# 日志服务（CLS）

## 产品文档



# 目录

日志服务（CLS） ..... 5
- 产品简介 ..... 5
  - 产品概述 ..... 5
  - 产品优势 ..... 7
  - 规格与限制 ..... 8
    - 基础资源 ..... 8
    - 日志采集 ..... 9
      - Kafka 协议上传 ..... 9
      - LogListener 限制说明 ..... 10
      - API日志上传限制 ..... 15
      - COS 数据导入规格限制 ..... 16
    - 检索分析 ..... 17
    - 监控告警 ..... 20
    - 投递与消费 ..... 22
  - 基本概念 ..... 24
    - 日志与日志组 ..... 24
    - 日志主题与日志集 ..... 27
    - 机器组 ..... 30
    - 分词与索引 ..... 32
    - 主题分区 ..... 34
- 快速入门 ..... 35
  - 入门指南 ..... 35
  - 日志上传API示例 ..... 40
- 操作指南 ..... 50
  - 资源管理 ..... 50
    - 日志集 ..... 50
    - 日志主题 ..... 52
    - 主题分区 ..... 54
      - 分裂主题分区 ..... 54
      - 合并主题分区 ..... 56
    - 机器组 ..... 57
  - 日志采集 ..... 64
    - 采集概述 ..... 64
    - LogListener采集 ..... 71
      - 采集机制 ..... 71
      - LogListener 使用流程 ..... 73
      - LogListener 安装和部署
      - LogListener 升级指南 ..... 74
        - LogListener 升级指南 ..... 74
        - Loglistener 2.4.6/2.8.2 升级到2.8.8 升级流程 ..... 77
      - LogListener 配置指南 ..... 81
      - LogListener 服务日志 ..... 84
      - LogListener安装指南 ..... 90
      - LogListener 采集配置导入 ..... 94
    - 采集文本日志 ..... 96
      - 单行全文模式 ..... 96
      - 多行全文模式 ..... 102
      - 单行-完全正则模式 ..... 108
      - 多行-完全正则模式 ..... 115
      - JSON 模式 ..... 123
      - 分隔符模式 ..... 129
      - 组合解析模式 ..... 135
      - 配置时间格式 ..... 142
    - 采集 Nginx 访问日志 ..... 145
    - 使用 Kafka 协议上传日志 ..... 151
    - 匿名日志采集 ..... 161
    - 使用API上传日志 ..... 164
    - 数据导入 ..... 170
      - 导入 COS 数据 ..... 170
  - 检索分析 ..... 175
    - 检索分析概述 ..... 175
    - 语法规则 ..... 177
    - 索引配置 ..... 180
    - 重建索引 ..... 186
    - SQL语法 ..... 188
      - SELECT 语法 ..... 188
      - AS 语法 ..... 190
      - GROUP BY 语法 ..... 191
      - ORDER BY 语法 ..... 193
      - LIMIT 语法 ..... 195
      - WHERE 语法 ..... 196
      - HAVING 语法 ..... 197
      - 嵌套子查询 ..... 198
      - JOIN 语法 ..... 200
      - UNION 语法 ..... 202
      - INTERSECT 语法 ..... 204
      - EXCEPT 语法 ..... 205
      - VALUES 语法 ..... 206
    - SQL函数 ..... 207
      - 函数概览 ..... 207
      - 通用聚合函数 ..... 221
      - 逻辑运算符 ..... 223
      - 算数及比较运算符 ..... 225
      - 估算函数 ..... 228
      - 同环比函数 ..... 230
      - 字符串函数 ..... 233
      - 日期和时间函数 ..... 239
      - IP函数 ..... 248
      - URL 函数 ..... 251

- JSON 函数 252
- 正则式函数 260
- 数组函数 266
- 类型转换函数 290
- 条件表达式 293
- 数学计算函数 298
- 数学统计函数 301
- 二进制字符串函数 302
- 位运算函数 303
- Lambda 函数 308
- 窗口函数 311
- 空间几何函数 315
- 快速分析 319
- 多主题检索 320
- 自定义跳转 322
- 上下文检索 324
- 仪表盘 327
- 仪表盘概述 327
- 创建仪表盘 329
- 添加图表 330
- 配置图表 333
- 表格 333
- 时序图 337
- 柱状图 343
- 单值图 346
- 计量仪 349
- 饼图 351
- 桑基图 353
- 地图 355
- 词云图 358
- 漏斗图 359
- 日志 361
- 交互事件 363
- 数据转换 365
- 单位配置 372
- 图表自定义时间 374
- 过滤与变量 375
- 投递与消费 380
- 投递与消费概述 380
- 投递任务角色授权 382
- 投递到对象存储 384
- 新建投递任务 384
- 投递日志原文 387
- 投递任务管理 389
- 配置告警 391
- 投递到 CKafka 392
- 创建投递任务 392
- 投递任务管理 396
- 配置告警 397
- Kafka 协议消费 398
- Kafka 协议消费 398
- 监控告警 401
- 监控告警简介 401
- 管理告警策略 403
- 配置告警策略 403
- 触发条件表达式 407
- 管理通知渠道组 410
- 告警通知变量 412
- 查看告警历史 436
- 常见问题 439
- 健康监测问题解释 439
- 日志上传相关 439
- 采集配置相关 440
- 索引配置相关 446
- 采集相关 447
- 机器组状态异常问题 447
- LogListener常见问题 453
- LogListener安装异常问题 457
- 采集配置常见问题 458
- 检索分析相关 463
- 检索不到日志问题 463
- 检索分析报错 465
- 统计分析(SQL)结果小数位数不正确 467
- 其他问题 469
- 词汇表 473
- API文档 474
- 日志服务（cls） 474
- 版本（2020-10-16） 474
- API 概览 474
- 调用方式 479
- 接口签名v1 479
- 接口签名v3 486
- 请求结构 495
- 返回结果 496
- 公共参数 499
- COS数据导入任务相关接口 501
- 预览cos导入信息 501
- Kafka协议消费相关接口 504
- 关闭Kafka协议消费 504
- 获取Kafka协议消费信息 506
- 修改Kafka协议消费信息 508

- 打开Kafka协议消费 510
- 主题分区相关接口 512
  - 获取分区列表 512
  - 合并分区 514
  - 分裂主题分区 516
- 仪表盘相关接口 518
  - 获取仪表盘 518
- 告警策略相关接口 521
  - 创建告警策略 521
  - 创建告警通知模板 525
  - 删除告警策略 528
  - 删除告警通知模板 530
  - 获取告警通知模板列表 532
  - 获取告警策略列表 534
  - 告警历史记录 536
  - 获取告警任务历史记录 538
  - 修改告警策略 542
  - 修改告警通知模板 546
- 对象存储数据导入任务相关接口 549
  - 创建对象存储导入任务 549
  - 获取对象存储导入配置 552
  - 修改对象存储导入任务 554
- 投递Ckafka任务相关接口 556
  - 创建投递Ckafka任务 556
- 投递对象存储任务相关接口 558
  - 创建投递规则 558
  - 删除投递规则 561
  - 获取投递任务列表 563
  - 获取投递规则 565
  - 修改投递规则 567
  - 重试失败的投递任务 570
- 日志主题相关接口 572
  - 创建日志主题 572
  - 删除日志主题 575
  - 获取日志主题列表 577
  - 修改日志主题 580
- 日志相关接口 583
  - 上下文检索 583
  - 获取直方图 586
  - 检索分析日志 589
- 日志集相关接口 592
  - 创建日志集 592
  - 删除日志集 594
  - 获取日志集列表 596
  - 修改日志集 598
- 机器组相关接口 600
  - 添加机器组信息 600
  - 创建机器组 602
  - 删除机器组 605
  - 删除机器组信息 607
  - 获取机器组列表 609
  - 获取机器状态 611
  - 修改机器组 614
- 索引相关接口 616
  - 创建索引 616
  - 删除索引配置 619
  - 获取索引配置信息 621
  - 修改索引 624
- 采集配置相关接口 626
  - 应用采集配置到指定机器组 626
  - 创建采集规则配置 628
  - 创建特殊采集配置任务 630
  - 删除采集规则配置 634
  - 删除应用到机器组的采集配置 636
  - 获取特殊采集配置 638
  - 获取采集规则配置所绑定的机器组 640
  - 获取采集规则配置 642
  - 获取机器组绑定的采集规则配置 644
  - 修改采集规则配置 646
- 数据结构 648
- 错误码 690

# 产品简介
# 产品概述

日志服务（Cloud Log Service）提供一站式的日志数据解决方案。您无需关注扩缩容等资源问题，五分钟快速便捷接入，即可享受从日志采集、日志存储到日志内容搜索等全方位稳定可靠的日志服务。帮助您轻松解决业务问题定位，指标监控、安全审计等日志问题，大大降低日志运维门槛。

## 功能概览

### 日志采集

日志服务目前支持 LogListener 、API、Kafka 协议、对象存储导入等多种采集方式：

- LogListener 实时采集：使用 LogListener 采集日志，便捷安装，服务稳定可靠安全、支持大部分主流 Linux 操作系统，兼顾高性能与低资源占用。
- 通过 API：无需安装 LogListener，直接调用 API 上传日志，支持多种语言。
- Kafka 协议：支持使用 Kafka Producer SDK 上传日志到 CLS。
- COS 导入：将对象存储中的数据导入到日志服务。

### 日志存储

根据用户对日志不同的检索时延性及日志处理能力需求，提供两种存储：

- 实时存储：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。
- 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。长期存储下，整体使用成本相比实时存储降低80%。

### 日志检索分析

针对存储在 CLS 的日志进行实时检索分析，帮助用户快速定位异常日志、统计系统及业务指标。

- 日志检索：使用关键词按全文或字段检索日志。
- 统计分析：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，兼容 SQL 92标准，支持200+ SQL 函数。例如按省份统计业务请求数，按时间查询请求错误率变化趋势。
- 性能优异：秒级返回查询结果，支持亿级别日志检索分析。

### 仪表盘

仪表盘提供数据分析全局视图，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。

- 仪表盘：仪表盘支持保存多个基于查询与分析结果的统计图表，构成场景化的分析大盘。

- 预置仪表盘：面向 CLB、TKE、COS 等产品，CLS 提供预置仪表盘，实现常见监控能力的开箱即用。
- 模板变量：模板变量支持数据源变量与快速过滤，实现仪表盘图表统计分析对象与维度的快速切换，千变万化支持更复杂的业务场景。

## 告警

日志内出现较多错误日志或系统及业务指标超出阈值时秒级告警，主动发现系统及业务异常问题。

- 通知渠道：支持电话、短信、邮件、企业微信和自定义接口回调。
- 多维分析：触发告警时可针对原始日志进行额外的检索分析，将结果附加在告警通知中，辅助定位告警原因。

## 日志数据处理

- 日志过滤清洗：按条件过滤日志。
- 日志结构化：将非结构化的日志处理成结构化数据，方便您后续使用 SQL 来计算分析日志。

## 日志投递与消费

您可以将日志投递至云产品中，如对象存储 COS、消息中间件 CKafka，也可以将 CLS 日志消费到其它大数据组件。

- 日志投递至 COS：用于日志归档、日志的离线计算。
- 日志投递至 CKafka：适用于日志的实时处理和计算。
- 日志实时消费：消费日志数据到大数据组件，例如 Flink、Logstash、Flume。

# 产品优势

## 功能丰富

- 提供采集、存储、检索、转存投递等功能一站式日志服务。
- 采集客户端 LogListener 提供单行/多行全文、分隔符、JSON、正则等日志结构化解析方式。
- 提供多种数据接入方式，用户可根据业务情况选择适合的接入方式，详情请参见 采集概述。
- 提供丰富的检索语法，方便用户进行关键词查询、模糊查询、范围查询等日志查询操作。

## 稳定可靠

- 日志服务采用高可扩展性的分布式存储架构，支持横向水平扩容，服务弹性伸缩，轻松存储管理海量日志数据。
- 日志服务后端存储采用多副本机制管理存储日志数据，为数据安全提供可靠性保障。

## 简单高效

- 采集端 LogListener 提供界面式的配置方式，配置简单直观，使用 LogListener 可快速接入日志服务。
- 数据写入 CLS 即可被消费，亿级数据查询支持秒级返回结果。
- 服务按实际用量收费，无需单独搭建和运维日志系统，避免了资源闲置浪费问题。

## 生态扩展

- 支持采集多个云产品的日志数据。
- 日志数据投递 COS，满足对日志数据长时间归档存储的需求。
- 日志数据投递 Ckafka，满足对日志数据实时消费的需求，便于进一步处理分析。

# 规格与限制

# 基础资源

## 配额限制

相关词汇及说明如下表所示：

| 相关词汇 | 说明 |
| --- | --- |
| 日志集 | 每个账号每个地域下最多可以创建100个日志集<br>每个日志集下最多可以创建500个日志主题及指标主题<br>名称长度1至255个字符，允许的字符为 a-z、A-Z、0-9、_、- |
| 日志主题 | 主题名称长度为1至255个字符，允许的字符为 a-z、A-Z、0-9、_、-<br>每个日志主题下最多50个分区，新增日志主题时最多能创建10个分区<br>每个主题最多可以绑定100个机器组<br>标准存储类型的日志主题，保存时间可为1 - 3600天，或永久保存<br>低频存储类型的日志主题，保存时间可为7 - 3600天，或永久保存<br>标准存储类型的日志主题，保存时间大于7天时，才能开启日志沉降功能 |
| 指标主题 | 主题名称长度为1至255个字符，允许的字符为 a-z、A-Z、0-9、_、- |
| 机器组 | 每个账号在不同地域下最多可以创建200个机器组<br>每个 IP 机器组最多可以绑定200个机器 IP<br>每个标识机器组最多可以绑定50个 Label |
| 仪表盘 | 每个账号最多创建200个仪表盘<br>单个仪表盘最多创建50个图表<br>每个账号最多创建100个仪表盘订阅任务 |

# 日志采集
# Kafka 协议上传

## 写入限制

日志写入限制及说明如下表所示：

| 限制 | 说明 |
| --- | --- |
| 上传日志 | 1. 单次上传请求的单条日志消息大小上限不超过1MB。<br>1. 单次上传请求的日志总和压缩前不能超过100MB。 |

