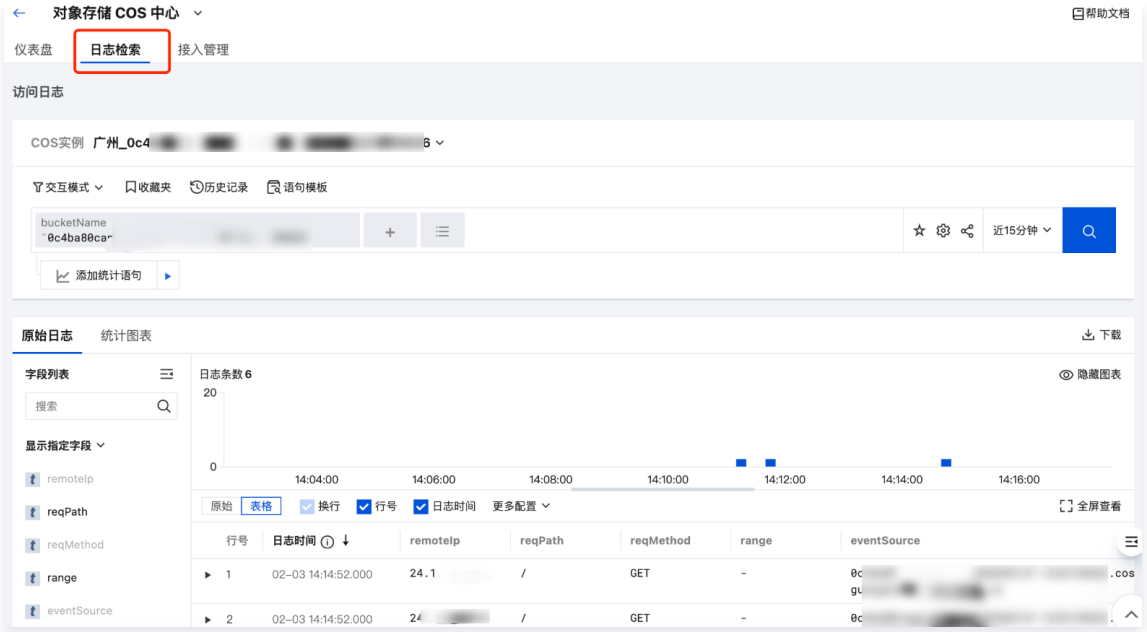


3. 选中实例后，即可查看与选中存储桶相关的存储桶访问分析大盘。

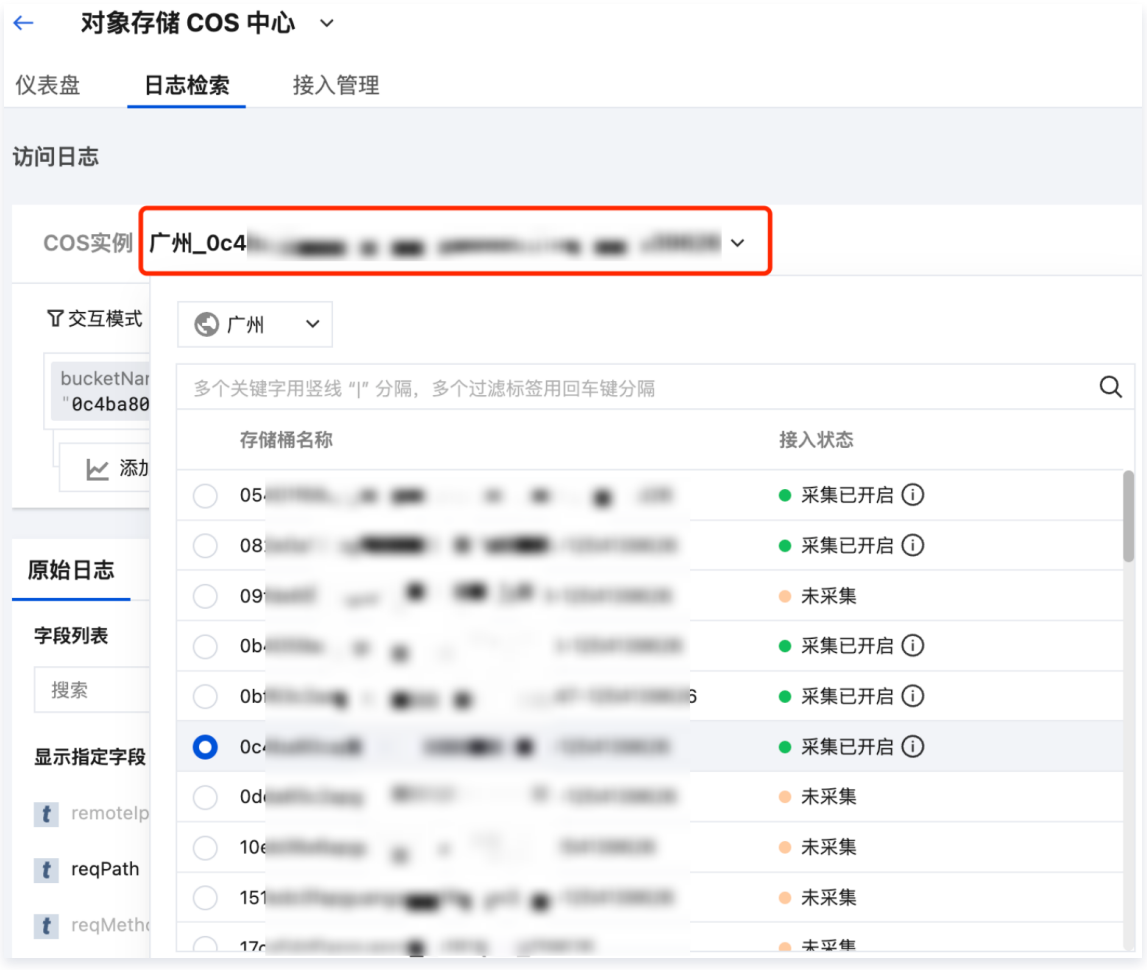


检索分析存储桶访问日志

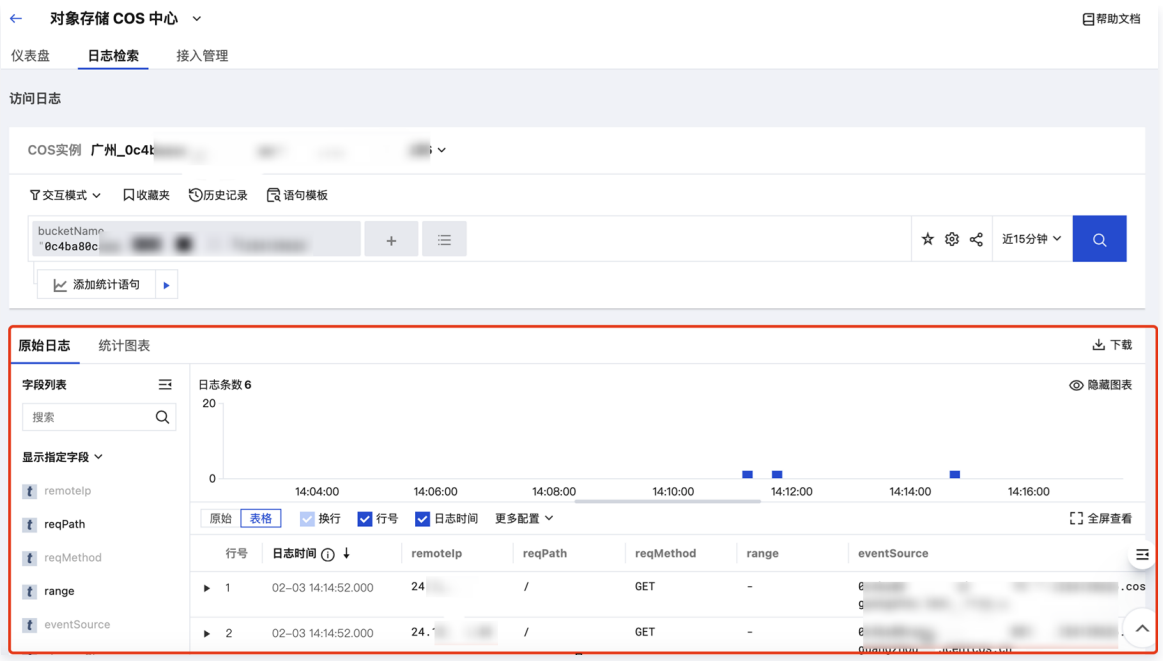
1. 完成开启存储桶访问日志采集后，在顶部 Tab 栏中，选择**日志检索**。



2. 在存储桶下拉列表中，选中已开启访问日志投递的目标存储桶。



3. 选中实例后，即可检索分析与选中存储桶相关的存储桶访问日志。



关闭存储桶访问日志采集

1. 在**对象存储 COS 中心**中，选择**接入管理 > 实例接入**。



2. 在存储桶列表中选中需要关闭日志采集的存储桶，并单击**关闭日志采集**。

> **注意：**
> 关闭日志采集仅代表不再采集新增日志，存量已采集日志将持续存储与日志主题中直至过期，期间将持续产生 存储费用 。若不希望持续存储，请选择**接入管理 > 存储主题**，在其中找到目标主题并删除。



3. 在关闭采集弹窗中，需再次**确认关闭**，然后单击**确认**即可关闭选中存储桶的日志采集。



SCF 云产品中心

最近更新时间：2025-02-24 20:28:12

SCF 云产品中心提供云函数 SCF 调用日志的一站式采集与检索分析，助力您快速构建云函数调用分析的可观测性。

功能特性

- 统一云函数调用日志检索分析入口。
- 批量开启云函数调用日志采集。

前提条件

已 创建 SCF 云函数 。

操作步骤

开启云函数调用日志采集

步骤1：进入 SCF 云产品中心

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**云产品中心**，进入云产品中心页面。
3. 在**云产品中心**中，选择并单击**云函数 SCF 中心**。

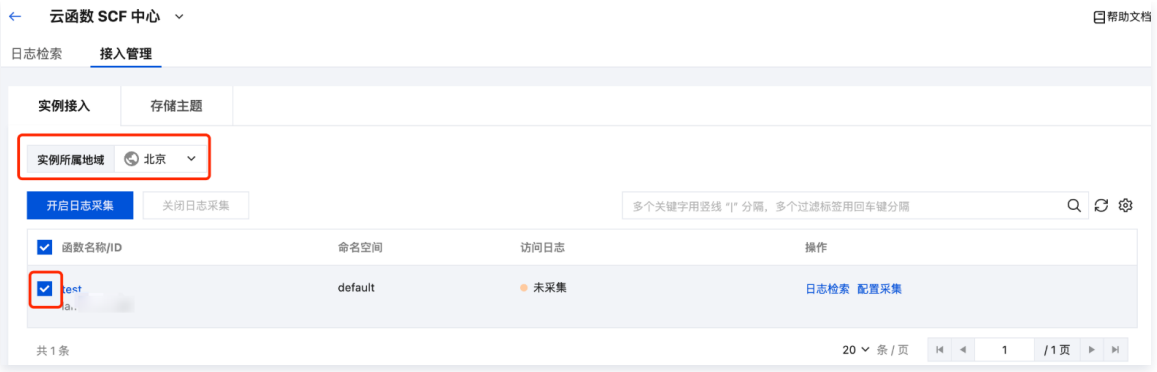


步骤2：开启云函数调用日志采集

1. 在**云函数 SCF 中心**中，选择**接入管理 > 实例接入**。



2. 选中目标云函数所在地域，在云函数列表中，找到并勾选一个或多个目标云函数。



3. 单击**开启日志采集**。



4. 在开启日志采集弹窗中，选择或创建目标 日志主题，并单击**确认**，即可开启目标云函数的调用日志采集。

> **说明：**
> 目标日志主题与日志集默认为 SCF 云函数专用日志主题与日志集，日志主题名称为 SCF_logtopic_${自动生成唯一标识}，日志集名称为 SCF_logset_${自动生成唯一标识}。

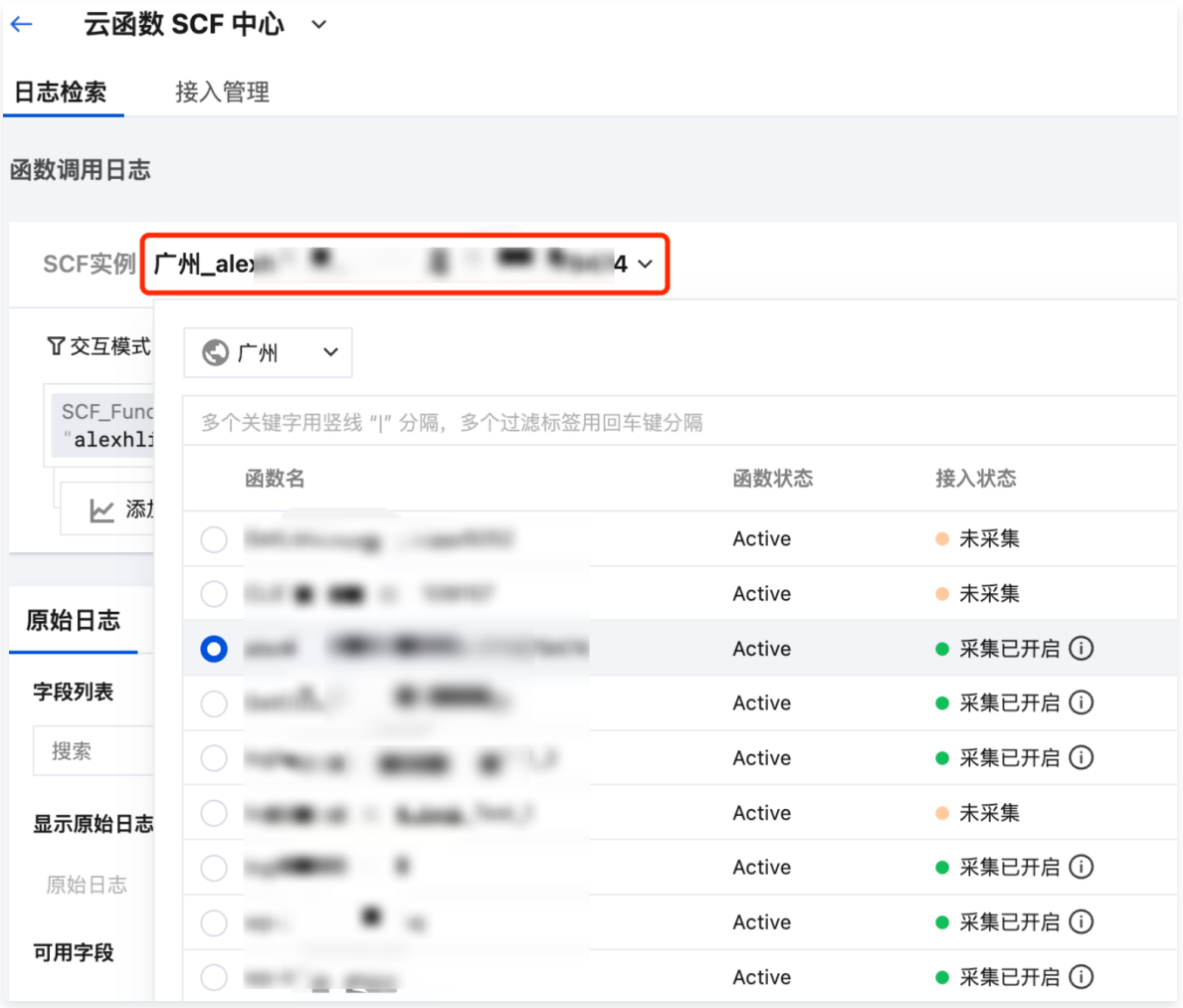


检索分析云函数调用日志

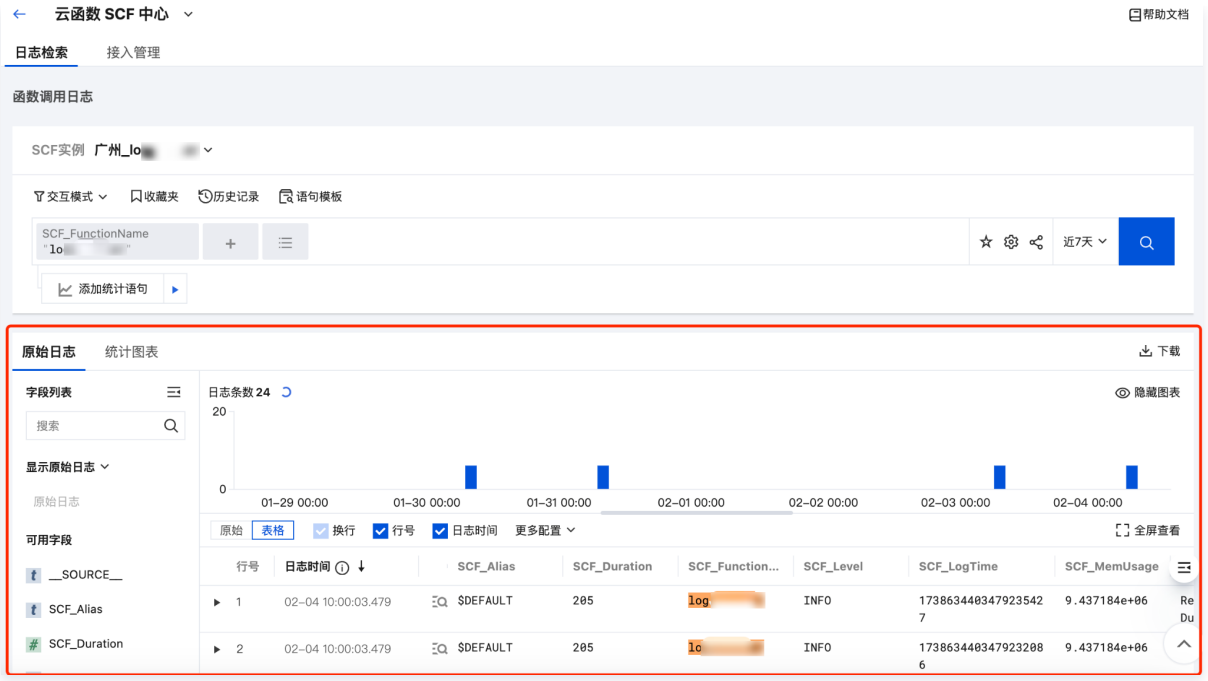
1. 完成开启云函数调用日志采集后，在顶部 Tab 栏中，选择**日志检索**。



2. 在云函数下拉列表中，选中已开启访问日志投递的目标云函数。



3. 选中云函数后，即可检索分析与选中云函数相关的函数调用日志。



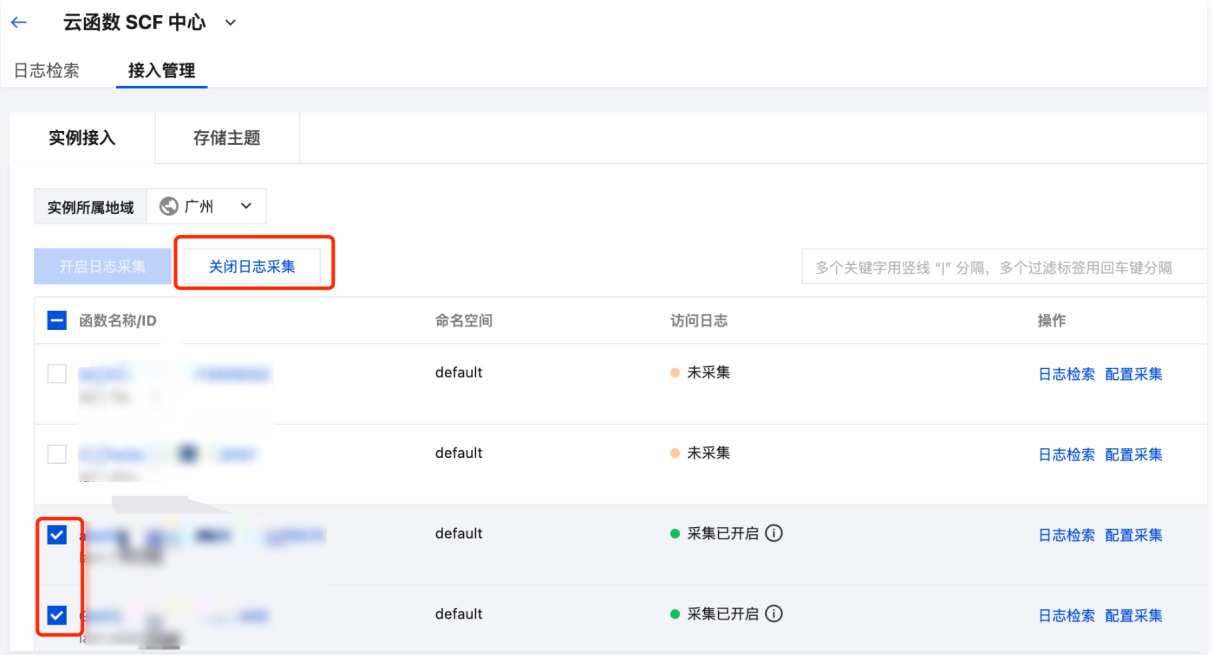
关闭云函数调用日志采集

1. 在**云函数 SCF 中心**中，选择**接入管理** > **实例接入**。



2. 在云函数列表中选中需要关闭日志采集的云函数，并单击**关闭日志采集**。

> **注意：**
> 关闭日志采集仅代表不再采集新增日志，存量已采集日志将持续存储与日志主题中直至过期，期间将持续产生 存储费用 。若不希望持续存储，请选择**接入管理** > **存储主题**，在其中找到目标主题并删除。



3. 在关闭采集弹窗中，需再次**确认关闭**，然后单击**确认**即可关闭选中云函数的日志采集。



TKE 云产品中心

最近更新时间：2025-02-24 20:28:12

TKE 云产品中心提供容器服务 TKE 业务日志、审计日志、事件日志的一站式采集与检索分析，同时基于审计与事件日志，提供开箱即用的审计与事件分析仪表盘，助力快速构建 TKE 访问分析的可观测性。