# LogListener 限制说明

本文简介 Loglistener 采集数据时在文件采集、配置、资源、性能、错误处理等方面的能力与限制，以及相关使用说明。

## 文件采集限制

| 项目 | 能力与限制 |
| --- | --- |
| 文件编码 | 支持 UTF8，GBK 格式编码。<br>**注意：**<br>GBK 编码格式需要 Loglistener 2.8.8及以上版本。 |
| 软链接 | 支持。 |
| 单条日志大小 | 单行日志大小限制为512KB，若日志超过512KB 后，会截断只保留前512KB。多行日志按行首正则表达式划分后，单条日志最大限制为1MB。 |
| 正则表达式 | 正则表达式类型支持 Perl 兼容正则表达式。 |
| 首次日志采集行为 | Loglistener 支持全量采集/增量采集策略：<br>- 全量采集：首次安装启动 Loglistener 后，会采集所有符合条件的日志，包括已经没有写入的文件。<br>- 增量采集：首次安装启动 Loglistener 后，存量文件会从最新位置开始采集。<br>**注意：**<br>增量/全量采集需要 Loglistener 2.8.8及以上版本。 |
| 日志文件轮转 | 支持。（推荐轮转后的文件名，不要被采集通配路径覆盖到）<br>**注意：**<br>文件轮转后60秒内，若无新增日志写入轮转后的日志文件，轮转后的日志文件将不会再被采集，否则将持续采集轮转后的日志文件。 |
| 日志解析失败时采集行为 | 推荐开启解析失败上传功能，开启后将会把解析失败的日志按照单行全文格式上传至预设索引中。否则，解析失败日志将被丢弃。 |
| 文件打开行为 | Loglistener 会在采集读取文件时打开，读完关闭。 |
| 过滤规则 | 单、多行全文采集最多可配置1条过滤规则。 |
| | 分隔符、正则提取模式、JSON 格式最多可以配置5条过滤规则。 |
| Key 字段长度 | 分隔符、正则提取模式 Key 字段长度最多为128Byte。 |

| 项目 | 能力与限制 |
| --- | --- |
| Key-Value 组个数 | 所有解析模式下，提取 Key-Value 组的个数最多为1000个，超过部分将会被丢弃。 |
| LogListener 最大连接数 | LogListener 最大连接数限制：1024。 |
| 内存占用 | LogListener 的内存占用，空载情况最多100MB。 |
| CPU占用 | LogListener 的 CPU 使用在5MB/s日志量下，业务进程合计不超过单核20%。 |
| 采集配置生效延时 | 更新采集配置之后会有1分钟的生效延时。 |
| 日志文件 | LogListener 运行的采集目标日志文件大小建议小于500MB。 |

# Checkpoint 管理

| 项目 | 能力与限制 |
| --- | --- |
| Checkpoint 保存位置 | 保存路径默认为 Loglistener 安装目录下 data 目录下。若需修改，可参见 配置 LogListener，修改 LogListener 配置文件中的 checkpoint_cache_file，修改 Checkpoint 保存位置。 |
| Checkpoint 保存策略 | Loglistener 存有两份 checkpoint 元数据：<br>- 一份只记录已上传完成的位点信息，实时持久化到磁盘上。<br>- 一份同时记录了已完成位点信息与已采集但尚未完成的位点信息，周期性持久化到磁盘上。程序退出时也会优先进行持久化。 |

# 资源、性能限制

| 项目 | 能力与限制 |
| --- | --- |
| 默认 CPU 核数限制 | LogListener 仅支持单核运行。 |
| 默认 CPU 资源限制 | LogListener 默认未做 CPU 资源限制，若需限制，可参见 配置 LogListener，修改 LogListener 配置文件中的 cpu_usage_thres 最大 CPU 利用率。目前 Loglistener 的实现架构，如未作 CPU 资源限制，其最高能够达到的单核 CPU 使用率为110%左右（业务线程最大100%， 管控线程10%左右）。 |
| 默认线程数限制 | LogListener 2.x.x版本仅支持单线程运行。 |

| 项目 | 能力与限制 |
| --- | --- |
| 默认内存资源限制 | Loglistener 默认内存阈值设置为2G，若需限制，可参见 配置 LogListener，修改 LogListener 配置文件中的 max_mem 修改最大内存占用。建议不低于300M。 |
| 默认带宽资源限制 | Loglistener 默认未限制带宽资源，若需限制，可参见 配置 LogListener，修改 LogListener 配置文件中的 max_send_rate，对程序使用的网络带宽进行限制。 |
| 资源超限处理策略 | 若 Loglistener 占用相关 CPU 和内存资源超过最大限制的时间超过5分钟，则采集程序会强制自动重启。 |
| 日志压缩 | 采集日志默认会压缩后上传，若不需要压缩，可参见 配置 LogListener，修改 LogListener 配置文件中的 request_compression 关闭压缩上传。 |
| 监控目录数 | 默认最大监控目录数5000，如超出可能会引起采集失败。 |
| 监控文件数 | 推荐最大监控文件数10000，如超出可能会引起采集失败。 |
| 监控事件数 | Polling 模式下，推荐最大事件数10000，如超出可能会引起采集失败。 |

# 错误处理

| 项目 | 能力与限制 |
| --- | --- |
| 网络错误处理 | 非需要特殊处理的异常（如日志主题删除），其它错误都会进行重试（网络异常、超时、频控、欠费等）。 |
| 超时最大重试时间 | 若数据持续发送失败超过1小时，则丢弃该数据。<br>默认行为是间隔重试，且重试间隔时间会越来越长，直至超过超时最大尝试时间。 |
| 重试次数 | 可通过 loglistener.conf 配置文件设置，默认不配置。此时默认会一直重试，直至超过最大重试时间，之后丢弃。<br>如果配置了重试次数，则按照重试次数进行重试，超过最大次数则丢弃。 |
| 换行符缺失 | 采集时，若单条日志结尾缺少换行符， 采集器将等待五分钟， 若五分钟内没有检测到换行符，将自动填充换行符，并继续往下采集 |

# 文件采集规则

| 项目 | 能力与限制 |
| --- | --- |
| 日志上传策略 | Loglistener 会将同一文件的日志自动聚合上传，聚合条件为：10000条日志、日志集合总大小达到1M或者日志采集时间超过3秒，任一条件满足则触发上传行为。 |
| 文件采集的处理策略 | 单个目标文件（采集路径所能匹配到的一个文件）只支持上传日志到一个日志主题中，不支持多个 topic 的采集路径覆盖到同一个文件。如果想要将一个文件上传到多个日志主题中，可以通过软链接进行；对同一个目标文件创建多个软链接，不同的日志主题采集不同的软链接。 |
| 日志采集延迟 | 实时采集情况下，会在1分钟之内完成数据采集、传输、存储落盘，达到控制台可检索的效果。<br>如果日志生产量巨大，或者将采集程序使用的资源限制的较小，会有一定的采集延迟。 |

# 日志主题相关规则

| 项目 | 能力与限制 |
| --- | --- |
| 采集配置数量限制 | 一个日志主题可关联的采集配置数量上限为100个 |

# 机器组相关规则

| 项目 | 能力与限制 |
| --- | --- |
| 机器组逻辑 | 目前机器组分为两类，其使用方法是相互独立的，且两种用法是不兼容的，如果混合使用，采集机器将拉取不到正确的采集配置，造成不采集的现象。<br>- IP 机器组，机器 IP 需要在控制台上手动加入机器组，对应机器上 loglistener.conf 的 group_label 需为空。<br>- 标签机器组，控制台设置机器组标签，对应机器上 loglistener.conf 的 group_label 需要设置为相同的标签。 |
| 机器组与日志主题的关系 | - 单个日志主题，可以绑定多个机器组。<br>- 单个机器组，可以绑定到多个日志主题上。 |
| 机器组与采集机器的关系 | - 单个采集机器，可以加入到多个机器组中。<br>- 对于 IP 机器组，采集机器加入的机器组数量不受限制。<br>- 对于标签机器组，采集机器加入的机器组数量上限为20个。 |
| 标签机器组 label 限制 | - 标签机器组的 label，目前限制为32字符。<br>- 标签机器组，单个机器组最多可设置20个 label。 |

# 采集路径/采集黑名单用法

| 项目 | 能力与限制 |
| --- | --- |
| 采集黑名单 | 此功能是用来指定采集路径下，需要忽略采集的内容，目前采集黑名单分为两类：<br>- 文件路径模式：采集路径下，需要忽略采集的完整文件路径，支持通配模式。<br>- 目录路径模式：采集路径下，需要忽略采集的目录前缀，支持通配模式。<br>注意：<br>- FILE/PATH 模式可以同时使用。<br>- 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。 |

# API日志上传限制

## 写入限制

日志写入限制及说明如下表所示：

| 限制项 | 说明 |
| --- | --- |
| 上传日志 | 一个 pb 包里 logGroup 不能超过10个。<br>一个 pb 包里 logGroup 包含日志条数最多为10000条，至少包含1条。<br>log 中单个 value 最大为1MB。<br>一个 pb 包里 logGroup 部分所有 Value 大小最大总和为5MB。<br>Log 中的 Key 不能以 _ 开头，且不能是以下保留字<br>段："__FILENAME__"、"__TIMESTAMP__"、"__LOGSETID__"、"__TOPICID__"、"__SOURCE__"、"__HOSTNAME__"、"__CONTENT__"。<br>单次上传请求的包体压缩前不能超过6MB。 |
| 频控 | 单个主题分区写请求限制：500QPS。超限返回状态码429，提示错误 SpeedQuotaExceed 。写频率较大时，建议分裂主题分区。<br>说明：<br>此处500QPS不代表500条日志。 一次请求中的 pb 包可以包含多条日志批量上传。 |
| 流控 | 单个主题分区压缩后写流量限制：5MB/s。超限返回状态码429，提示错误 SpeedQuotaExceed。写流量较大时，建议 分裂主题分区。 |

# COS 数据导入规格限制

## 控制台限制

| 限制项 | 说明 |
| --- | --- |
| 文件预览限制 | 仅支持预览小于1G的文件 |

## 导入限制

| 限制项 | 说明 |
| --- | --- |
| 单个文件大小 | 单个文件大小不可超过1G。若超过限制，导入任务将忽略整个文件。 |
| 单条日志大小 | 单行日志大小限制为512KB，若日志超过512KB 后，会截断只保留前512KB。<br>多行日志按行首正则表达式划分后，单条日志最大限制为1M。 |
| 正则表达式 | 正则表达式类型支持 Perl 兼容正则表达式。 |
| 日志解析失败时采集行为 | 推荐开启解析失败上传功能，开启后将会把解析失败的日志按照单行全文格式上传至预设索引中。否则，解析失败日志将被丢弃。 |
| Key 字段长度 | 分隔符、正则提取模式 Key 字段长度最多为128Byte。 |
| Key-Value 组个数 | 所有解析模式下，提取 Key-Value 组的个数最多为1000个，超过部分将会被丢弃。 |

# 检索分析

## 日志主题

日志主题检索分析规格限制如下：

### 索引

| 限制项 | 说明 |
| --- | --- |
| 字段数量 | 单个日志主题键值索引最多可添加300个字段 |
| 字段名称 | - 仅支持字母、数字及 _-./@ ，且不能以 _ 开头( __CONTENT__ 字段例外)<br>- 不允许同时包含 JSON 父子级字段，例如 a 及 a.b |
| 字段层级 | 为多层级 JSON 配置键值索引时，Key 的层级不能超过10层，例如 a.b.c.d.e.f.g.h.j.k |
| 分词符 | 仅支持英文符号、\n\t\r及转义符\| |
| 字段长度 | 字段开启统计后，仅前32766个字节参与 SQL 运算，超出部分无法执行 SQL，但仍会完整存储日志 |
| 分词长度 | 分词后，单个词仅前10000个字符参与检索，超出部分无法执行检索，但仍会完整存储日志 |
| 数值字段精度及范围 | - long 类型字段支持数据范围为-1E15 ~ 1E15，超出该范围的数据可能会丢失精度或不支持检索<br>- double 类型字段支持数据范围为-1.79E+308 ~ +1.79E+308，如果浮点数编码位数超过64位，会造成精度丢失<br>超长数值字段索引配置建议：<br>- 如果不需要通过数值范围比较来检索该字段，可将该字段存储为 text 类型<br>- 如果需要通过数值范围比较来检索该字段，可将该字段存储为 double 类型，可能会丢失部分精度 |
| 生效机制 | 索引配置仅针对新采集的数据生效，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。如需更新已有数据，需重建索引 |
| 修改索引配置 | 创建、修改及删除索引配置时，单个用户最大同时拥有10个执行中的任务，超出后需等待之前的任务执行完毕，单个任务执行一般不超过1分钟 |
| 重建索引 | - 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<br>- 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<br>- 仅支持通过控制台重建写流量500GB以内的时间范围的日志，超过该限制时建议您缩小需要重建索引的日志时间范围<br>- 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 |

| 限制项 | 说明 |
| --- | --- |
| | 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 |

## 查询

| 限制项 | 说明 |
| --- | --- |
| 语句长度 | 检索分析语句最长支持12000字符 |
| 查询并发 | 单个日志主题支持15并发，包括检索及分析，告警和仪表盘 |
| 模糊检索 | 不支持前缀模糊检索，例如通过`*rror`检索 error |
| 短语检索 | 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 |
| 逻辑分组嵌套深度 | CQL 语法规则下使用括号对检索条件进行逻辑分组时，最多可嵌套10层，Lucene 语法规则无该限制<br>例如`(level:ERROR AND pid:1234) AND service:test`为2层嵌套，可正常执行检索；而下面的语句为11层嵌套，执行检索将会报错`status:"499" AND ("0.000" AND (request_length:"528" AND ("https" AND (url:"/api" AND (version:"HTTP/1.1" AND ("2021" AND ("0" AND (upstream_addr:"169.254.128.14" AND (method:"GET" AND (remote_addr:"114.86.92.100"))))))))))` |
| 内存占用量（分析） | - 统计分析每次所占用的服务端内存不能超过3GB<br>- 通常在使用 group by、distinct()、count(distinct()) 时可能触发该限制，是由于被统计的字段在通过 group by 或 distinct() 去重后值过多导致的。建议优化查询语句，使用值更少的字段对数据进行分组统计；或使用 approx_distinct() 替代 count(distinct()) |
| 查询结果 | 查询结果为原始日志时，单次最大返回1000条原始日志，API 中通过游标 Context 翻页最多可以获取10000条原始日志 |
| | 查询结果为统计分析结果时，单次默认返回100条结果，使用 SQL LIMIT 语法 单次可最大返回1万条结果 |
| | 查询结果返回数据包最大49MB，使用 API 时可启用 gzip 压缩（Header Accept-Encoding:gzip） |
| 超时时间 | 单次查询超时时间为30秒，包括检索及分析 |
| 查询延迟 | 从日志上报到可被检索分析的延迟小于1分钟 |

## 下载

| 限制项 | 说明 |
| --- | --- |

| 限制项 | 说明 |
| --- | --- |
| 日志条数 | 后端单次下载日志条数最多为5000万条，依赖后端服务Exporter的部署；前端单次下载日志条数最多为1万条。 |
| 任务数量 | - 单个日志主题最多2个任务处于“文件生成中”状态，其它未完成的任务处于“等待中”状态，排队等待执行<br>- 单个日志主题最多同时存在1000个任务，包含处于“文件已生成”状态的已完成的任务 |
| 文件保存时长 | 生成的日志文件仅保留3天 |

# 监控告警

监控告警规格限制如下：

## 告警策略

| 限制项 | 说明 |
| --- | --- |
| 监控对象 | 单个告警策略最多可选择20个日志主题，不支持跨地域。 |
| 执行语句 | 限制同 检索分析，包括语句长度、并发限制等。 |
| 执行语句数量 | - 各个执行语句使用相同日志主题时，单个告警策略最多可添加3个执行语句。<br>- 各个执行语句单独选择日志主题时，单个告警策略最多可添加10个执行语句。 |
| 执行语句查询时间范围 | 最大时间范围为最近24小时。 |
| 执行周期 | 最小执行周期为1分钟，最大执行周期为1440分钟。 |
| 触发条件判断次数 | 当执行语句结果为多个值时（分组后），会根据结果数据依次判断是否满足触发条件，当满足触发条件或超过1000次后停止判断。 |

## 告警渠道

| 限制项 | 说明 |
| --- | --- |
| 电话告警 | - 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。<br>- 同一号码22:00至次日08:00最多接收3条电话告警。<br>说明：<br>以上限制不局限于日志服务产生的电话告警通知，消息中心产生的电话通知会共享该限制。 |
| 企业微信 | 每个机器人发送的消息不能超过20条/分钟。 |
| 自定义接口回调 | - 支持公网访问，不支持内网地址。<br>- 接口返回状态码为200时表示回调成功。 |

## 告警历史

| 限制项 | 说明 |
| --- | --- |
| 存储时长 | 30天 |

# 投递与消费

相关限制说明如下：

## 日志主题

| 限制项 | 说明 |
| --- | --- |
| 频控 | 200QPS每分区。分区指的是源日志主题的分区。超限返回状态码429，提示错误SpeedQuotaExceed 。 |

## 投递COS

| 限制项 | 说明 |
| --- | --- |
| 投递任务数 | 每个日志主题投递 COS 任务配置个数最多为10个。 |
| 可投递的数据范围 | 日志主题的生命周期内的数据（包含历史数据和实时数据）。 |
| 投递任务时延 | 投递 COS 的时延在60秒左右。<br>说明：假如您配置投递时间间隔为5分钟，那么12:00-12:05的日志，满足了5分钟间隔的条件，这部分日志会作为一个批次，整体投递到 COS，时延为60秒，那么这个批次的日志将在12:06左右到达 COS。 |
| 投递触发 | 单个分区满足以下条件任意一个时，触发一次投递：<br>- 投递日志大小（累计日志数据），您可配置该参数，5-250MB（未压缩）。<br>- 投递时间间隔，5-15分钟，您可配置该参数，5-15分钟。 |
| 跨账号/跨地域 | 暂不支持跨账号/跨地域的投递任务。 |

## 投递CKafka

| 限制项 | 说明 |
| --- | --- |
| 投递任务数 | 每个日志主题可配置一个 CKafka 投递任务。 |
| 可投递的数据范围 | 实时日志数据，暂不能投递历史数据。 |

| 限制项 | 说明 |
| --- | --- |
| 投递任务时延 | 99.9%的日志投递 CKafka 的时延在5秒内。（目标 CKafka 状态正常，且吞吐能力足够处理来自 CLS 的消息，不存在磁盘写满或者限流的情况）。<br>说明：某一条日志采集到 CLS 的时间戳（_TIMESTAMP_字段的值）为12:00:00，该条日志将在12:00:05秒之前到达 CKafka。 |
| 跨账号/跨地域 | 暂不支持跨账号/跨地域的投递任务。 |

# Kafka 协议消费

| 限制项 | 说明 |
| --- | --- |
| 版本 | 支持 Kafka 协议版本为：1.1.1及更早的版本。 |
| 消费组个数 | 每个日志主题可配置20个消费组。 |
| 认证方式 | 仅支持 SASL_PLAINTEXT。 |
| 可消费的数据范围 | 开启该功能后，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。<br>说明：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，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。 |
| 消费时延 | 99.9%的日志消费的时延在5秒内。<br>说明：某一条日志采集到 CLS 的时间戳（_TIMESTAMP_字段的值）为12:00:00，该条日志将在12:00:05秒之前到达消费端。 |
| 跨账号/跨地域 | 不支持跨账号消费，用户可通过外网消费实现跨地域消费。 |

# 基本概念
# 日志与日志组

## 日志

日志（Log）是应用系统运行过程中产生的记录数据，如用户操作日志、接口访问日志、系统错误日志等。日志通常以文本的形式存储在应用系统所在的机器上，一条系统运行记录对应的日志可能为一行文本（单行日志），也可能为多行文本（多行日志）。

单行日志示例：

```
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" "-"
```

多行日志示例：

```
java.net.SocketTimeoutException:Receive timed out
  at j.n.PlainDatagramSocketImpl.receive0(Native Method)[na:1.8.0_151]
  at j.n.AbstractPlainDatagramSocketImpl.receive(AbstractPlainDatagramSocketImpl.java:143)[^]
  at j.n.DatagramSocket.receive(DatagramSocket.java:812)[^]
  at o.s.n.SntpClient.requestTime(SntpClient.java:213)[classes/]
  at o.s.n.SntpClient$1.call(^:145)[^]
  at ^.call(^:134)[^]
  at o.s.f.SyncRetryExecutor.call(SyncRetryExecutor.java:124)[^]
  at o.s.f.RetryPolicy.call(RetryPolicy.java:105)[^]
  at o.s.f.SyncRetryExecutor.call(SyncRetryExecutor.java:59)[^]
  at o.s.n.SntpClient.requestTimeHA(SntpClient.java:134)[^]
  at ^.requestTimeHA(^:122)[^]
  at o.s.n.SntpClientTest.test2h(SntpClientTest.java:89)[test-classes/]
  at s.r.NativeMethodAccessorImpl.invoke0(Native Method)[na:1.8.0_151]
```

对于一条日志，其主要组成部分如下：

- 内置保留字段

| 组成部分 | 含义 | 示例 |
| --- | --- | --- |

| 组成部分 | 含义 | 示例 |
| --- | --- | --- |
| 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，在控制台显示为“日志时间” | 1640005601188 |
| FILENAME | 日志采集的文件名 | /data/log/nginx/access.log |
| SOURCE | 日志采集的源 IP | 10.0.1.2 |
| HOSTNAME | 日志来源机器名称，需使用2.7.4及以上版本的 Loglistener 才会采集该字段 | localhost |
| RAWLOG | 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，可能会导致索引创建异常，此时 CLS 为了确保日志不丢失，会将原始日志存储在该字段中，作为兜底的异常处理方式 | {"ip":"10.20.20.10","request":"GET /online/sample HTTP/1.1","status":200,"time":"[2018-07-16 13:12:57]"} |
| INDEX_STATUS | 日志创建索引异常原因，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__为异常原因 | IndexPartFail:part of fields do not match the index configuration |

- 日志正文

| 组成部分 | 含义 | 示例 |
| --- | --- | --- |
| CONTENT | 日志提取模式为单行全文或多行全文（即原始日志不经切分，直接将整条日志上报）时，整条日志存储在该字段中 | 10.20.20.10;[2018-07-16 13:12:57];GET /online/sample HTTP/1.1;200 |
| 普通字段 | 日志采集时对日志进行结构化后，原始日志以 key:value 的形式存储 | IP: 10.20.20.10<br>request: GET /online/sample HTTP/1.1<br>status: 200<br>time: [2018-07-16 13:12:57] |

- 元数据

| 组成部分 | 含义 | 示例 |
| --- | --- | --- |

| 组成部分 | 含义 | 示例 |
| --- | --- | --- |
| 元数据字段 | 对日志的简单描述或归类，例如 TKE 日志中，某条日志所属的集群或容器，以 key:value 的形式存储，key 以__TAG__.开头 | TAG.clusterId:1skzv59c |

# 日志组

日志组（LogGroup）是一个包含多条日志的集合。在上传日志的过程中，为提高数据读写效率，将多条日志打包成一个日志组，并以日志组为单位上传到 CLS。

一个日志组里的日志具有相同的基本信息（TIMESTAMP、FILENAME、SOURCE、HOSTNAME 和元数据等）。

# 日志主题与日志集

## 日志主题

日志主题（Topic）是日志数据在日志服务（Cloud Log Service，CLS）平台进行采集、存储、检索和分析的基本单元，采集到的海量日志以日志主题为单元进行管理，包括采集规则配置、保存时间配置、日志检索分析以及日志下载/消费/投递等。

一个日志主题通常对应某一个应用/服务，建议将同一个应用/服务在不同机器上的同类日志采集到同一个日志主题。例如，某支付服务（payService）部署在数十台机器上，包含访问日志（access_log）和错误日志（error_log）两类日志。可创建 payService_access_log_topic 和 payService_error_log_topic 两个日志主题，分别对应这数十台机器上的两类日志，通过这两个日志主题即可完成数十台机器上的所有日志的集中检索和分析。

日志主题与应用/服务之间并非严格的一一对应关系，如果两个服务之间的日志结构相似度较高，且经常需要集中分析日志，也可以将这两个服务的日志上报至同一日志主题下。

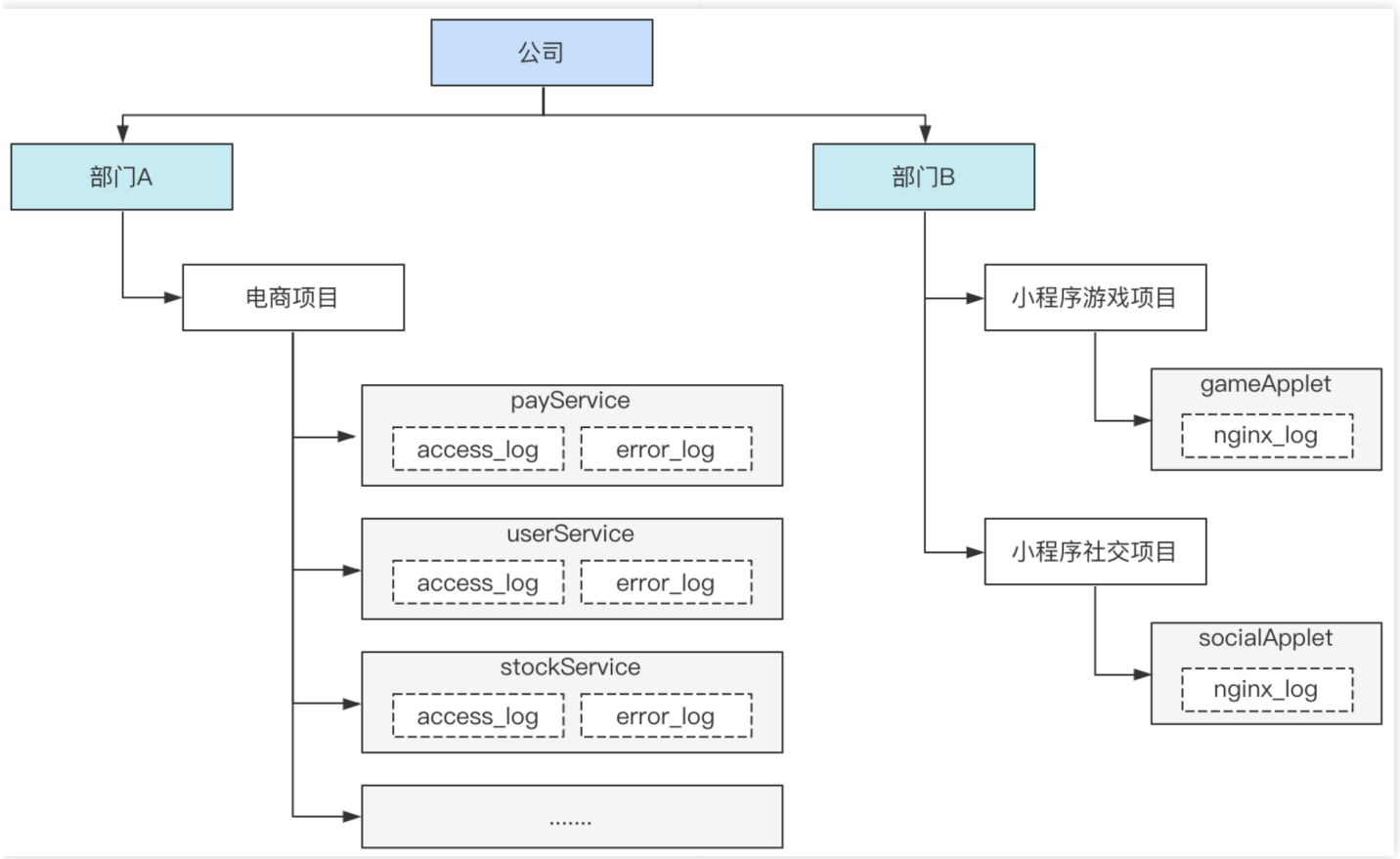
## 日志集

日志集（Logset）是对日志主题的分类，一个日志集可包含多个日志主题。日志集本身不存储任何日志数据，仅方便用户管理日志主题。

一个日志集通常对应公司内的某一个项目/业务，建议将某个项目/业务下的多个应用/服务的日志主题归属到同一个日志集下。例如，公司某电商项目下包含多个服务（支付服务 payService、用户服务 userService、库存管理服务 stockService 等），可创建一个日志集 e_commerce_logset，将这些服务的日志主题均归属至该日志集下。这样当公司有多个项目时，具体的项目人员只需要查看所属项目对应的日志集下的日志主题即可，其它项目的日志主题不会对其产生干扰。