功能特性

- 开箱即用 TKE 集群审计与事件分析仪表盘。
- 统一 TKE 集群审计与事件日志检索分析入口。
- 批量开启 TKE 集群业务、审计与事件日志采集。

前提条件

已 创建 TKE 集群。CLS 支持采集 TKE 的 标准集群 和 边缘集群。

操作步骤

开启集群业务日志采集

步骤1：选择集群

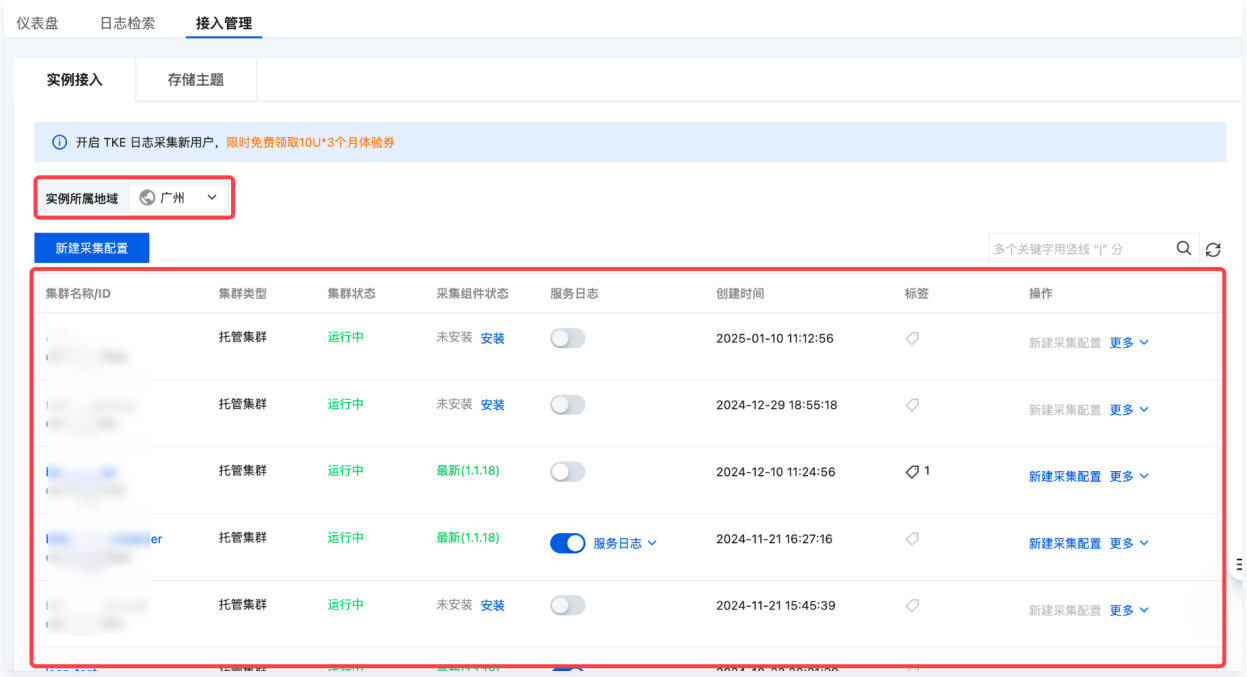
1. 登录 日志服务控制台。
2. 在左侧导航栏中，单击**云产品中心**，进入云产品中心页面。
3. 在**云产品日志中心**中，选择并单击**容器服务 TKE 中心**。



4. 在**容器服务 TKE 中心**中，选择**接入管理** > **实例接入**。



5. 选择 TKE 集群所在地域，并在集群列表中找到目标采集集群。

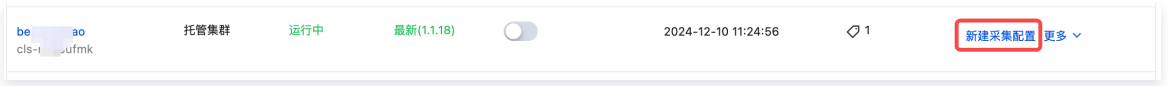


若采集组件状态为**未安装**，单击**安装**，安装日志采集组件。



> **注意：**
> - 安装日志采集组件将在集群 kube-system 命名空间下，以 DaemonSet 的方式部署一个 tke-log-agent 的 pod 和一个 cls-provisioner 的 pod。请为每个节点至少预留0.1核16Mib以上的可用资源。
> - 在 TKE 集群上安装日志采集组件需要具有访问集群的 BRAC 权限，若没有请联系集群管理员添加。

6. 当采集组件状态为**最新**，单击右侧**新建采集配置**，进入集群日志采集配置流程。



> **注意：**
> 为 TKE 集群配置日志采集配置需要具有访问集群的 BRAC 权限，若没有请联系集群管理员添加。

步骤2：配置日志主题

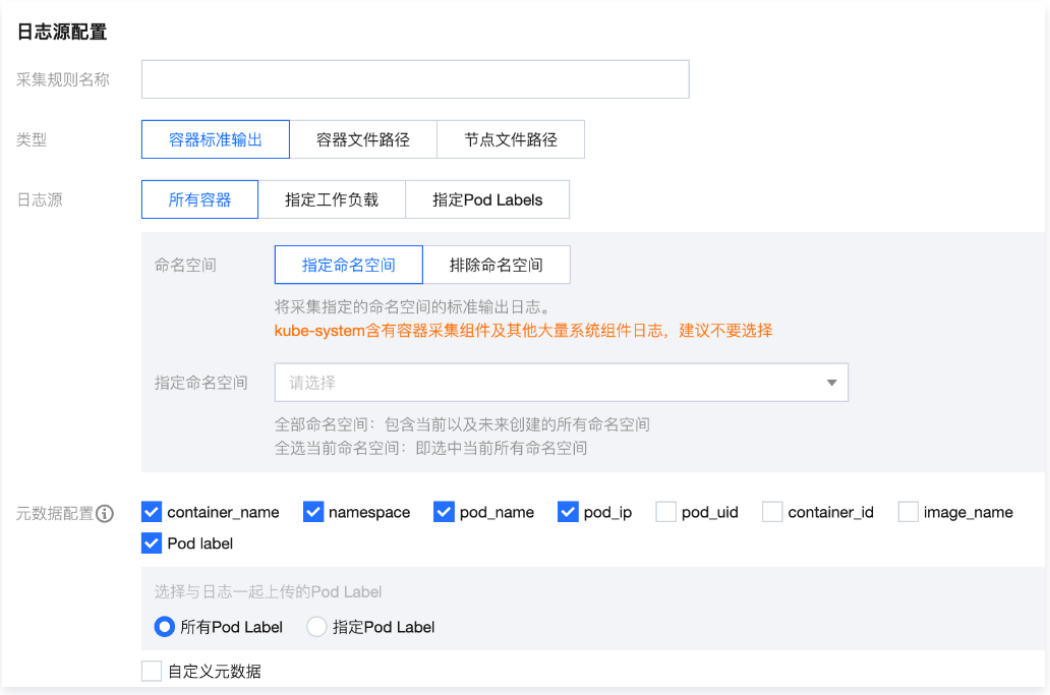
进入集群日志采集配置流程，在**创建日志主题**步骤中，您可选择已有或创建用于存储日志的日志主题。关于日志主题请参见 日志主题 。



步骤3：配置采集规则

完成日志主题的选择后，单击**下一步进入采集配置**即可配置采集规则，配置信息如下：

- **日志源配置**：



  - **采集规则名称**：您可以自定义日志采集规则名称。
  - **采集类型**：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。

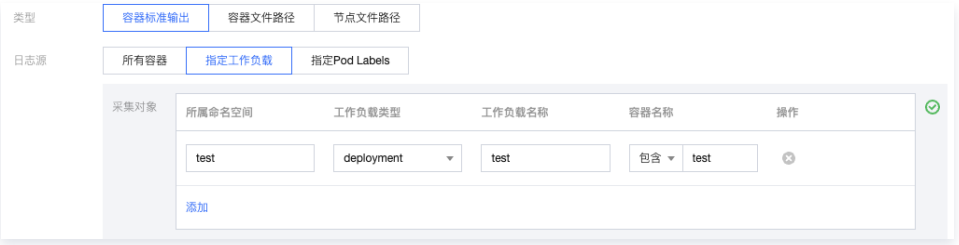
容器标准输出

支持所有容器、指定工作负载、指定 Pod Lables 三种方式指定容器标准输出的采集日志源。

- **所有容器**：代表采集指定命名空间下的所有容器中的标准输出日志，如下图所示：



- **指定工作负载**：代表采集指定命名空间下，指定工作负载中，指定容器的标准输出日志如下图所示：



- **指定 Pod Lables**：代表采集指定命名空间下，所有带有指定 Pod Label 的容器中的标准输出日志，如下图所示：



容器文件路径

> **注意：**
> **容器文件路径不能为软链接**，否则会导致软链接的实际路径在采集器的容器内不存在，采集日志失败。

支持指定工作负载、指定 Pod Lables 两种方式指定容器文件路径的采集日志源。

- **指定工作负载**：代表采集指定命名空间下、指定工作负载中、指定容器中的容器文件路径，如下图所示：



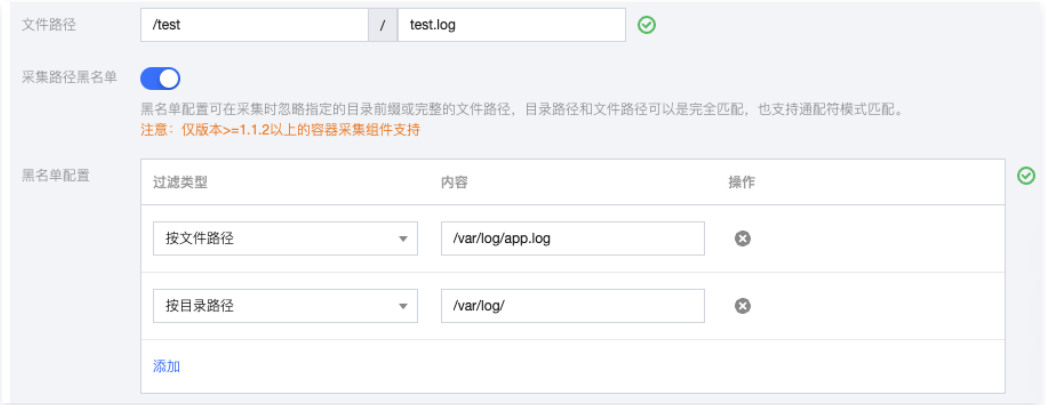
- **指定 Pod Lables**：代表采集指定命名空间下，所有带有指定 Pod Label 的容器中的容器文件路径，如下图所示：



  - **文件路径**：由日志目录与文件名组成，日志目录前缀以 / 开头，文件名以非 / 开头，前缀和文件名均支持使用通配符 ？和 *，不支持逗号。/**/代表日志采集组件将监听所填前缀目录下的所有层级匹配上的日志文件。多文件路径之间为或关系。例如：当容器文件路径为 `/opt/logs/*.log`，可以指定目录前缀为 `/opt/logs`，文件名为 `*.log` 的日志。