> 注意：
>
> 新建主题时可指定其归属的日志集，保存后不能再变更。

## 场景示例




如上图，该公司有两个部门：

- 部门 A 有一个电商项目，采用微服务架构，每个服务均包含访问日志（access_log）和错误日志（error_log）两类日志。
- 部门 B 有两个项目，分别为小程序游戏项目和小程序社交项目，技术架构比较简单，均各有一个 Nginx 的访问日志（nginx_log）。

使用 CLS 监控上述这些应用日志时，可创建如下的日志集及日志主题：

| 日志集 | 日志主题 | 标签 |
| --- | --- | --- |
| e_commerce_logset | payService_access_log_topic | 部门:部门A |
| e_commerce_logset | payService_error_log_topic | 部门:部门A |
| e_commerce_logset | userService_access_log_topic | 部门:部门A |
| e_commerce_logset | userService_error_log_topic | 部门:部门A |
| e_commerce_logset | stockService_access_log_topic | 部门:部门A |
| e_commerce_logset | stockService_error_log_topic | 部门:部门A |
| e_commerce_logset | ..... | 部门:部门A |

| 日志集 | 日志主题 | 标签 |
| --- | --- | --- |
| gameApplet_logset | gameApplet_nginx_log_topic | 部门:部门B |
| socialApplet_logset | socialApplet_nginx_log_topic | 部门:部门B |

其中的标签用来区分日志集及日志主题归属的部门，结合权限策略还可以控制每个部门的人员仅可查看所属部门的数据。
对于部门 A，e_commerce_logset 日志集涵盖了其电商业务下的所有服务的日志主题，后续如果新增其他的项目，新建一个日志集即可。
对于部门 B，虽然目前的技术架构比较简单，总共只有两类日志，但却创建了两个日志集，每个日志集只有一个日志主题，是出于如下目的：

- 支持后续架构扩展：如果业务规模上升，技术架构也演变为微服务架构后，可继续沿用当前的日志集，在当前日志集下新增日志主题即可，不同的项目之间互不影响
- 灵活应对项目调整：如果把整个部门作为一个日志集，当其中的某个项目需要独立为一个部门，或需调整至另一个部门时，由于日志主题不能直接变更其归属的日志集，调整将变得非常麻烦，可能需要重新采集日志。而每个项目分别对应一个日志集时，则不存在该情况，只需调整日志集和日志主题对应的标签即可。

# 机器组

## 机器组

机器组（MachineGroup）是一组需要采集日志的机器列表，日志服务通过机器组来管理所有需要通过loglistener采集日志的机器。

一个机器组可以包含多台机器。当应用/服务部署在多台机器上，且这些机器的日志文件路径相同时，便可将其归为一个机器组。这样在控制台只需要配置一次日志数据采集规则，便可批量生效至机器组内的所有机器。

机器组可关联至一个日志主题，即机器组内所有的日志均上报至同一个日志主题；也可以关联至多个日志主题，即机器组内不同路径的日志分别上报至不同的日志主题。

机器组具备两种定义方式：

- IP 地址：添加 IP 地址列表到机器组，机器组将自动添加这些IP对应的机器。
- 标识（Label）：安装loglistener时为所在机器添加标识，机器组将自动添加包含这些标识的机器。

## 场景示例

某电商项目，共有6台机器及三个服务。其部署方式如下表：
payService 部署在2台机器上，有2个日志文件路径。userService和stockService 混部在4台机器上，共有4个日志文件路径。每个服务的每类日志（访问日志 access_log 和错误日志 error_log）均需上报至单独的日志主题。

| 机器 | 部署服务 | 日志文件路径 |
| --- | --- | --- |
| 192.168.1.1 | payService | /data/log/payService/access_log.log<br>/data/log/payService/error_log.log |
| 192.168.1.2 | payService | /data/log/payService/access_log.log<br>/data/log/payService/error_log.log |
| 192.168.1.3 | userService,stockService | /data/log/userService/access_log.log<br>/data/log/userService/error_log.log<br>/data/log/stockService/access_log.log<br>/data/log/stockService/error_log.log |
| 192.168.1.4 | userService,stockService | /data/log/userService/access_log.log<br>/data/log/userService/error_log.log<br>/data/log/stockService/access_log.log<br>/data/log/stockService/error_log.log |
| 192.168.1.5 | userService,stockService | /data/log/userService/access_log.log<br>/data/log/userService/error_log.log<br>/data/log/stockService/access_log.log<br>/data/log/stockService/error_log.log |

| 机器 | 部署服务 | 日志文件路径 |
| --- | --- | --- |
| 192.168.1.6 | userService,stockService | /data/log/userService/access_log.log<br>/data/log/userService/error_log.log<br>/data/log/stockService/access_log.log<br>/data/log/stockService/error_log.log |

在机器上部署 LogListener 时，可为每台机器按其上运行的服务添加标识，具体如下：

| 机器 | 部署服务 | 标识 |
| --- | --- | --- |
| 192.168.1.1 | payService | payService |
| 192.168.1.2 | payService | payService |
| 192.168.1.3 | userService,stockService | userService,stockService |
| 192.168.1.4 | userService,stockService | userService,stockService |
| 192.168.1.5 | userService,stockService | userService,stockService |
| 192.168.1.6 | userService,stockService | userService,stockService |

然后在控制台创建3个机器组，采用标识方式进行定义，分别为 payService、userService 和 stockService，再在日志主题中关联该机器组并添加对应的采集配置即可完成日志上报。

| 日志主题 | 关联机器组 | 采集路径 |
| --- | --- | --- |
| payService_access_log_topic | payService | /data/log/payService/access_log.log |
| payService_error_log_topic | payService | /data/log/payService/error_log.log |
| userService_access_log_topic | userService | /data/log/userService/access_log.log |
| userService_error_log_topic | userService | /data/log/userService/error_log.log |
| stockService_access_log_topic | stockService | /data/log/stockService/access_log.log |
| stockService_error_log_topic | stockService | /data/log/stockService/error_log.log |

后续服务需要扩容时，只需要在新增的机器上添加部署的服务作为标识，即可自动将新增的机器添加至对应的机器组并采集日志，极大的提升运维部署效率。

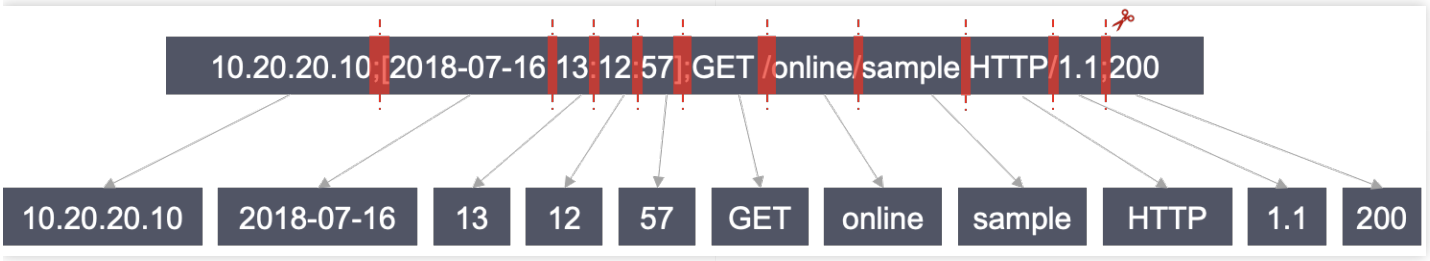
# 分词与索引

## 分词

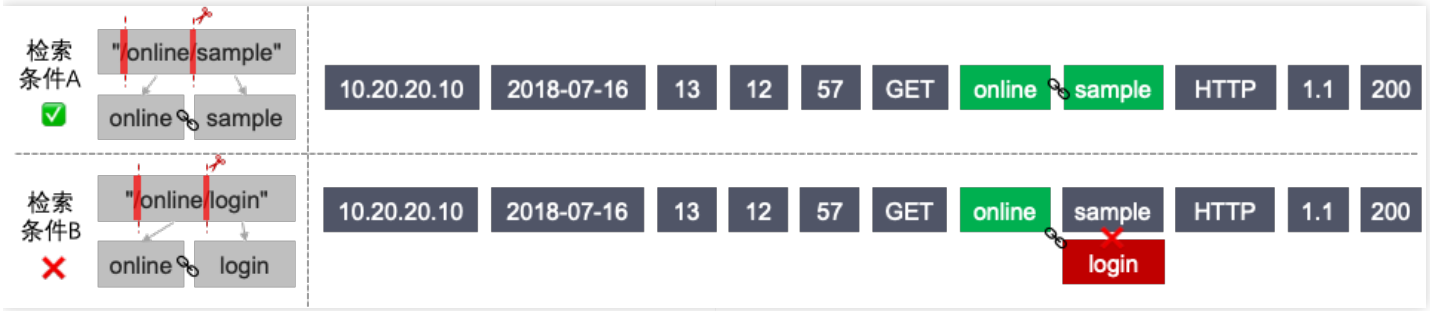
分词示意
检索一个较长的日志时，通常只使用其中一部分内容进行检索。例如：需要检索出包含 sample 的如下完整日志：

```
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。

  分词设置
  日志分词依据包含两类，可在 配置索引 中设置。

- 分词符：可自定义需要按照哪些符号对日志进行切分，支持英文符号及\n\t\r\。在上面的例子中， @&()='",;:<>[]{}/ \n\t\r\\ 即为分词符。
- 是否包含中文：中文较为特殊，不能使用中文符号作为分词符，而且仅按照符号对中文进行分词也往往达不到预期效果。例如，日志为 用户登录失败，密码错误 ，需要使用 "登录失败" 进行检索，是不能通过符号对日志进行切分来满足检索需求的。此时可在 配置索引 中设置该日志“包含中文”，日志服务（Cloud Log Service，CLS）将自动将日志中的每一个汉字及汉字符号切分为独立的词。

# 索引

CLS 对日志进行分词后，采用倒排索引来存储日志中每个“词”在文档中的位置，用户快速根据检索条件匹配具体的原始日志。倒排索引是文档检索系统中最常用的数据结构。
索引决定了日志能够以什么样的条件来进行检索和分析，开启索引后才能进行检索分析。因此在上传日志数据前，需要为日志主题设置一个合理的索引规则，配置方式参见 配置索引。

# 主题分区

## 分区定义

主题分区（Partition）是日志服务（Cloud Log Service，CLS）的最小读写单元，一个日志主题可以包含多个分区，分区数量决定了单个日志主题的最大数据读写能力。单个分区的读写能力如下表：

| | |
|---|---|
| 频控 | 单个主题分区读请求限制：200QPS。超限返回状态码429，提示错误 SpeedQuotaExceed。<br>单个主题分区写请求限制：500QPS。超限返回状态码429，提示错误 SpeedQuotaExceed。<br>说明：此处500QPS不代表500条日志。 一次请求中的 pb 包可以包含多条日志批量上传。 |
| 流控 | 单个主题分区读流量无限制，建议不超过5MB/s。<br>单个主题分区压缩后写流量限制：5MB/s。超限返回状态码429，提示错误 SpeedQuotaExceed。 |

假设某个日志主题有10个读写态分区，则该日志主题最大可承载的压缩后日志写流量为 5MB/s * 10 = 50MB/s 。单个日志主题最大可拥有50个分区，对应的最大压缩后日志写流量为 5MB/s * 50 = 250MB/s 。

> 说明：
>
> 实际使用过程中建议开启 自动分裂 功能，CLS 将根据实际数据量自动扩增主题分区数量，无需手动管理分区。

## 分区属性

主题分区包含以下三个属性，这些属性主要由日志服务内部使用。

- 分区 ID：每个分区在同一个日志主题下有唯一 ID，该 ID 在创建或操作后由系统确定。
- 分区范围：每个分区均有区间范围，每个区间范围均为左闭右开区间。一个日志主题的有效范围为 MD5 的取值范围为 [00000000000000000000000000000000,ffffffffffffffffffffffffffffffff），所有读写态的主题分区会按左闭右开的原则切分整个取值范围，上报数据时会随机写入日志主题的某个分区。
- 分区状态：创建主题分区时，所有分区状态均为读写态，合并和分裂操作会将原分区状态改变为只读态。
  - 读写态：表示当前分区可以进行读写操作。
  - 只读态：表示当前分区仅允许进行读操作，不可再写入数据。
    可通过如下方式手动管理分区：
- 合并分区：将两个范围相邻的两个读写态分区合并成一个分区。合并完成之后，原来的两个分区状态将会变成只读态，新合成的分区为读写态，新分区的范围会覆盖原来两个分区范围。
- 分裂分区：将一个读写态的分区分裂成两个小范围分区，分裂分区时需指定一个分裂点的 MD5 值（该值必须大于起止位置，小于终止位置）。分裂成功后，原来的分区状态将会变成只读态，新分裂的分区为读写态，且新分区的范围会覆盖原来分区的范围。

# 快速入门
# 入门指南

## 概述

日志服务是集日志采集、日志存储、日志检索于一体的平台化服务。通过日志服务，可以将分散的日志汇聚起来进行集中式存储管理，并进行检索。

为了帮助用户快速入门日志服务，本文将演示使用日志服务的基本功能，演示包括：

- 如何使用 LogListener 采集日志
- 如何检索日志

## 操作步骤

### 1. 服务开通

首先，您需要申请开通日志服务。

### 2. 下载安装 LogListener

LogListener 是日志服务的采集客户端，通过 LogListener 可实现快速无侵入式的把日志数据采集到日志服务中来，具体安装步骤如下。

> 说明：
>
> - 云服务器(CVM)安装LogListener前，必须检查目标CVM上LogListener进程是否存在，执行命令: ps aux|grep loglistener|grep -v grep 。若进程不存在，则可以继续安装LogListener。
> - 若进程已存在，则不允许重复安装LogListener，需排查该CVM节点的安装历史。若该CVM节点属于某云产品的服务节点，可参照该云产品的产品文档，在租户端页面使用日志相关功能。

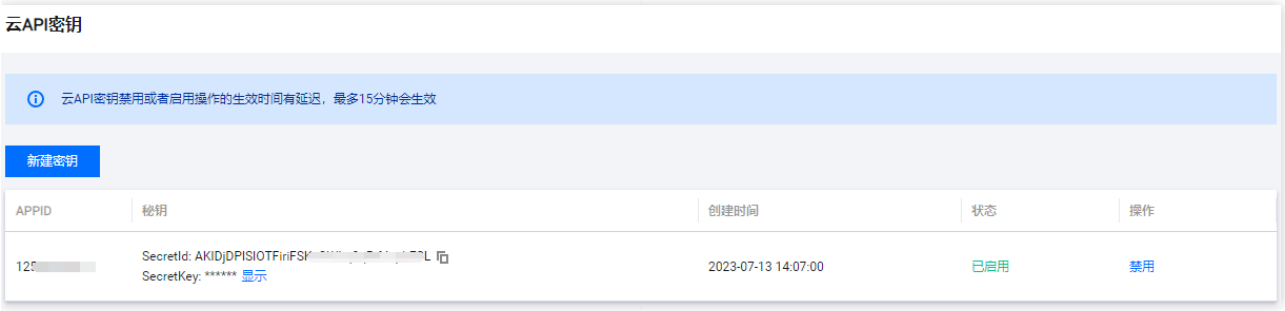
#### 2.1 判断网络是否可达

安装 LogListener 要求日志源机器的网络与日志服务的可用地域网络互通（云服务器 CVM 默认内网访问日志服务）。您可以执行以下命令检查网络连通性。