    > **注意：**
    > - 仅版本1.1.12及以上的容器采集组件支持多采集路径。
    > - 仅升级至1.1.12及以上版本后创建的采集配置支持定义多采集路径。
    > - 版本1.1.12以下创建的采集配置在升级至1.1.12后不支持配置多采集路径，需重新创建采集配置。

  - **采集路径黑名单**：开启后可在采集时忽略指定的目录路径或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



  - **黑名单配置**：采集黑名单分为两类过滤类型，且可以同时使用：

- **文件路径**：采集路径下，需要忽略采集的完整文件路径，支持通配符*或？，支持**路径模糊匹配。
- **目录路径**：采集路径下，需要忽略采集的目录前缀，支持通配符*或？，支持**路径模糊匹配。

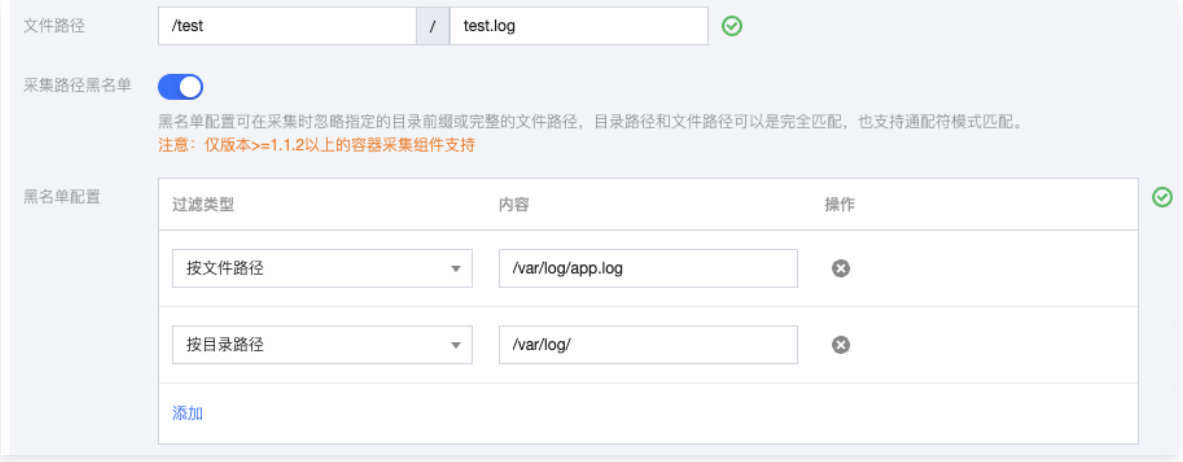
> ⚠ **注意：**
> - 需要容器日志采集组件1.1.2及以上版本。
> - 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

节点文件路径

- **节点文件路径**由日志目录与文件名组成，日志目录前缀以 / 开头，文件名以非 / 开头，前缀和文件名均支持使用通配符？和 *，不支持逗号。/**/代表日志采集组件将监听所填前缀目录下的所有层级匹配上的日志文件。多文件路径之间为或关系。例如：当节点文件路径为 `/opt/logs/*.log`，可以指定目录前缀为 `/opt/logs`，文件名为 `*.log` 的日志。

> ⚠ **注意：**
> - 仅版本1.1.12及以上的采集组件支持多采集路径。
> - 仅升级至1.1.12及以上版本后创建的采集配置支持定义多采集路径。
> - 版本1.1.12以下创建的采集配置在升级至1.1.12后不支持配置多采集路径，需重新创建采集配置。

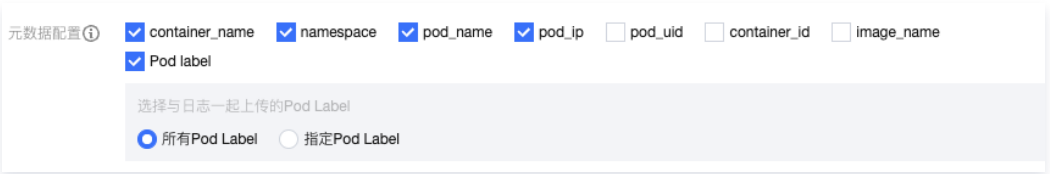
- **采集路径黑名单**：开启后可在采集时忽略指定的目录路径或完整的文件路径。目录路径和文件路径可以是完全匹配，也支持通配符模式匹配。



- **黑名单配置**：采集黑名单分为两类过滤类型，且可以同时使用：
  - **文件路径**：采集路径下，需要忽略采集的完整文件路径，支持通配符*或？，支持**路径模糊匹配。
  - **目录路径**：采集路径下，需要忽略采集的目录前缀，支持通配符*或？，支持**路径模糊匹配。

> ⚠ **注意：**
> - 需要容器日志采集组件1.1.2及以上版本。
> - 采集黑名单是在采集路径下进行排除，因此无论是文件路径模式，还是目录路径模式，其指定路径要求为采集路径的子集。

- **元数据配置**：



除了原始的日志内容， 日志服务还会带上容器或 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。您可以自行选择是否需要上报这些元数据，按需勾选上传。
容器或 kubernetes 相关的元数据请参见下方表格：

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。
pod_lable_{label name}	日志所属 pod 的 label（例如一个 pod 带有两个 label：app=nginx，env=prod，则在上传的日志会附带两个 metadata：pod_label_app:nginx，pod_label_env:prod）。

说明：
如果想采集部分 podlabel，需要手动输入想要的 label key（可以输入多个，每输入一个以回车结束），命中将会采集。

- **解析规则配置：**



  - 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**：增量采集指只采集文件内新增的内容
  - **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：

单行全文格式

单行全文日志是指一行日志内容为一条完整的日志。日志服务在采集的时候，将使用换行符 \n 来作为一条日志的结束符。为了统一结构化管理，每条日志都会存在一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条日志原始数据为：

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

采集到日志服务的数据为：

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

多行全文格式

多行全文日志是指一条完整的日志数据可能跨占多行（例如 Java stacktrace）。在这种情况下，以换行符 \n 为日志的结束标识符就显得有些不合理，为了能让日志系统明确区分开每条日志，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。
多行全文也会设置一个默认的键值__CONTENT__，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。
假设一条多行日志原始数据为：

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
    at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
    at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

首行正则表达式为如下：

```
\d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2},\d{3}\s.+
```

采集到日志服务的数据为：

```
__CONTENT__:2019-12-15 17:13:06,043 [main] ERROR
com.test.logging.FooFactory:\njava.lang.NullPointerException\n    at
com.test.logging.FooFactory.createFoo(FooFactory.java:15)\n    at
com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

单行-完全正则格式

单行完全正则格式通常用来处理结构化的日志，指将一条完整日志按正则方式提取多个 key-value 的日志解析模式。
假设一条日志原始数据为：

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.1 200 782 9703
"http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum"
"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0"  0.354 0.354
```

配置的正则表达式为如下：

```
(\S+)[^\[]+(\[[^:]+:\d+:\d+:\d+\s\S+)\s"(\w+)\s(\S+)\s([^"]+)"\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s"
([^"]+)"\s"([^"]+)"\s+(\S+)\s(\S+).*
```

采集到日志服务的数据为：

```
body_bytes_sent: 9703
http_host: 127.0.0.1
```

```
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&orderBy=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

多行-完全正则格式

假设您的一条日志原始数据为：

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

行首正则表达式为：

```
\[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*
```

配置的自定义正则表达式为：

```
\[(\d+-\d+-\w+:\d+:\d+,\d+)\]\s\[(\w+)\]\s(.*)
```

系统根据 `()` 捕获组提取对应的 key-value 后，您可以自定义每组的 key 名称如下所示：

```
time: 2018-10-01T10:30:01,000`
level: INFO`
msg: java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

JSON 格式

假设您的一条 JSON 日志原始数据为：

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34
+0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost":"unix:/tmp/php-
cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","request":"POST
/event/dispatch HTTP/1.1","xff":"-","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0
(Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
```

```
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

分隔符

假设您的一条日志原始数据为：

```
10.20.20.10 - ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/1.1 ::: 127.0.0.1
::: 200 ::: 647 ::: 35 ::: http://127.0.0.1/
```

当日志解析的分隔符指定为 `:::` ，该条日志会被分割成八个字段，并为这八个字段定义唯一的 key，如下所示：

```
IP: 10.20.20.10 -
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

组合解析

假设您的一条日志的原始数据为：

```
1571394459, http://127.0.0.1/my/course/4|10.135.46.111|200, status:DEAD,
```

自定义插件内容如下：

```
{
 "processors": [
   {
     "type": "processor_split_delimiter",
     "detail": {
       "Delimiter": ",",
       "ExtractKeys": [ "time", "msg1","msg2"]
     },
     "processors": [
       {
         "type": "processor_timeformat",
         "detail": {
           "KeepSource": true,
           "TimeFormat": "%s",
```

```
            "SourceKey": "time"
          }
        },
        {
          "type": "processor_split_delimiter",
          "detail": {
            "KeepSource": false,
            "Delimiter": "|",
            "SourceKey": "msg1",
            "ExtractKeys": [ "submsg1","submsg2","submsg3"]
          },
          "processors": []
        },
        {
          "type": "processor_split_key_value",
          "detail": {
            "KeepSource": false,
            "Delimiter": ": ",
            "SourceKey": "msg2"
          }
        }
      ]
    }
  ]
}
```

经过日志服务结构化处理后，该条日志将变为如下：