```
telnet <CLS-server host> <CLS-server port>
```

#### 2.2 查看（或创建）密钥对

登录访问管理，查看（或创建）密钥对，并确认密钥状态为启用。




### 2.3 安装 LogListener

本文演示日志采集的环境搭建在云服务器 CentOS 7.2（64位）环境上。LogListener 下载及详细安装步骤请参见 LogListener安装指南。

## 3. 创建日志集和日志主题

日志主题是日志数据采集、存储、检索和分析的基本单元，一个日志主题通常对应某一个应用/服务（具备类似的日志结构）。还可以使用日志集对日志主题进行分组，日志集本身不存储任何日志数据，仅方便用户管理日志主题。

1. 在左侧导航栏中单击日志主题，进入到日志主题管理页面。
2. 单击创建日志主题，填写相关信息后，单击确定。
   创建好的日志主题将会出现在日志主题列表中。




## 4. 创建机器组

日志服务使用机器组来统一管理一组日志源机器。
登录日志服务控制台后，在左侧导航栏单击机器组管理，进入到机器组管理页面。在页面顶部选择合适的地域，单击新建机器组开始创建，一个机器组可以填入多个机器 IP 地址（每行一个 IP 地址），更多信息请参考 机器组管理。

创建好机器组后，单击机器组列表中的查看，检查 LogListener 与服务端的连接状态，若状态正常，则表示客户端 LogListener 已成功连接到日志服务。

## 5. 配置采集规则和索引配置

1. 在左侧导航栏单击日志主题，进入日志主题页面。

2. 单击日志主题ID，进入到对应的日志主题基本信息页面。

3. 选择采集配置页签，在 LogListener 采集配置中单击新增。



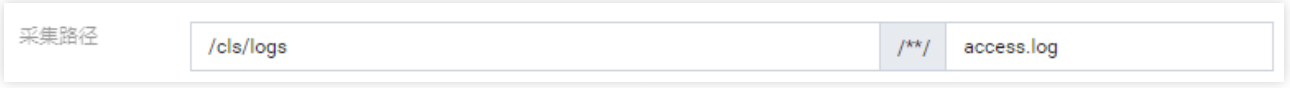

4. 在日志数据源页面，选择分隔符-文件日志。

5. 选择预先创建好的机器组，将当前日志主题与机器组关联起来后，LogListener 将按照所配置的规则监听采集机器组上的日志文件。
一个日志主题可以绑定多个机器组，但一个日志文件只能上报到一个日志主题。

6. 配置采集路径
采集路径需要匹配机器上日志文件的绝对路径，填写参数有两个：目录前缀和日志文件名，填写格式为 [目录前缀表达式]/**/[文件名表达式] ，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，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 |
| /**/ | 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 |

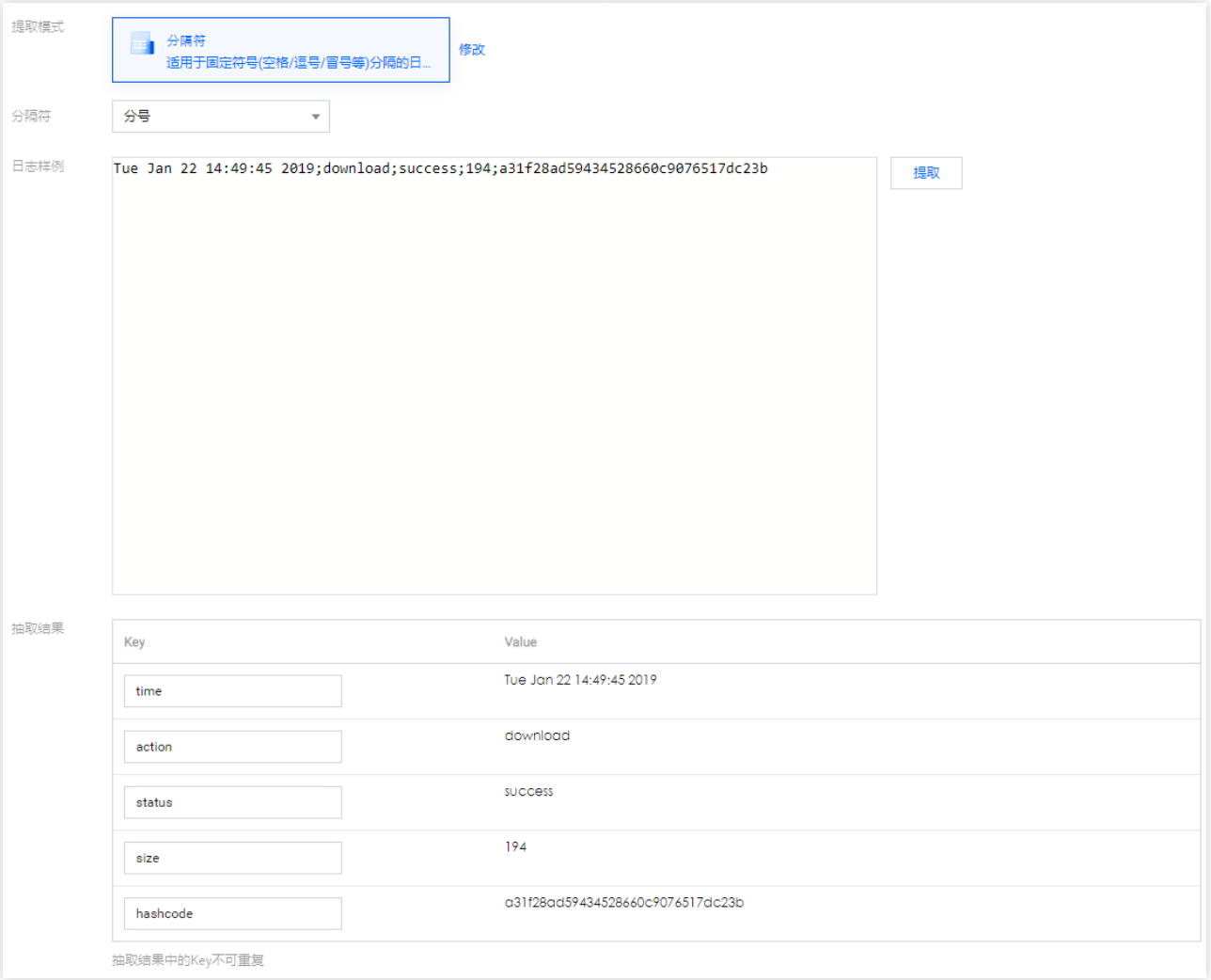
例如待采集文件的绝对路径是 /cls/logs/access.log ，则采集路径填写的目录前缀是 /cls/logs ，日志文件名填写 access.log，如下图所示：



7. 配置解析模式
   日志服务提供多种日志解析模式（例如单行全文、分隔符、JSON、完全正则等模式），本文以分隔符格式日志为例进行说明（详情参考 分隔符格式），日志样例如下：

   Tue Jan 22 14:49:45 2019;download;success;194;a31f28ad59434528660c9076517dc23b

   - 选择提取模式
     本文以分隔符格式日志举例，所以在"键值提取模式"配置项中选择分隔符，并且选择分号作为日志分隔符。
   - 输入日志样例并抽取键值对
     在日志样例框中输入一条完整的日志，确认后将自动抽取键值对（key-value），然后为每组键值对定义唯一的键名称（key）。
     在本示例中，日志被解析成 Tue Jan 22 14:49:45 2019 ， download ， success ， 194 和 a31f28ad59434528660c9076517dc23b 五个字段，依次为每个字段定义键名称（key）：time ， action ， status ， size ， hashcode ，这样 LogListener 将按照所定义的结构化格式进行数据采集。




8. 填写索引配置，开启全文索引、键值索引，填写相关信息后，单击提交。
   在上述例子中，为 time ， action ， status ， size ， hashcode 设置键值索引，其中 size 设置为 long 类型。




开启索引后，新写入的数据将会按照所配置规则建立索引，索引会持久化存储一段时间（根据您所配置的存储周期而定），只有建立索引的部分才能进行日志查询分析。所以，修改索引规则或关闭索引仅对新写入的数据生效，未过期的历史数据仍可被检索。

## 6. 检索分析日志

1. 在左侧导航栏单击检索分析，进入到日志检索页。
2. 选择检索的时间范围和日志主题，然后在输入框填写检索语法（语法支持关键词检索、模糊检索、范围检索等方式，详情参考 语法规则），最后单击搜索，即可检索日志。
   - 示例一：查询失败的日志
     检索语句： status:fail
   - 示例二：查询下载文件大小超过300K的日志
     检索语句： action:download and size:>300

# 日志上传API示例

## cls_kv_pb2.py

```
# Generated by the protocol buffer compiler.  DO NOT EDIT!
# source: cls_kv.proto

import sys
_b=sys.version_info[0]<3 and (lambda x:x) or (lambda x:x.encode('latin1'))
from google.protobuf import descriptor as _descriptor
from google.protobuf import message as _message
from google.protobuf import reflection as _reflection
from google.protobuf import symbol_database as _symbol_database
from google.protobuf import descriptor_pb2
# @@protoc_insertion_point(imports)

_sym_db = _symbol_database.Default()

DESCRIPTOR = _descriptor.FileDescriptor(
  name='cls_kv.proto',
  package='cls',
  serialized_pb=_b('\n\x0c\x63ls_kv.proto\x12\x03\x63ls\"^\n\x03Log\x12\x0c\n\x04time\x18\x01 \x02
(\x03\x12\"\n\x08\x63ontents\x18\x02 \x03(\x0b\x32\x10.cls.Log.Content\x1a%\n\x07\x43ontent\x12
\x0b\n\x03key\x18\x01 \x02(\t\x12\r\n\x05value\x18\x02 \x02(\t\"$\n\x06LogTag\x12\x0b\n\x03key\x
18\x01 \x02(\t\x12\r\n\x05value\x18\x02 \x02(\t\"w\n\x08LogGroup\x12\x16\n\x04logs\x18\x01 \x03
(\x0b\x32\x08.cls.Log\x12\x13\n\x0b\x63ontextFlow\x18\x02 \x01(\t\x12\x10\n\x08\x66ilename\x18\x
03 \x01(\t\x12\x0e\n\x06source\x18\x04 \x01(\t\x12\x1c\n\x07logTags\x18\x05 \x03(\x0b\x32\x0b.cl
s.LogTag\"3\n\x0cLogGroupList\x12#\n\x0clogGroupList\x18\x01 \x03(\x0b\x32\r.cls.LogGroup')
)
_sym_db.RegisterFileDescriptor(DESCRIPTOR)

_LOG_CONTENT = _descriptor.Descriptor(
  name='Content',
  full_name='cls.Log.Content',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='key', full_name='cls.Log.Content.key', index=0,
      number=1, type=9, cpp_type=9, label=2,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
```

```
    _descriptor.FieldDescriptor(
      name='value', full_name='cls.Log.Content.value', index=1,
      number=2, type=9, cpp_type=9, label=2,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=78,
  serialized_end=115,
)

_LOG = _descriptor.Descriptor(
  name='Log',
  full_name='cls.Log',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='time', full_name='cls.Log.time', index=0,
      number=1, type=3, cpp_type=2, label=2,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='contents', full_name='cls.Log.contents', index=1,
      number=2, type=11, cpp_type=10, label=3,
      has_default_value=False, default_value=[],
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[_LOG_CONTENT, ],
```

```
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=21,
  serialized_end=115,
)

_LOGTAG = _descriptor.Descriptor(
  name='LogTag',
  full_name='cls.LogTag',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='key', full_name='cls.LogTag.key', index=0,
      number=1, type=9, cpp_type=9, label=2,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='value', full_name='cls.LogTag.value', index=1,
      number=2, type=9, cpp_type=9, label=2,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=117,
  serialized_end=153,
)
```

```
_LOGGROUP = _descriptor.Descriptor(
  name='LogGroup',
  full_name='cls.LogGroup',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='logs', full_name='cls.LogGroup.logs', index=0,
      number=1, type=11, cpp_type=10, label=3,
      has_default_value=False, default_value=[],
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='contextFlow', full_name='cls.LogGroup.contextFlow', index=1,
      number=2, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='filename', full_name='cls.LogGroup.filename', index=2,
      number=3, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='source', full_name='cls.LogGroup.source', index=3,
      number=4, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='logTags', full_name='cls.LogGroup.logTags', index=4,
      number=5, type=11, cpp_type=10, label=3,
      has_default_value=False, default_value=[],
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
```

```
  ],
  options=None,
  is_extendable=False,
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=155,
  serialized_end=274,
)