```
time: 1571394459
submsg1: http://127.0.0.1/my/course/4
submsg2: 10.135.46.111
submsg3: 200
status: DEAD
```

- **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。

步骤4：配置索引

完成采集规则配置后，单击下一步，进入索引配置，有关索引请参见 索引 。并在索引配置中配置如下信息：

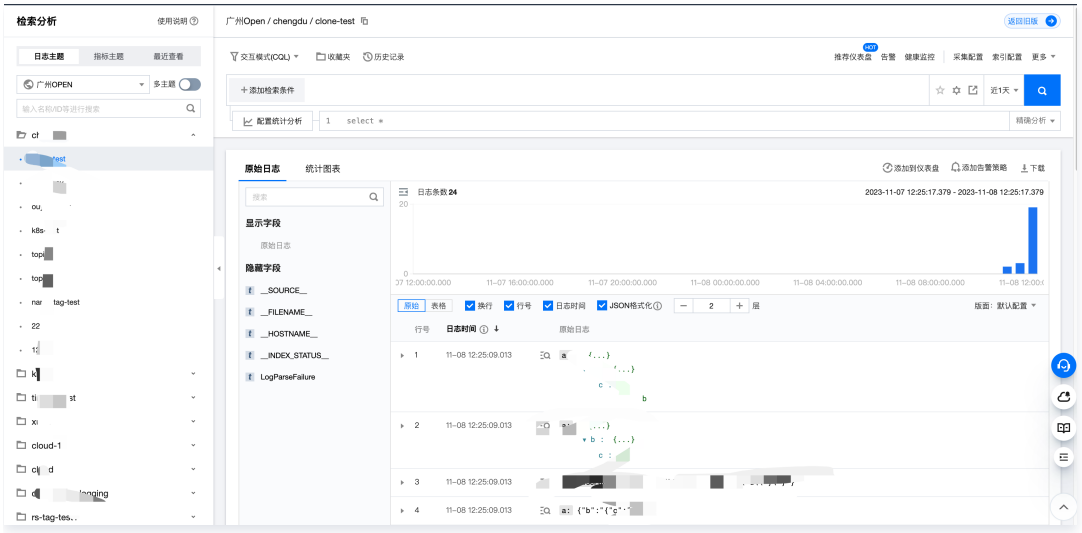


- **索引状态**：确认是否开启索引， 以使用日志检索等分析功能。
- **全文索引**：确认是否需要设置大小写敏感。全文分词符：默认为"@&()='",;:<>[]{}/ \n\t\r"，确认是否需要修改。
- **包含中文**：确认是否开启。
- **键值索引**：您可根据 key 名按需进行字段类型、分词符以及是否开启统计分析的配置。若您需要开启键值索引，可打开开关。

> **注意：**
> - 检索必须开启索引配置，否则无法检索。
> - 若需要基于日志字段检索日志，需配置键值索引。
> - 若需要基于日志字段进行统计分析，需配置键值索引，并开启统计。
> - 索引规则编辑后仅对新写入的日志生效，已有数据不会更新。

步骤5：检索业务日志

至此，即完成了 TKE 集群业务日志采集的所有部署。您可以前往 CLS 控制台 > 检索分析 查看采集上来的日志。



管理业务日志采集配置

1. 在**接入管理 > 实例接入**中，找到目标 TKE 集群，并单击集群名称，进入集群详情页。
2. 在集群详情页中，您可以在**集群业务日志**中查看并管理您的集群业务日志采集配置。



开启集群审计/事件日志采集

> **说明：**
> - 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。
> - 集群事件日志记录了包括了集群的运行和各类资源的调度情况。

步骤1：选择集群

1. 若采集组件状态为**未安装**，单击**安装**，安装日志采集组件。



> **注意：**
> 安装日志采集组件将在集群 kube-system 命名空间下，以 DaemonSet 的方式部署一个 tke-log-agent 的 pod 和一个 cls-provisioner 的 pod。请为每个节点至少预留0.1核16Mib以上的可用资源。

2. 若采集组件状态为**最新**，单击集群名称，进入集群详情页，并在集群详情页中找到**集群审计日志**或**集群事件日志**。



3. 单击**开启**集群审计或事件日志，并进入**集群审计日志**或**集群事件日志**配置流程。

步骤2：选择日志主题

进入审计或事件日志配置流程，在**创建日志主题**步骤中，您可选择已有或创建用于存储日志的日志主题。日志主题的相关信息请参见 日志主题 。



步骤3：索引配置

完成日志主题配置后，单击**下一步进入索引配置**，索引的相关信息请参见 索引 。索引配置中的配置信息如下：

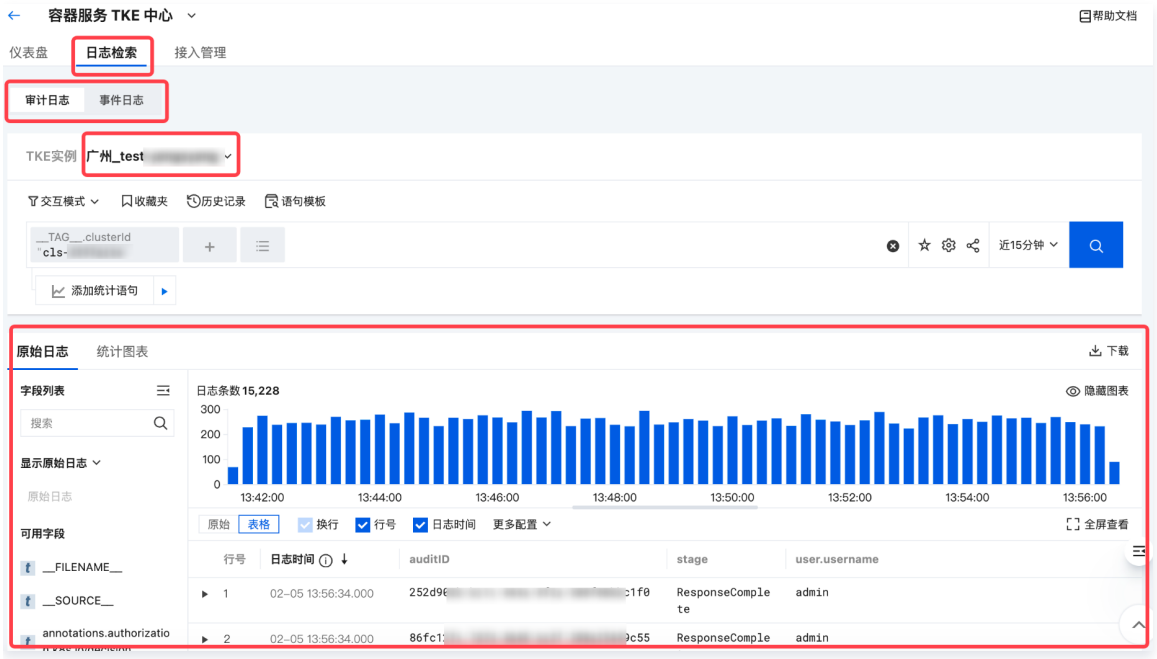


- **索引状态**：确认是否开启索引， 以使用日志检索等分析功能。
- **全文索引**：确认是否需要设置大小写敏感。全文分词符：默认为"@&()='",;:<>[]{}/ \n\t\r"，确认是否需要修改。
- **包含中文**：确认是否开启。
- **键值索引**：您可根据 key 名按需进行字段类型、分词符以及是否开启统计分析的配置。若您需要开启键值索引，可打开开关。

> **注意**
> - 检索必须开启索引配置，否则无法检索。
> - 若需要基于日志字段检索日志，需配置键值索引。
> - 若需要基于日志字段进行统计分析，需配置键值索引，并开启统计。
> - 索引规则编辑后仅对新写入的日志生效，已有数据不会更新。

步骤4：检索审计与事件日志

至此，即完成了 TKE K8S 集群审计或事件业务日志采集的所有部署。您可单击**日志检索**，选中要目标要检索的日志类型（审计或事件），同时选择目标集群，即可检索与选择集群相关的审计或事件日志。



查看 TKE 集群审计与事件分析仪表盘

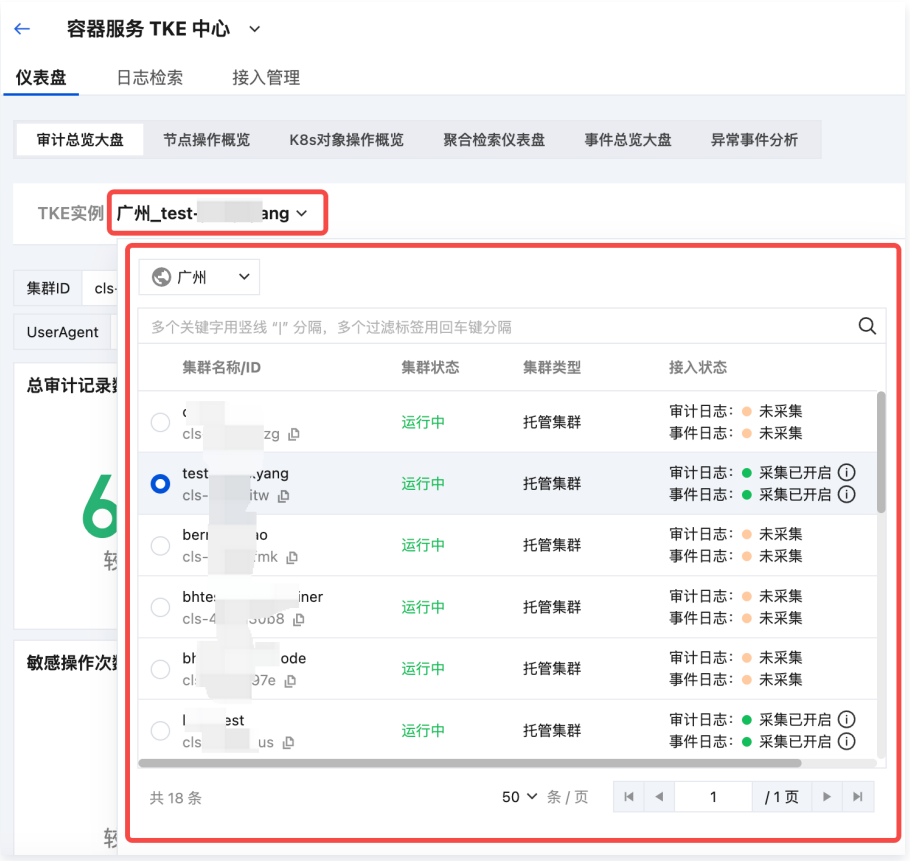
1. 完成开启 TKE 集群的审计或事件日志采集后，在顶部 Tab 栏中，选择**仪表盘**。



2. 在顶部二级 Tab 中，选择目标审计或事件仪表盘。



3. 在 TKE 集群下拉列表中，选择已开启审计或事件日志投递的目标 TKE 集群。



4. 选中集群后，即可查看与选中集群相关的事件或审计分析仪表盘。



升级日志采集组件

在**接入管理**页中，找到目标 TKE 集群，若采集组件状态显示**可升级**，可单击**升级**，将日志采集组件升级至最新版本。



卸载日志采集组件

在**接入管理**页中，找到目标 TKE 集群，单击操作中的**更多**，并在下拉框中单击**卸载日志采集组件**。



版权所有：腾讯云计算（北京）有限责任公司

DataSight 独立控制台
使用 DataSight 独立控制台

最近更新时间：2024-07-26 15:31:21

操作场景

适用于多人多团队共用 CLS 日志服务的场景。该场景下，不同的用户需要频繁登录腾讯云控制台检索分析日志，会碰到以下问题：

- **账号管理成本高**：公网访问控制台，需要运维人员频繁给申请人添加腾讯云账号，要求及时发现并清理离职人员的账号等。
- **功能访问路径深**：开发人员查看日志或运营人员查看仪表盘的使用过程中登录验证频繁，且每次都需要从云产品菜单开始，数次跳转才能打开日志服务。期望可以打开即用，避免频繁登录和跳转。

为解决以上问题，CLS 发布 DataSight 独立控制台，具备了以下特性：

- **降低账号管理成本**：
  - 用户不需要腾讯云账号，即可使用 CLS 日志服务。
  - 支持内网访问，安全性高，人员离职后即无法访问等。
  - 支持用户按角色授予不同的 CLS 日志服务访问权限。
  - 即使权限配置错误也只影响 CLS 日志服务，不会影响其他云产品，降低权限安全管控风险。
- **快捷的功能访问路径**：
  - 支持账号密码登录、免登录、单点登录等多种方式访问 CLS 日志服务。
  - 支持通过访问地址快速打开、分享、内嵌对应页面。

> **说明**：
> DataSight 当前可免费使用。

DataSight 体验环境
访问地址：DataSight 体验环境
免腾讯云账号使用日志服务的方式概览：

使用方式	功能支持	是否支持免腾讯云账号登录	开发工作量	账号权限管理
通过 DataSight 使用 CLS	支持 CLS 的日志检索、仪表盘、告警功能	支持，独立部署，免密登录或密码鉴权登录	无	无/密码鉴权，支持多角色
通过 Grafana 使用 CLS	只支持 Grafana 的仪表盘、告警功能	支持，数据接入 Grafana 使用	需要安装 CLS 插件	Grafana 的权限管理系统，支持多角色

其他旧方案：
请参见 iframe 内嵌腾讯云控制台 。

功能特性

降低账号管理成本

用户不需要腾讯云账号，即可使用 CLS 日志服务。

降低权限安全管控风险

- 支持用户按角色授予不同的 CLS 日志服务访问权限。
- 即使权限配置错误也只影响 CLS 日志服务，不会影响其他云产品。
- 支持内网访问。

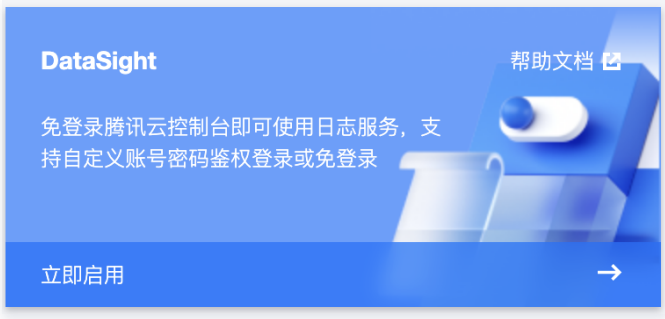
快捷的功能访问路径

- 支持账号密码登录、匿名登录、第三方认证登录等多种方式访问 CLS 日志服务。
- 支持通过访问地址快速打开、分享对应页面。

操作步骤

步骤一：创建 DataSight

1. 登录 日志服务 控制台，在**概览**页面，找到 DataSight，单击**立即启用**。



2. 在实例配置页面，选择**访问方式**。支持公网与内网两种访问方式。

访问方式	说明
公网访问	公网访问支持从互联网访问 DataSight 查看分析日志。仅支持密码鉴权登录。
内网访问	前提条件：用户办公网与腾讯云通过物理专线、VPN 安全隧道等方式互联互通。 内网访问支持从腾讯云内网访问 DataSight 查看分析日志，互联网无法访问，更安全。支持密码鉴权登录、匿名登录等多种登录方式。

步骤二：创建子用户获取 SecretId、SecretKey

1. 登录 访问管理 控制台，在**用户列表**页面，单击**新建用户**。
2. 选择**自定义创建**进入创建流程，再选择**可访问资源并接收消息**，单击**下一步**。
3. 配置子用户信息，并勾选**编程访问**，单击**下一步**。
4. 设置用户权限，选择**从策略列表中选取策略关联**。建议使用 CLS 只读访问权限策略 **QcloudCLSReadOnlyAccess** ，也可以根据需求自建策略，策略配置见 CLS 访问策略模板 。配置完成后单击**下一步**。



5. 配置用户标签后审阅角色信息，完成子用户创建。
6. 单击**显示密钥**，复制密钥信息。



步骤三：选择登录方式

密码鉴权登录

填写用户名、密码、以及用户权限对应的密钥后，保存配置。支持添加多个用户。

配置项	配置说明
用户名	登录 DataSight 的用户名
密码	登录 DataSight 的用户密码 ，密码要求8 ~ 30位字符，不能以" / "开头，至少包含：小写字母 a ~ z、大写字母 A ~ Z、数字0 ~ 9、()!@#$%^&*|?>< 中的三项。
SecretId、SecretKey	腾讯云的 SecretId、SecretKey，决定了该 DataSight 用户的权限。获取方式见 创建子用户获取 SecretId、SecretKey 。
访问 IP 限制	限制访问来源的 IP，支持输入 IP 地址或 CIDR，多个之间使用英文逗号分隔。 • 白名单：符合白名单的 IP 地址才能访问 DataSight。 • 黑名单：符合黑名单的 IP 地址将被拒绝访问 DataSight。

匿名登录

填入 SecretId、SecretKey 后保存配置。匿名登录仅内网访问支持，无需鉴权。建议使用只读权限的密钥，避免安全风险。

配置项	配置说明
SecretId、SecretKey	腾讯云的 SecretId、SecretKey，决定了匿名访问的权限。获取方式见 创建子用户获取 SecretId、SecretKey
访问 IP 限制	限制访问来源的 IP，支持输入 IP 地址或 CIDR，多个之间使用英文逗号分隔。 • 白名单：符合白名单的 IP 地址才能访问 DataSight。 • 黑名单：符合黑名单的 IP 地址将被拒绝访问 DataSight。

第三方认证登录

DataSight 支持反向代理处理身份认证后访问，实现对接第三方身份认证系统（LDAP、OAuth 等）。第三方认证登录仅内网访问支持，配置步骤如下：



1. 创建 DataSight 角色

填写角色名称及角色权限对应的密钥信息。角色将与第三方身份认证系统的用户角色进行映射。

配置项	配置说明
角色名	DataSight 的角色名称
SecretId、SecretKey	腾讯云的 SecretId、SecretKey 决定了该 DataSight 角色的权限。获取方式见 创建子用户获取 SecretId、SecretKey 。

2. 配置反向代理并设置用户与 DataSight 角色的映射

2.1 准备一个 Nginx 环境用作反向代理

该反向代理**必须可以访问 DataSight 所属的私有网络**，同时需确保已安装 --with-http_auth_request_module 模块，可使用 `nginx -V` 命令可查看是否已存在该模块。1.12版本以上的 Nginx 默认安装该模块，如果没有安装，请升级 Nginx 版本或参考 ngx_http_auth_request_module 进行安装。

2.2 登录机器并配置反向代理，配置模板可参见 腾讯云日志服务 CLS DataSight 登录校验代理 。目前提供 LDAP 与 Oauth 两种对接方式。

- **对接 LDAP**

  2.2.1 复制模板文件到本地