_LOGGROUPLIST = _descriptor.Descriptor(
  name='LogGroupList',
  full_name='cls.LogGroupList',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='logGroupList', full_name='cls.LogGroupList.logGroupList', index=0,
      number=1, type=11, cpp_type=10, label=3,
      has_default_value=False, default_value=[],
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=276,
  serialized_end=327,
)

_LOG_CONTENT.containing_type = _LOG
_LOG.fields_by_name['contents'].message_type = _LOG_CONTENT
_LOGGROUP.fields_by_name['logs'].message_type = _LOG
_LOGGROUP.fields_by_name['logTags'].message_type = _LOGTAG
_LOGGROUPLIST.fields_by_name['logGroupList'].message_type = _LOGGROUP
DESCRIPTOR.message_types_by_name['Log'] = _LOG
DESCRIPTOR.message_types_by_name['LogTag'] = _LOGTAG
DESCRIPTOR.message_types_by_name['LogGroup'] = _LOGGROUP
```

```
DESCRIPTOR.message_types_by_name['LogGroupList'] = _LOGGROUPLIST

Log = _reflection.GeneratedProtocolMessageType('Log', (_message.Message,), dict(

  Content = _reflection.GeneratedProtocolMessageType('Content', (_message.Message,), dict(
    DESCRIPTOR = _LOG_CONTENT,
    __module__ = 'cls_kv_pb2'
    # @@protoc_insertion_point(class_scope:cls.Log.Content)
    ))
  ,
  DESCRIPTOR = _LOG,
  __module__ = 'cls_kv_pb2'
  # @@protoc_insertion_point(class_scope:cls.Log)
  ))
_sym_db.RegisterMessage(Log)
_sym_db.RegisterMessage(Log.Content)

LogTag = _reflection.GeneratedProtocolMessageType('LogTag', (_message.Message,), dict(
  DESCRIPTOR = _LOGTAG,
  __module__ = 'cls_kv_pb2'
  # @@protoc_insertion_point(class_scope:cls.LogTag)
  ))
_sym_db.RegisterMessage(LogTag)

LogGroup = _reflection.GeneratedProtocolMessageType('LogGroup', (_message.Message,), dict(
  DESCRIPTOR = _LOGGROUP,
  __module__ = 'cls_kv_pb2'
  # @@protoc_insertion_point(class_scope:cls.LogGroup)
  ))
_sym_db.RegisterMessage(LogGroup)

LogGroupList = _reflection.GeneratedProtocolMessageType('LogGroupList', (_message.Message,), dict(
  DESCRIPTOR = _LOGGROUPLIST,
  __module__ = 'cls_kv_pb2'
  # @@protoc_insertion_point(class_scope:cls.LogGroupList)
  ))
_sym_db.RegisterMessage(LogGroupList)

# @@protoc_insertion_point(module_scope)
```

## env.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
SECRETID = ""
SECRETKEY = ""
```

```
HOST = ""
PORT = 8028
REGION = ""
```

## signature.py

```
#!/USR/bin/env python
# -*- coding: utf-8 -*-

import hashlib
import random
import time
import hmac
import binascii
import sys
import urllib
from env import *
import hashlib, hmac, json, os, sys, time
from datetime import datetime

def genV3Sig(timestamp, content_type, payload, host):
    # 密钥参数
    secret_id = SECRETID
    secret_key = SECRETKEY

    service = "cls"
    region = REGION
    version = "2020-10-16"
    algorithm = "TC3-HMAC-SHA256"
    #timestamp = int(time.time())
    #timestamp = 1551113065
    date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
    #params = {"Limit": 1, "Filters": [{"Name": "instance-name", "Values": [u"未命名"]}]}

    # ************* 步骤 1：拼接规范请求串 *************
    http_request_method = "POST"
    canonical_uri = "/"
    canonical_querystring = ""
    ct = content_type
    #payload = json.dumps(params)
    canonical_headers = "content-type:%s\nhost:%s\n" % (ct, host)
    signed_headers = "content-type;host"
    hashed_request_payload = hashlib.sha256(payload).hexdigest()
    #hashed_request_payload = payload
    #print hashed_request_payload
    canonical_request = (http_request_method + "\n" +
```

```
                canonical_uri + "\n" +
                canonical_querystring + "\n" +
                canonical_headers + "\n" +
                signed_headers + "\n" +
                hashed_request_payload)
    #print(canonical_request)

    # ************* 步骤 2：拼接待签名字符串 *************
    credential_scope = date + "/" + service + "/" + "tc3_request"
    hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
    string_to_sign = (algorithm + "\n" +
                str(timestamp) + "\n" +
                credential_scope + "\n" +
                hashed_canonical_request)
    #print(string_to_sign)

    # ************* 步骤 3：计算签名 *************
    # 计算签名摘要函数
    def sign(key, msg):
        return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
    secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
    secret_service = sign(secret_date, service)
    secret_signing = sign(secret_service, "tc3_request")
    signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
    #print(signature)

    # ************* 步骤 4：拼接 Authorization *************
    authorization = (algorithm + " " +
                "Credential=" + secret_id + "/" + credential_scope + ", " +
                "SignedHeaders=" + signed_headers + ", " +
                "Signature=" + signature)
    #print hashed_request_payload
    #print(authorization)
    return authorization
```

## upload_client.py

```
#!/bin/python
#-*- coding:utf-8 -*-
from signature import *
import sys
import httplib
import json
import time
import string
import random
```

```
import cls_kv_pb2
from env import *
from datetime import datetime, date, timedelta

def push_log(conn, topic_id, body, host):
    content_type = "application/x-protobuf"

    timestamp = int(time.time())

    headers = {
        "Content-Type" : content_type,
        "Content-Length" : str(len(body)),
        "X-TC-Action": "UploadLog",
        "X-TC-Region": "ap-city",
        "X-TC-Timestamp": timestamp,
        "X-TC-Version": "2020-10-16",
        "TopicId": topic_id,
        "Authorization" : genV3Sig(timestamp, content_type, body, host)
    }
    print(headers)

    conn.request(method="POST", url='http://%s/' % host, body=body, headers=headers)
    return conn.getresponse()

def pb_gen(start_index, size):
    logGroupList = cls_kv_pb2.LogGroupList()
    LogGroup = logGroupList.logGroupList.add()
    log = LogGroup.logs.add()
    log.time = int(time.time())
    for index in range(size):
        content = log.contents.add()
        key = "name,index,test_name_index,"
        content.key = key + ''.join(random.sample(string.ascii_letters + string.digits, 60))
        value = "test_" + str(start_index) + "_" + str(index) + "_"
        for i in range(2):
            content.value = content.value + ''.join(random.sample(string.ascii_letters + string.digits, 60))
    data = logGroupList.SerializeToString()

    return data

if __name__ == '__main__':

    host = HOST
    port = PORT
    conn = httplib.HTTPConnection(host, port)

    if(len(sys.argv) == 1):
```

```
    print 'invalid command '
    print 'python ' + sys.argv[0] + ' pushlog $topic_id $log_size(236B per log) $log_count'
    exit()

if(sys.argv[1] == 'pushlog'):
    size = sys.argv[3]
    index = sys.argv[4]
    for i in range(1, int(index)+1):
        data = pb_gen(i, int(size))
        response = push_log(conn, sys.argv[2], data, host)
        print response.getheaders()
        print response.read()
        print response.status
        if (i % 300) == 0:
            time.sleep(2)
```

# 操作指南
# 资源管理
# 日志集

本文档主要介绍如何通过控制台操作日志集。

## 更新日志集名称

1. 进入日志主题管理页面。
2. 单击管理日志集。
3. 找到需要更新的日志集，在其右侧操作栏中，单击编辑。




4. 修改日志集信息，单击确定，即可更新日志集信息。

## 删除日志集

1. 进入日志主题管理页面。
2. 单击管理日志集。

> 注意：
>
> 删除日志集之前，需清空所包含的全部日志主题。

3. 找到需要删除的日志集，在其右侧操作栏中，单击删除。

4. 在弹出的确认删除窗口中，单击确定，即可删除该日志集。

# 编辑标签

1. 进入日志主题管理页面。
2. 单击管理日志集。
3. 找到需要设置标签的日志集，在其右侧操作栏中，单击编辑标签。
4. 输入标签键和标签值，单击确定。
   系统修改对应的资源标签。
5. 日志集关联标签成功后，单击确定。

# 日志主题

本文档主要介绍如何通过控制台对日志主题进行增删更新等操作。

## 新增日志主题

1. 登录日志服务控制台。
2. 在左侧导航栏中，单击日志主题。
3. 单击创建日志主题，输入相应信息。
    - 日志主题名称：例如 nginx。
    - 日志保存策略：
        - 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。
    - 匿名写入：开启后支持匿名写入浏览器和各种移动端的日志数据，详情请参见 匿名日志采集。
    - 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。
    - 日志主题描述：描述信息，方便后续了解该日志主题用途。
    - 高级设置：
        - 分区数量： 默认新建1个分区，关于主题分区请参见 主题分区。
        - 分区自动分裂：默认开启。
        - 最大分裂数量：可自定义分裂数量，最大分裂数量为50个。
4. 单击确定，新增日志主题。

## 删除日志主题

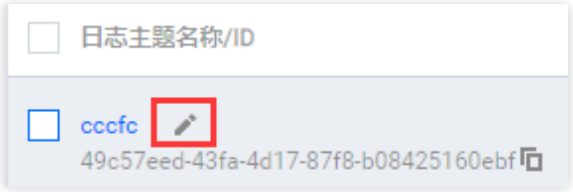
1. 进入日志主题管理页面，找到您需要删除的日志主题，在其右侧操作栏中，单击删除。

> 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。

2. 在弹出的确认删除窗口中，单击确定，即可删除该日志主题。

# 更新日志主题名称

1. 进入日志主题管理页面，鼠标移动到待修改名称的日志主题名称，单击旁边的编辑图标。




2. 修改日志主题名称，单击确定。

# 编辑日志主题标签

1. 进入日志主题管理页面，勾选需要编辑标签的日志主题，单击编辑标签。
2. 输入标签键和标签值，单击确定。
   系统修改对应的资源标签。
3. 日志主题关联标签成功后，单击确定。

# 主题分区
# 分裂主题分区

本文介绍如何在日志服务控制台上分裂主题分区。日志服务提供了自动分裂和手动分裂两种操作模式。为了减小因流量突发而受到读写限制，建议您开启自动分裂。

## 操作场景

在创建日志主题时，会设置一个初始的分区数量，后续您可根据实际情况，使用分裂分区或者合并分区的功能来扩缩分区数。由于单个主题分区限制写请求（最高500次/S）和写流量（最高5MB/S），如果您的服务请求超过单个分区上限，可通过分裂分区增大日志主题的吞吐，避免写入失败。

## 操作步骤

### 自动分裂

> 说明：
>
> 开启自动分裂功能后，如果主题分区持续触达了写请求或者写流量的阈值，日志服务会根据实际写入情况，自动分裂至合理的分区数（最大分裂数量为50个）。
>
> 如果日志主题的分区数已达到所设置的最大值，日志服务将不再触发自动分裂，所超限的部分将被拒绝，并返回请求超限错误码。

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。
3. 找到您需要开启自动分裂的日志主题，单击右侧操作列的编辑。



4. 在弹出的对话框中，将分区自动分裂设置为 ，并设置最大分裂数量（建议设为最大值50）。
5. 单击确定，完成分区自动分裂操作。

### 手动分裂

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。
3. 单击您需要开启手动分裂的日志主题的名称，进入主题详情基本信息页面。
4. 在主题分区（Partition）管理区域中，找到需要分裂的目标主题分区，单击编辑。
5. 在弹出的对话框中，将“操作”选择为分裂，设置“分区个数”。

每个分区均有区间范围，且每个区间范围均为左闭右开区间，默认平均分裂。

6. 单击确定，完成分裂操作。

# 合并主题分区

## 操作场景

本文介绍如何在日志服务控制台上合并主题分区。通过对相邻区间范围的分区进行合并，可以减少主题分区的数量，避免资源浪费。

## 注意事项

- 每个日志主题至少有一个分区。若您只有1个分区，将不能进行合并。
- 每次只能合并指定主题分区相邻的一个合并对象。

## 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，进入日志主题管理页面。
3. 单击待合并主题分区的日志主题的名称，进入主题详情基本信息页面。
4. 在主题分区（Partition）管理区域中，找到需要合并的目标主题分区，单击编辑。
5. 在弹出的对话框中，将“操作”选择为合并，并选择需要合并的合并对象。
6. 单击确定，完成合并操作。

# 机器组

## 操作场景

机器组是日志服务中 LogListener 所采集日志的服务器对象，通过机器组的方式来配置 LogListener 采集日志的服务器。一般而言，根据不同业务场景来划分不同的机器组，可以方便您管理日志服务。

## 操作步骤

### 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，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。
- 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Linux机器 IP 地址。

  > 注意
  >
  > - 所关联机器需已安装 LogListener。
  > - 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 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。

5. 单击确定，即可完成创建。

### 通过机器标识创建机器组

如果您的机器 IP 地址经常变动，利用机器 IP 配置机器组操作繁琐，每次 IP 变更将需要修改对应的机器组配置。
CLS 支持利用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，低版本的 LogListener 需手动更新。

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，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。
   - 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，详情请查看 LogListener 安装指南（Linux版）
     > - 每行填写一个 IP 地址。

   - LogListener 自动升级：建议开启 LogListener 自动升级，详情可参见 LogListener 升级指南。
   - 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，详情可参见 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 /user/local 为例：

   ```
   vi /usr/local/loglistener-2.3.0/etc/loglistener.conf
   ```

7. 按 i，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）。

```
proxy_host = ap-guangzhou.
proxy_port =
secret_id =
secret_key =
group_ip =
# define Loglistener by labels, separated by ,
group_label = nginx_access
instance_id = loglistener-2059
pos_file = data/pos.dat
# 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 = 10
# Max memory loglistener would use.
max_mem = 2097152000
# Max bytes send per second, 0 for unlimited.
max_send_rate = 0
# Max scanning depth corresponding to ** in wildpath
max_depth = 10
# Compress the upload traffic by LZ4.
request_compression = true
# Replace special charactors to space. ['\0'] => ' '
replace_special_charactors = false
# Logistener do its best to save memorys when true.
memory_tight_mode = false
```

9. 按 Esc，退出编辑模式。

10. 输入 :wq，按 Enter，保存设置。

11. 执行如下命令，重启 LogListener，即可完成创建。

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

## 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。
   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。
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. 在机器组管理页面，选择需要删除的机器组，单击更多 > 删除。
2. 在弹出的提示窗口，单击确定，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。

# 编辑机器组标签

1. 进入机器组管理页面，勾选需要编辑标签的机器组，单击编辑标签。
2. 输入标签键和标签值，单击确定。
3. 机器组关联标签成功后，单击确定。

# 日志采集
# 采集概述

## 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/Web 端 | SDK 上传日志/ 使用匿名写入采集日志 |
| 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 |

- 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 文档 |
| 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 |

- 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：

| 解析方式 | 说明 |
| --- | --- |
| 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。<br>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``<br> 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
```

# 规格限制

日志采集的规格限制，请您参见：

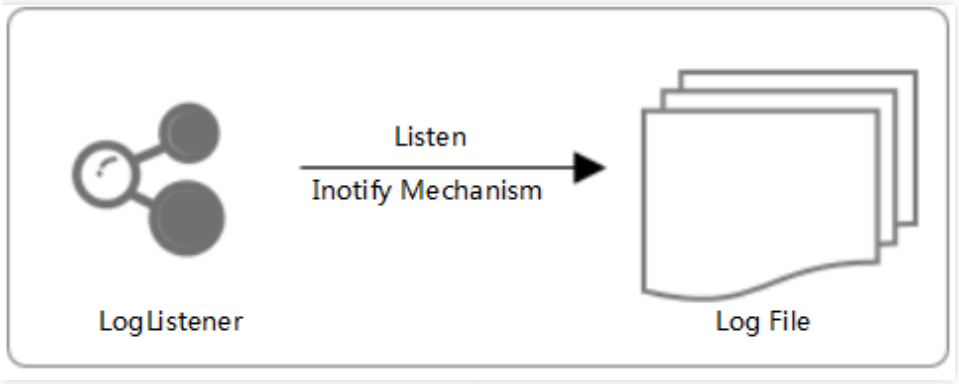
- LogListener 限制说明
- API 日志上传限制说明
- COS 数据导入规格限制

# LogListener采集
# 采集机制

LogListener 是日志服务 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，本篇文档将详细阐述 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：

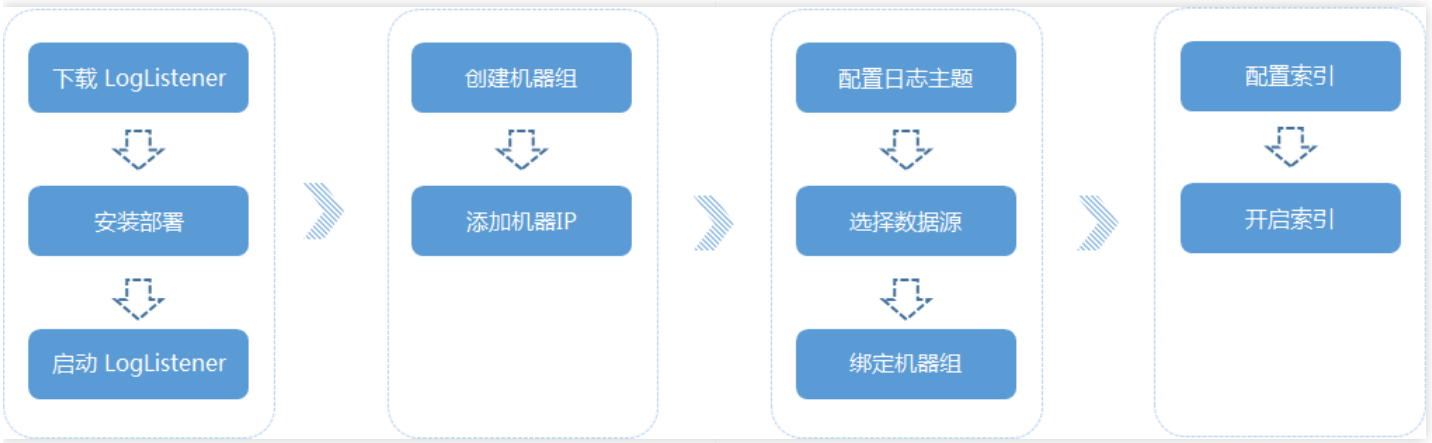
- 机器重启后，会自动拉起 LogListener。
- LogListener 进程挂掉重启后，会根据所记录的偏移位置继续上报日志。
- 目前一个日志文件仅能上报到一个日志主题。若有多个日志主题关联到同一个日志文件，配置信息会覆盖，因此该日志文件实际只会上报到最后一个主题中。

# LogListener 使用流程

## 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，并添加机器 IP。
3. 前往日志主题的采集配置页，输入日志路径确定数据源，并绑定机器组。详细操作示例可参阅 单行全文采集 文档。
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。详细操作示例可参阅 单行全文采集 文档。
   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。

# LogListener 升级指南
LogListener 升级指南

## 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。

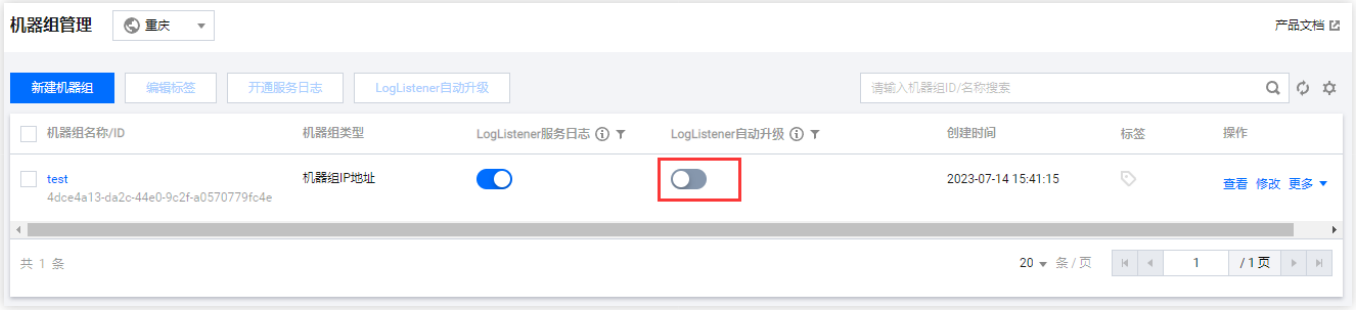
> 说明
>
> - LogListener 自动升级功能仅在 LogListener 2.8.8 及以上版本开始支持，为了更好的使用体验，建议先通过 半自动升级 或者 手动升级LogListener将 LogListener 升级至2.8.8以上。
> - 脚本半自动升级功能需 Python2.7 支持，若采集机器上安装的不是该版本，将无法使用此方式进行升级。建议升级 Python 版本， 或使用 手动升级LogListener。

- 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，单击机器组管理，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 ，即可开启自动升级。




4. 在弹出的窗口中，开启 LogListener 开关，指定升级时间段（默认为当前时间至后两小时，如17:01～19:01）。




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，单击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，单击机器组管理，进入机器组管理页面。
3. 找到需要升级的目标机器组，单击操作列对应的更多 > 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 2.8.8及以上版本

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

```
./update auto_upgrade --root-dir /usr/local/loglistener --url http://cls-loglistener-1255000003.cos.shjr.cos.financetcloud.cn/loglistener-linux-x64-2.X.X.tar.gz
```

参数描述如下：

| 参数 | 描述 |
| --- | --- |
| --root-dir | 升级的新版本 LogListener 的安装路径。此处默认为 /usr/local/loglistener |
| --url | 需指定版本，可在 loglistener-linux-x64后增加“-版本号”。例：2.9.0版本的 LogListener 的安装包地址为：http://cls-loglistener-1255000038.cos.city.csp.abc.com/loglistener-linux-x64-2.9.0.tar.gz<br>注意：<br>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。 |

loglistener2.4.6版本以及loglistener2.8.2 版本升级方式，请参照loglistener 2.4.6/2.8.2 升级到2.8.8 升级流程。

# Loglistener 2.4.6/2.8.2 升级到2.8.8 升级流程

## 前置操作

1、获取新版Loglistener 2.8.8 版本的包；
2、上传Loglistener2.8.8的包到 resource_reserve@tce.com 账号下的CSP中桶名为cls-loglistener的存储桶中，并获取下载链接；
3、更新文档组的包 联系现场SPM，出文档包；

## 手动升级（推荐）

更新client端
ps：以下操作均需要在日志采集节点上（安装loglistener的机器上）（一般为客户业务overlay机器）。

## 执行升级

1. 通过前置操作阶段第一步，获取新安装包的对象存储链接，在部署机上下载新的安装包。

2. 检查机器上loglistener运行状态

   /etc/init.d/loglistenerd check

```
[root@VM_13_146_centos /usr/local/loglistener-2.4.6/tools]# /etc/init.d/loglistenerd check
[OK] loglistener is running ok
[OK] check loglistener hearbeat ok
group ip:10.19.13.146
host:data.tcloud-cls-cgi.chongqing.yfm4-v6-iaas.tcecloud.fsphere.cn
port:8028
gethostbyname ip:10.19.41.235
[OK] check loglistener config ok
[root@VM_13_146_centos /usr/local/loglistener-2.4.6/tools]#
```

   或

```
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]# /etc/init.d/loglistenerd check
[OK] loglistener is running ok
[OK] check loglistener hearbeat ok
group ip:10.19.13.151
host:cls.api3.yfm4-v6-iaas.tcecloud.fsphere.cn
port:80
gethostbyname ip:10.19.41.200
[OK] check loglistener config ok
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]#
```

3. 执行如下命令： /etc/init.d/loglistenerd stop

```
[root@VM_13_146_centos /usr/local/loglistener-2.4.6/tools]# /etc/init.d/loglistenerd stop
注意：正在将请求转发到“systemctl disable loglistenerd.service”。
Removed symlink /etc/systemd/system/multi-user.target.wants/loglistenerd.service.
[OK] stop loglistener success
[root@VM_13_146_centos /usr/local/loglistener-2.4.6/tools]#
[root@VM_13_146_centos /usr/local/loglistener-2.4.6/tools]#
[root@VM_13_146_centos /usr/local/loglistener-2.4.6/tools]# /etc/init.d/loglistenerd check
[ERROR] loglistener is not running
[root@VM_13_146_centos /usr/local/loglistener-2.4.6/tools]#
```

或

```
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]#
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]# /etc/init.d/loglistenerd stop
注意：正在将请求转发到“systemctl disable loglistenerd.service”。
Removed symlink /etc/systemd/system/multi-user.target.wants/loglistenerd.service.
[OK] stop loglistener success
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]#
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]#
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]#
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]#
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]# /etc/init.d/loglistenerd check
[ERROR] loglistener is not running
[root@VM_13_151_tlinux /usr/local/loglistener-2.8.2/tools]#
```

4. 执行如下命令：

   mkdir /tmp/loglistener_bak/

   cp -r /usr/local/loglistener-2.4.6 /tmp/loglistener_bak/

   cp -r /usr/local/loglistener-2.4.6/etc /tmp/loglistener_bak/etc

   cp -r /usr/local/loglistener-2.4.6/data /tmp/loglistener_bak/data

   或者

   mkdir /tmp/loglistener_bak/

   cp -r /usr/local/loglistener-2.8.2 /tmp/loglistener_bak/

   cp -r /usr/local/loglistener-2.8.2/etc /tmp/loglistener_bak/etc

   cp -r /usr/local/loglistener-2.8.2/data /tmp/loglistener_bak/data

5. cd *<安装包下载目录>*解压安装包至安装路径

   如： tar -zxvf loglistener-linux-x64-2.8.8.tar.gz -C /usr/local/

6. 将原组件的配置文件拷贝到新组件目录下：
   因2.4.6 以及2.8.2 初始化方式均不一样，所以针对升级到2.8.8 需要手动将配置文件初始化。
   cp -r /tmp/loglistener_bak/data /usr/local/loglistener/
   提示是否覆盖，直接回车。

7. 在/usr/local/loglistener/tools 目录下执行安装 ./loglistener.sh install
   输入yes 覆盖安装

```
[root@W*_**_***_tlinux /usr/local/loglistener-2.4.6/tools]# ./loglistener.sh install
There is a historical version installed, and continuing the installation will overwrite the old version. do you want to continue?(yes/n
o): yes
[OK] check dependencies ok
[RESULT] install loglistener success, use './loglistener.sh init' to init loglistener's config
```

8. 重新初始化
   2.4.6 和2.8.2 都需要执行该命令
   ./loglistener.sh init -secretid AKIDC6fvtXrpSkhJa3IDo40**** -secretkey xZr6JDn3LaW7NhynHQm43h3Rm***
   -host data.tcloud-cls-cgi.agent.city.*** -port 80 -region **

9. 执行启动命令： systemctl start loglistenerd 并检查loglistener的版本

```
[root@VM_13_146_centos /usr/local/loglistener/tools]# /etc/init.d/loglistenerd -v
loglistener-2.8.8
[root@VM_13_146_centos /usr/local/loglistener/tools]#
```

10. 观察日志是否无报错 /usr/local/loglistener/log/loglistener.log

11. 处理旧包
    mkdir -p /usr/local/delete_old_loglistener
    tar cvf /usr/local/loglistener-2.4.6.tar.gz /usr/local/loglistener-2.4.6
    mv /usr/local/loglistener-2.4.6 /usr/local/delete_old_loglistener
    mv /usr/local/loglistener-2.4.6.tar.gz /usr/local/delete_old_loglistener
    或者
    mkdir -p /usr/local/delete_old_loglistener
    tar cvf /usr/local/loglistener-2.8.2.tar.gz /usr/local/loglistener-2.8.2
    mv /usr/local/loglistener-2.8.2 /usr/local/delete_old_loglistener
    mv /usr/local/loglistener-2.8..tar.gz /usr/local/delete_old_loglistener

# 检查升级结果

在部署机上执行 /etc/init.d/loglistenerd -v
结果展示为loglistener 的正在运行的版本。