  ```
  git clone https://github.com/TencentCloud/cls-datasight-demo.git
  ```

  2.2.2 打开文件目录 `cls-datasight-demo/auth-proxy/ldap` 中的 nginx-ldap-auth.conf 文件，并按文档内的提示信息进行修改。

  2.2.3 将修改后的 `nginx-ldap-auth.conf` 文件复制到 Nginx 配置文件中，然后重启 Nginx 服务。

  ```
  cp nginx-ldap-auth.conf /etc/nginx/conf.d/nginx-ldap-auth.conf
  ## 请根据实际情况修改文件路径
  systemctl restart nginx
  ## 重启Nginx服务
  ```

2.2.4 打开文件目录 `cls-datasight-demo/auth-proxy/ldap` 中的 .env 文件，按照注释指引，生成并配置 ENCRYPTION_KEY 环境变量。

5.在目录 `auth-proxy/ldap` 下运行 LDAP 校验模块、登录模块。

a. (推荐) 通过 docker 运行：

```
docker build -f Dockerfile-auth -t nginx-ldap-auth .
docker build -f Dockerfile-login -t nginx-ldap-login .
docker run -d -p 8888:8888/tcp --env-file ".env" nginx-ldap-auth
docker run -d -p 9000:9000/tcp --env-file ".env" nginx-ldap-login
```

b. 本地直接通过 Python 运行。如遇到安装依赖报错，请根据报错提示搜索相关依赖包文档解决。

```
python3 -m pip install python-ldap python-dotenv setuptools_rust cryptography
nohup python3 nginx-ldap-login.py >login.log 2>&1 &
nohup python3 nginx-ldap-auth.py >auth.log 2>&1 &
```

- **对接 OAuth**

前提条件：本示例基于 Nginx + vouch-proxy 实现，需要已安装Nginx，并下载 vouch-proxy 到本地。您也可以根据实际情况使用其他 OAuth 校验软件，例如 oauth2-proxy 。

本示例使用 keycloak(通用 OIDC 协议) 作为 OAuth 提供商，如使用其他 OAuth 提供商请结合 vouch-proxy 官方指引，参考本示例修改使用。

2.2.1 复制模板文件到本地。

```
git clone https://github.com/TencentCloud/cls-datasight-demo.git
```

2.2.2 打开 `cls-datasight-demo/auth-proxy/oauth/keycloak(oidc)` 目录。其中 config.yml 是 vouch-proxy 配置文件，nginx-oauth-keycloak.conf 是 nginx 配置文件。

2.2.3 按文档内的提示信息修改 `nginx-oauth-keycloak.conf` 配置文件并复制到 Nginx 配置文件中，然后重启 Nginx 服务。

```
cp nginx-oauth-keycloak.conf /etc/nginx/conf.d/nginx-oauth-keycloak.conf
## 请根据实际情况修改文件路径
systemctl restart nginx
## 重启Nginx服务
```

2.2.4 按文档内的提示信息修改 `config.yml`，放到本地 vouch-proxy 的 config 文件夹中。

```
cp config.yml /root/vouch-proxy/config/config.yml
## 请按实际情况自行修改 vouch-proxy 的 config 文件路径
```

2.2.5 参考 vouch-proxy 指引运行 vouch 服务。推荐通过 docker 方式运行。

如果需要 vouch-proxy 使用本机网络访问 OAuth 服务，也可以在 `docker run` 时指定 --network=host。

```
docker run -d \
    -p 9090:9090 \
    -v ${PWD}/config:/config \
    quay.io/vouch/vouch-proxy
```

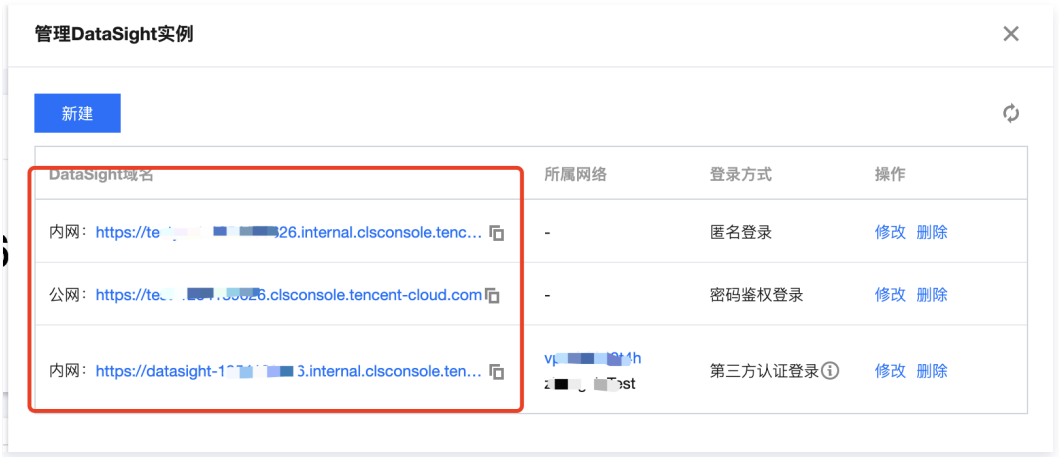
3. **填写反向代理的内网 IP**

填写 Nginx 内网 IP 地址或 CIDR。支持填写多个，使用英文逗号分隔。

步骤四：登录 DataSight 查看日志

- **密码鉴权登录或匿名登录**

在**云控制台概览页面**查看 DataSight 管理列表，获取访问地址。在浏览器打开访问地址，登录 DataSight 使用日志服务。如果为密码鉴权登录，则需要在登录页面输入用户名与密码。





- **第三方认证登录**

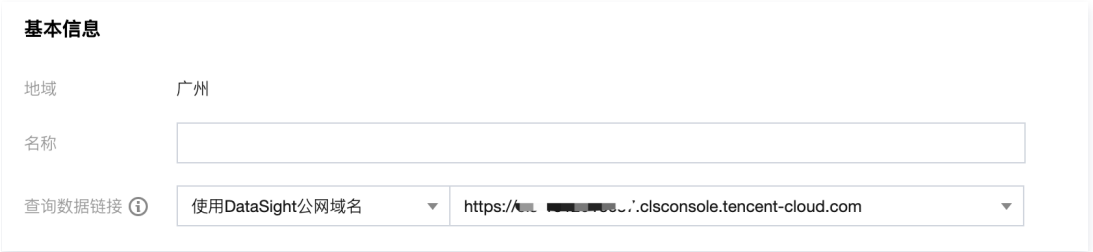
浏览器访问 Nginx 配置文件中配置的**自定义域名**，通过身份验证后登录 DataSight 控制台。

步骤五：让仪表盘订阅与告警通知的跳转链接默认打开 DataSight 控制台（可选）

1. 创建 订阅仪表盘，配置仪表盘链接为**使用 DataSight 域名**。保存后，发送的订阅仪表盘携带的链接将默认打开 DataSight。



2. 创建 告警通知渠道组，配置查询数据链接为**使用 DataSight 域名**。保存后，发送的告警通知携带的链接将默认打开 DataSight。



DataSight 场景案例

请参见 通过 DataSight 使用 CLS 。

DataSight 权限配置

请参见 DataSight 权限配置模板 。

配置控制台内嵌参数

最近更新时间：2024-09-18 22:02:41

版权所有：腾讯云计算（北京）有限责任公司

日志服务的腾讯云控制台与 DataSight 独立控制台的 URL 包含页面资源信息以及功能入口控制，支持打开指定页面资源或显示/隐藏对应功能入口。
相关内容：内嵌 DataSight 控制台

页面 URL 参数说明

检索分析页面

- 日志服务检索分析页面地址：

```
// 腾讯云控制台示例：
https://console.cloud.tencent.com/cls/search?region=<region>&topic_id=<topic_id>
```

```
// DataSight独立控制台示例，请根据实际情况修改 <domain-appid>（datasight域名前缀）部分：
https://<domain-appid>.clsconsole.tencent-cloud.com/cls/search?region=<region>&topic_id=<topic_id>
```

- 日志服务检索分析页面 URL 参数说明：

参数名称	必填	类型	描述
region	是	String	地域简称，例如：ap-shanghai 表示上海地域。其它可用地域简称参见 可用地域
topic_id	否	String	日志主题 ID
logset_name	否	String	日志集名称
topic_name	否	String	日志主题名称
multiple	否	String	指定为 true 时表示使用多主题检索功能；未指定时表示检索单个主题
topicType	否	String	指定为 metric 时表示查询指标主题；未指定时表示查询日志主题
time	否	String	要检索日志的起始时间至结束时间，格式样例：2021-07-15T10:00:00.000,2021-07-15T12:30:00.000
queryBase64	否	String	检索分析语句，采用 base64Url 编码。检索分析语法规则参见 语法规则
hideWidget	否	Boolean	隐藏右下角客服/文档按钮：默认不隐藏，true 表示隐藏
hideTopNav	否	Boolean	隐藏腾讯云控制台顶部菜单：默认不隐藏，true 表示隐藏
hideLeftNav	否	Boolean	隐藏腾讯云控制台左侧菜单：默认不隐藏，true 表示隐藏
hideTopicSelect	否	Boolean	隐藏主题选择列表：默认不隐藏，true 表示隐藏
collapseTopicSelect	否	Boolean	折叠主题选择列表：默认不折叠，true 表示折叠
hideTopTips	否	Boolean	隐藏顶部产品公告提示：默认不隐藏，true 表示隐藏
hideConfigMenu	否	Boolean	隐藏日志主题配置管理菜单：默认不隐藏，true 表示隐藏
hideDashboardMenu	否	Boolean	隐藏日志主题配置管理菜单中的仪表盘菜单：默认不隐藏，true 表示隐藏
hideAlertMenu	否	Boolean	隐藏日志主题配置管理菜单中的告警菜单：默认不隐藏，true 表示隐藏
hideCollectionMenu	否	Boolean	隐藏日志主题配置管理菜单中的采集配置菜单：默认不隐藏，true 表示隐藏
hideIndexMenu	否	Boolean	隐藏日志主题配置管理菜单中的索引配置菜单：默认不隐藏，true 表示隐藏
hideMoreMenu	否	Boolean	隐藏日志主题配置管理菜单中的更多菜单：默认不隐藏，true 表示隐藏
hideDashboard	否	Boolean	隐藏添加到仪表盘按钮：默认不隐藏，true 表示隐藏

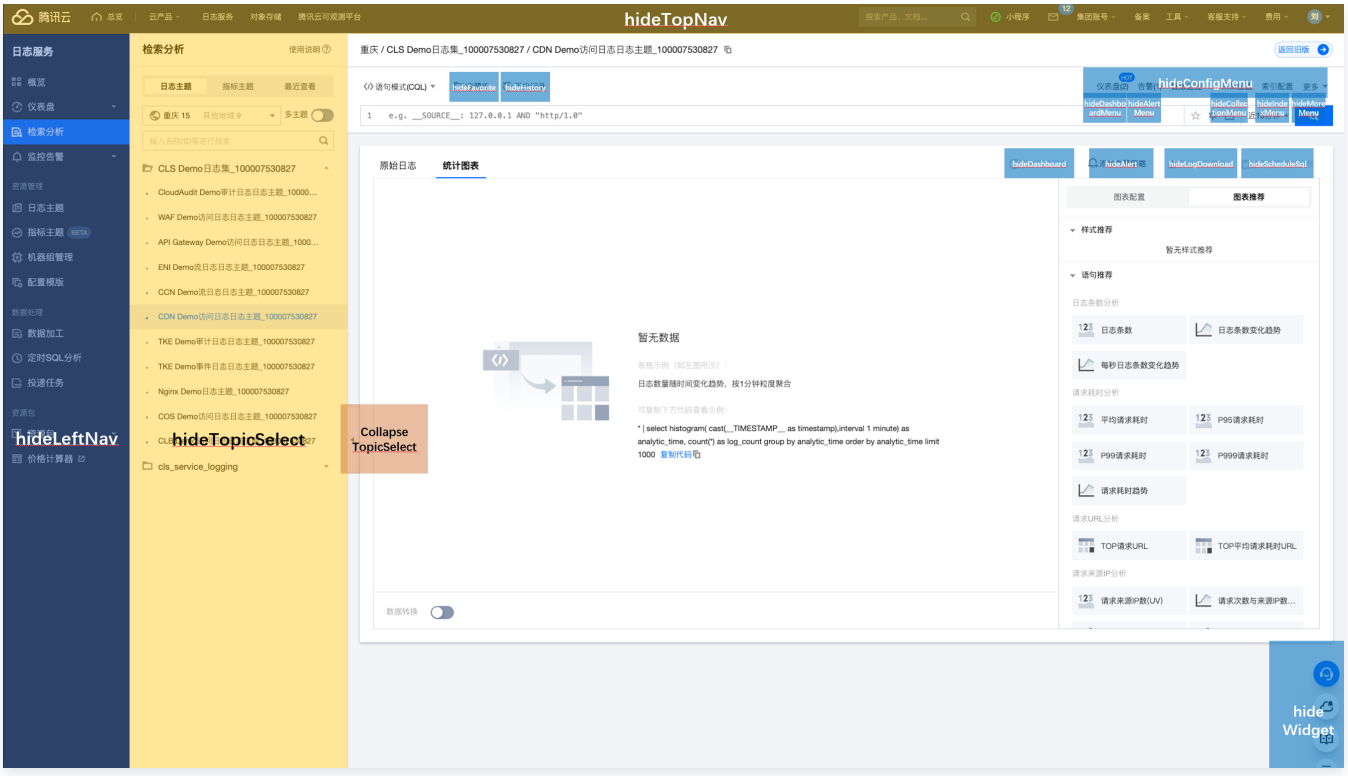
版权所有：腾讯云计算（北京）有限责任公司

hideAlert	否	Boolean	隐藏添加到告警按钮：默认不隐藏，true 表示隐藏
hideLogDownload	否	Boolean	隐藏原始日志下载按钮：默认不隐藏，true 表示隐藏
hideScheduleSql	否	Boolean	隐藏存为定时 SQL 分析按钮：默认不隐藏，true 表示隐藏
hideFavorite	否	Boolean	隐藏收藏夹按钮：默认不隐藏，true 表示隐藏
hideHistory	否	Boolean	隐藏历史记录按钮：默认不隐藏，true 表示隐藏

⚠ 注意：

- 可通过两种方式在 URL 参数中指定需要检索的日志主题：
  - 第一种 topic_id：根据日志主题 ID 指定日志主题。
  - 第二种 logset_name + topic_name：根据日志集名称及日志主题名称指定日志主题，需要注意的是当日志集或日志主题名称发生变化时，采用该方式的 URL 会失效。
  - 说明：如果 topic_id、logset_name + topic_name 同时存在，将以 topic_id 为准。
- 使用多主题检索功能时（multiple=true），最多可同时指定20个主题：
  - 使用 topic_id：用英文逗号连接多个主题 ID。
  - 使用 `logset_name + topic_name`：用英文逗号连接多个主题名称及日志集名称，主题名称需与所属日志集名称相互对应，例如 `topic_name=WAF 访问日志日志主题,CCN 流日志日志主题&logset_name=Demo 日志集,Demo 日志集。`

- 隐藏参数与页面模块对应关系：



仪表盘页面

- 日志服务仪表盘页面地址：

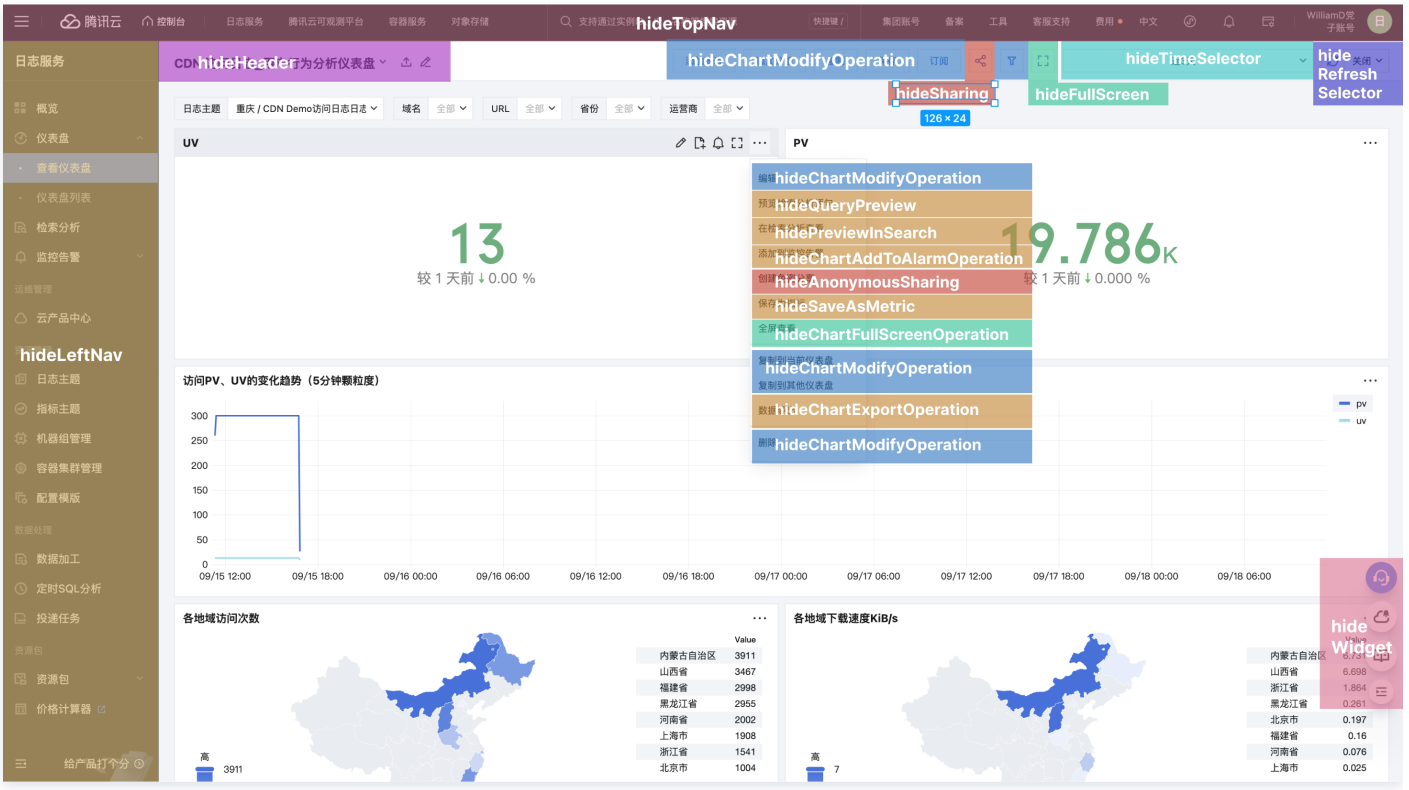
```
// 腾讯云控制台示例：
https://console.cloud.tencent.com/cls/dashboard/d?id=<dashboard_id>
```

```
// DataSight 独立控制台示例，请根据实际情况修改 <domain-appid>（datasight域名前缀）部分：
https://<domain-appid>.clsconsole.tencent-cloud.com/cls/dashboard/d?id=<dashboard_id>
```

- 日志服务仪表盘页面 URL 参数说明：

参数名称	必填	类型	描述
dashboard_id	是	String	仪表盘 ID
hideWidget	否	Boolean	隐藏右下角客服/文档按钮：默认不隐藏，true 表示隐藏
hideTopNav	否	Boolean	隐藏腾讯云控制台顶部菜单：默认不隐藏，true 表示隐藏
hideLeftNav	否	Boolean	隐藏腾讯云控制台左侧菜单：默认不隐藏，true 表示隐藏
hideHeader	否	Boolean	隐藏顶部 Header，包括 title 区域与返回按钮：默认不隐藏，true 表示隐藏
hideChartModifyOperation	否	Boolean	隐藏图表编辑操作入口（包含新建图表、复制图表、编辑图表、删除图表、保存按钮）
hideChartGridDrag	否	Boolean	关闭图表拖拽移动能力：默认不关闭，true 表示关闭
hideChartGridResize	否	Boolean	关闭图表修改尺寸能力：默认不关闭，true 表示关闭
hideTimeSelector	否	Boolean	隐藏时间选择器：默认不隐藏，true 表示隐藏
hideRefreshSelector	否	Boolean	隐藏自动刷新按钮：默认不隐藏，true 表示隐藏
hideChartFullScreenOperation	否	Boolean	隐藏图表全屏显示操作按钮：默认不隐藏，true 表示隐藏
hideChartExportOperation	否	Boolean	隐藏图表导出数据操作按钮：默认不隐藏，true 表示隐藏
hideChartAddToAlarmOperation	否	Boolean	隐藏图表添加到监控告警操作按钮：默认不隐藏，true 表示隐藏
hideQueryPreview	否	Boolean	隐藏图表预览检索分析语句操作按钮：默认不隐藏，true 表示隐藏
hidePreviewInSearch	否	Boolean	隐藏在检索分析查看操作按钮：默认不隐藏，true 表示隐藏
hideFullScreen	否	Boolean	隐藏仪表盘全屏查看操作按钮：默认不隐藏，true 表示隐藏
hideSaveAsMetric	否	Boolean	隐藏将图表数据保存为指标的操作按钮：默认不隐藏，true 表示隐藏
hideAnonymousSharing	否	Boolean	隐藏创建该图免密分享链接的操作按钮：默认不隐藏，true 表示隐藏
hideSharing	否	Boolean	隐藏获取仪表盘分享链接的操作按钮：默认不隐藏，true 表示隐藏

- 隐藏参数与页面模块对应关系：


hideTopNav
hideHeader
hideChartModifyOperation
hideSharing
126 × 24
hideTimeSelector
hideFullScreen
hide Refresh Selector
hideLeftNav
hideChartModifyOperation
hideQueryPreview
hidePreviewInSearch
hideChartAddToAlarmOperation
hideAnonymousSharing
hideSaveAsMetric
hideChartFullScreenOperation
hideChartModifyOperation
hideChartExportOperation
hideChartModifyOperation
hide Widget
UV
PV
13
较 1 天前 0.00 %
19.786K
较 1 天前 0.000 %
访问PV、UV的变化趋势（5分钟颗粒度）
各地域访问次数
各地域下载速度KiB/s

历史文档
低版本 LogListener 操作指南

最近更新时间：2024-10-16 16:03:12

> **注意**
> 本文档提供2.2.4以下版本的 LogListener 操作指南，后续可能不再维护，建议您更新到最新版本，最新版本安装操作详见 LogListener 安装指南 。

启动 LogListener

进入 `loglistener` 安装目录，通过以下脚本启动 LogListener：

```
cd loglistener/tools; ./start.sh
```

停止 LogListener

进入 `loglistener` 安装目录，通过以下脚本停止 LogListener：

```
cd loglistener/tools; ./stop.sh
```

查看 LogListener 进程状态

进入 `loglistener` 安装目录，通过以下命令查看 LogListener 进程：

```
cd loglistener/tools; ./p.sh
```



正常情况下，将存在以下三个进程：

```
bin/loglistenerm -d                              #守护进程
bin/loglistener --conf=etc/loglistener.conf      #主进程
bin/loglisteneru -u --conf=etc/loglistener.conf  #更新进程
```

卸载 LogListener

进入 `loglistener` 安装目录，通过以下命令卸载 LogListener：

```
cd loglistener/tools; ./uninstall.sh
```

检查 LogListener 心跳及配置

进入 `loglistener` 安装目录，通过以下命令卸载 LogListener：

```
cd loglistener/tools; ./check.sh
```

低版本 LogListener 异常状态排查

最近更新时间：2024-10-16 16:03:12

> **注意**
> 本文档提供2.2.4以下版本的 LogListener 故障处理排查。最新版本处理排查，请参见 机器组异常 处理文档。

现象描述

日志采集异常，检查关联的机器组，状态异常。

可能原因

机器组与日志服务系统之间的心跳中断，导致日志无法采集上报。导致机器组异常的可能原因有：

1. IP 地址填写错误。
2. 网络断开。
3. LogListener 进程失败。
4. LogListener 配置错误。

解决思路

针对以上可能原因，逐步进行排查。

处理步骤

1. 请检查核对机器组中所添加的 IP 地址是否正确。
1. 查看 LogListener 获取的 IP 地址，执行命令如下：

```
cd loglistener/tools && ./check.sh
```

```
[root@VM_30_69_centos tools]# ./check.sh
group ip:10.163.30.69
host:ap-chengdu.cls.myqcloud.com
port:80
```

2. 登录 日志服务控制台，单击**机器组管理**，查看机器组配置的 IP 地址，此处的 IP 地址必须与采集端获取的地址完全一致。



3. 请使用以下命令确认当前网络环境是否连通：

```
telnet <region>.cls.myqcloud.com 80
```

其中，`<region>` 为日志服务所在地域简称，具体地域信息请参见 可用地域 文档。
正常连通下，将显示如下代码。否则，显示连接失败，需排查网络环境，保证正常连网。

```
[                    ]# telnet ap-shanghai.cls.myqcloud.com 80
Trying                ...
Connected to ap-shanghai.cls.myqcloud.com.
Escape character is '^]'.
```

4. 请查看确认 LogListener 进程是否正常运行，进入安装目录查看命令如下：

```
cd loglistener/tools && ./p.sh
```

正常情况下，将存在以下三个进程：

```
bin/loglistenerm -d                              #守护进程
bin/loglistener --conf=etc/loglistener.conf      #主进程
bin/loglisteneru -u --conf=etc/loglistener.conf  #更新进程
```

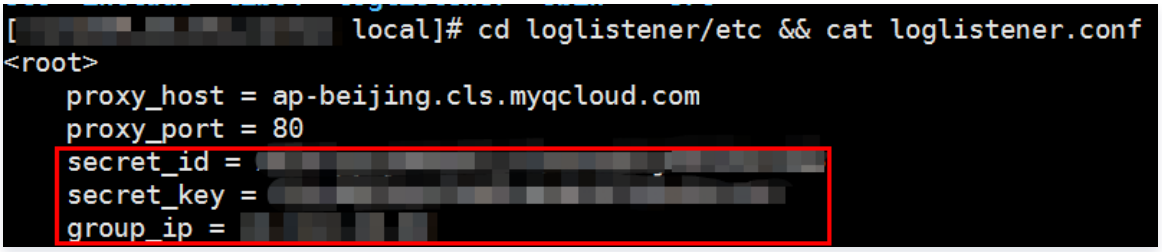
若有失败进程，需要重新启动，进入安装目录执行如下命令：

```
cd loglistener/tools && ./start.sh
```

5. 请检查确认 LogListener 中密钥及 IP 标识配置信息是否有误，进入安装目录查看配置信息，命令如下：

```
cd loglistener/etc && cat loglistener.conf
```

如图所示：



- 密钥为腾讯云账号或协作者的 API 密钥，但不支持项目密钥。
- 配置文件中 group_ip 信息必须与控制台机器组所填写的 IP 一致，LogListener 会自动获取机器 IP，当服务器有多个网卡时需留意检查是否一致。