```
[root@VM_13_146_centos /usr/local/loglistener/tools]# /etc/init.d/loglistenerd -v
loglistener-2.8.8
[root@VM_13_146_centos /usr/local/loglistener/tools]#
```

# LogListener 配置指南

## 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 |
| --- | --- | --- |
| 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。<br>- 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。 |
| secret_id | API 密钥 ID。<br>请前往 API 密钥管理获取密钥 ID。 | LogListener 初始化时指定的密钥 ID。 |
| secret_key | API 密钥 Key。<br>请前往 API 密钥管理获取密钥 Key。 | LogListener 初始化时指定的密钥 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。<br>取值范围：true、false | 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。 |
| group_ip | IP 标识，机器组可通过所配 IP 标识与该机器关联。 | 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。<br>支持配置多个，由逗号分隔。<br>注意：<br>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 | - |

| 参数名 | 参数描述 | 默认值 |
| --- | --- | --- |
| | IP 关联。 | |
| max_connection | 同一时刻，最大的日志上传请求并行数量。<br>大于0的整数 | 20 |
| max_mem | 最大内存占用（Bytes）。<br>大于0的整数 | 2097152000 |
| max_dir | 最大监听目录数。<br>大于0的整数 | 5000 |
| max_file | 最大监听文件数。<br>大于0的整数 | 15000 |
| cpu_usage_thres | 单核最大 CPU 占用（%）。0代表无限制。<br>取值范围：0-100 | 0 |
| max_send_rate | 最大上传速率（Bytes/s）。0代表无限制。<br>大于0的整数 | 0 |
| 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。<br>大于0的整数 | 10 |
| request_compression | 是否压缩上传日志。<br>取值范围：true、false | true |
| 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。<br>取值范围：true、false | false |
| memory_tight_mode | 是否开启内存节省模式。<br>取值范围：true、false | 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。<br>取值范围：true、false | false |
| polling_mode | 是否开启 Polling 模式。该配置仅在 LogListener v3.0.0及以上版本 | true |

| 参数名 | 参数描述 | 默认值 |
| --- | --- | --- |
| | 支持。<br>取值范围：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。<br>大于0的整数 | 0 |
| max_send_retry_time | 日志上传请求失败并重试时，最大超时时间（s）。<br>大于0的整数 | 10800 |
| send_retry_period | 日志上传请求失败时，最大重试时间间隔（s）。<br>大于0的整数 | 3 |

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 start 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
```

# LogListener 服务日志

## 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，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：<br>- loglistener_status：对应 LogListener 的心跳状态日志。<br>- loglistener_alarm：对应 LogListener 的采集指标/错误类型监控日志。<br>- 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。 |

> 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。
> - cls_service_logging 为统一的 LogListener 服务日志的日志集。

## 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。
- 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 2.8.8 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。

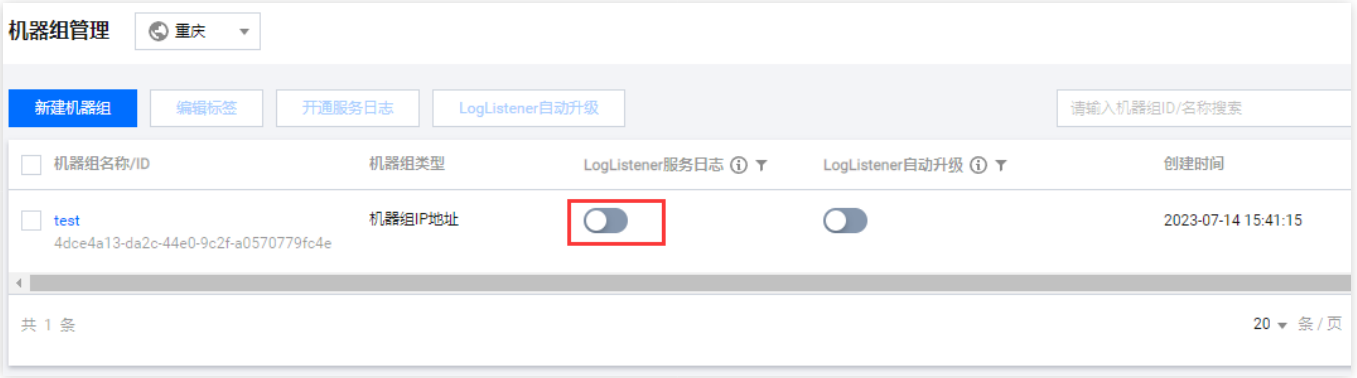
# 操作步骤

## 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。

# 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 |
| 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 |
| 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 |

| 字段 | 描述 |
| --- | --- |
| SendAvgCost | 平均发送耗时 |
| AvailConnNum | 可用连接数 |
| 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：

| 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 |
| 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 Memory 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 |

| Alarm Type | type ID | 描述 |
| --- | --- | --- |
| 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 |

| 字段 | 描述 |
| --- | --- |
| 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安装指南

LogListener 是日志服务 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 Linux 操作系统环境（暂不支持 Windows），并适配主流 Linux 操作系统版本。

| 操作系统类别 | 确定可安装环境 |
| --- | --- |
| 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 |
| 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 |
| Debian（64位） | Debian_8.2_64位、Debian_9.0_64位 |
| openSUSE（64位） | openSUSE_42.3_64位 |

## 安装启动

### 1. 下载安装 LogListener

LogListener 最新版本下载地址： 下载 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 loglistener/tools ，执行安装命令 。该操作命令如下：

```
wget http://cls-loglistener-1255000003.cos.shjr.cos.financetcloud.cn/loglistener-linux-x64-2.8.2.tar.gz && tar -zxvf loglistener-linux-x64-2.8.2.tar.gz -C /usr/local && cd /usr/local/loglistener-2.8.2/tools && ./loglistener.sh install
```

### 2. 初始化 LogListener

在 loglistener/tools 路径下，以 root 权限执行 LogListener 初始化命令（默认使用内网方式访问服务），初始化命令如下：

```
./loglistener.sh init -secretid AKIDPEtPyKabfW8Z3Uspdz83xxxxxxxxxxx -secretkey whHwQfjdLnzzCE1jIf09xxxxxxxxxxx -primary_region xxxxxx -host cls.api3.finance.cloud.tencent.com -port 80
```

> 说明：
> 初始化命令中 -secretid、-secretkey、-primary_region为需要自主填写的参数，详细介绍请见如下 参数说明。

参数说明

| 参数名 | 类型描述 |
| --- | --- |
| secretid | 云 API 密钥 的一部分，SecretId 用于标识 API 调用者身份 |
| secretkey | 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 |
| primary_region | primary_region 表示日志服务所在的地域 |
| ip | 机器的 ip 标识。若不填写，loglistener 会自动获取本机的 ip 地址 |
| label | 机器组标示，标示机器组需要填写标示信息，多个标示按逗号分隔 |

### 3. 启动 LogListener

成功安装后，执行 LogListener 启动命令：

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

# LogListener 常用操作

> 说明：
> 本文档示例的操作命令说明仅适用于 LogListener-2.8.2 及以上版本，低版本操作命令请参见 低版本 LogListener 安装指南。

1. 查看 LogListener 版本

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

2. 查看 LogListener 帮助文档

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

3. LogListener 进程管理

   /etc/init.d/loglistenerd (start|restart|stop) # 启动、重启、停止

4. 查看 LogListener 进程状态

   /etc/init.d/loglistenerd status

   LogListener 正常情况会运行两个进程：

5. 检查 LogListener 心跳及配置

   /etc/init.d/loglistenerd check

# 卸载 LogListener

以管理员权限执行 loglistener/tools 目录下的卸载命令：

./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. 备份旧版本中的断点文件目录（ loglistener/data ）。例如，将旧版的断点文件备份至 /tmp/loglistener-backup 目录下。

   cp -r loglistener-2.8.2/data /tmp/loglistener-backup/

3. 使用卸载命令卸载旧版本的 LogListener。

4. 下载最新版本的 LogListener，并使用相关命令安装和初始化新版本 LogListener。

5. 复制所备份的断点文件目录（步骤2）到新版本 LogListener 目录下。例如：

```
cp -r /tmp/loglistener-backup/data loglistener-2.8.2/
```

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. 下载最新版本的 LogListener，并使用相关命令安装和初始化新版本 LogListener。
4. 使用启动命令启动运行新版本 LogListener。

# LogListener 采集配置导入

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。

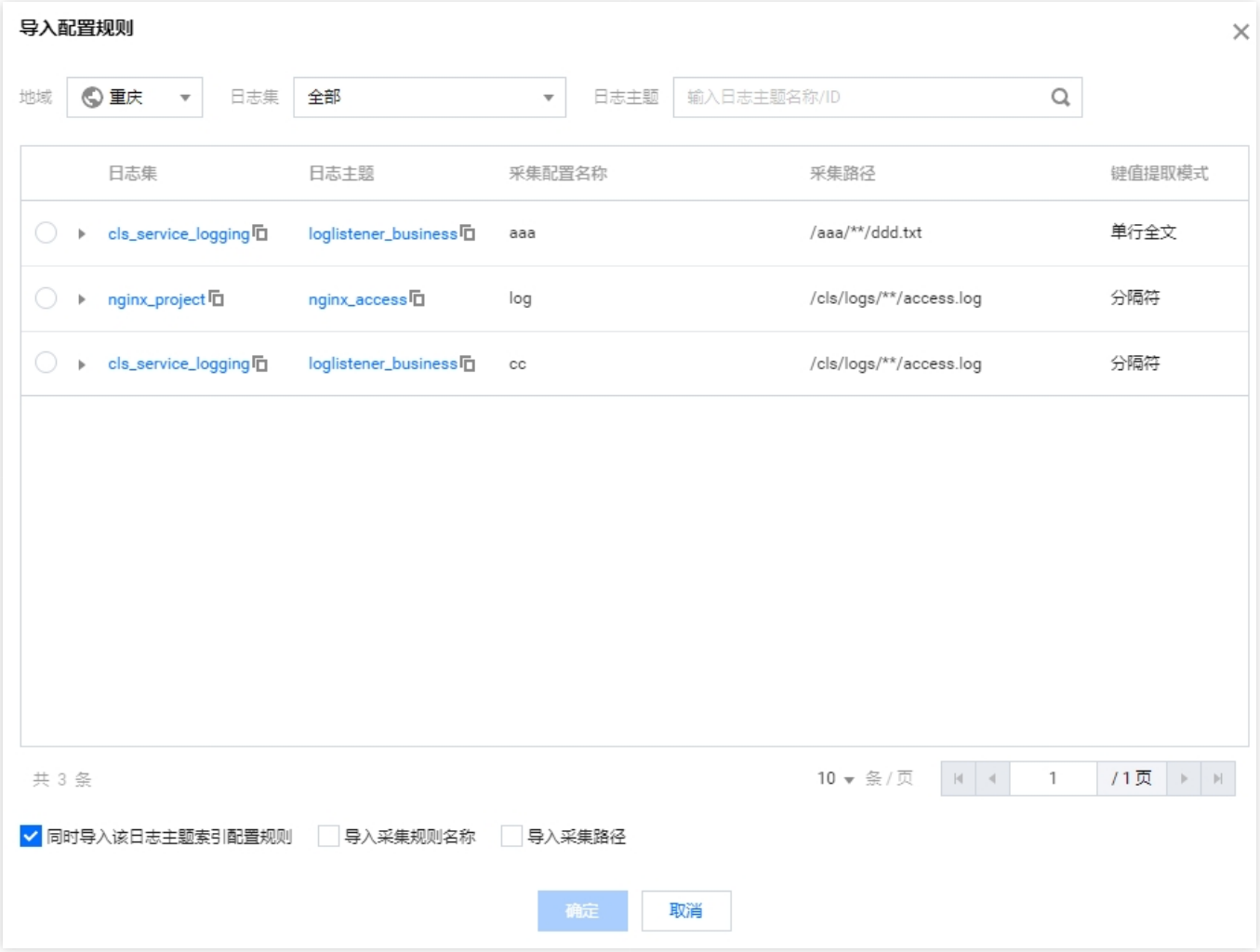
## 操作场景

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采集配置区域中，单击新增，进入日志数据源页面。
6. 单击要采集的数据源，进入采集配置流程。
7. 选择机器组，单击下一步。
8. 单击导入配置规则，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。




9. 单击确定，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、分隔符格式等模式采集文本日志。

# 采集文本日志
# 单行全文模式

## 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或自建 K8s 已安装 LogListener。
- LogListener 版本为2.1.1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：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。

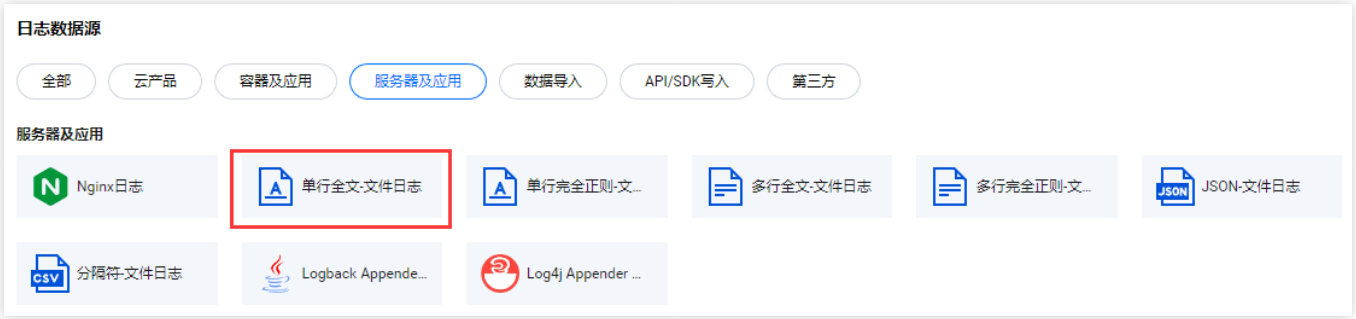
### 步骤2：创建日志主题

1. 单击创建日志主题。
2. 在弹出的对话框中，将“日志主题名称”填写为“test_full”，单击确定，即可新增日志主题。




## 步骤3：机器组管理

1. 日志主题创建成功后，进入该日志主题管理页面。
2. 选择采集配置页签，在 LogListener 采集配置中单击新增，并在服务器及应用栏中选择单行全文 - 文件日志。



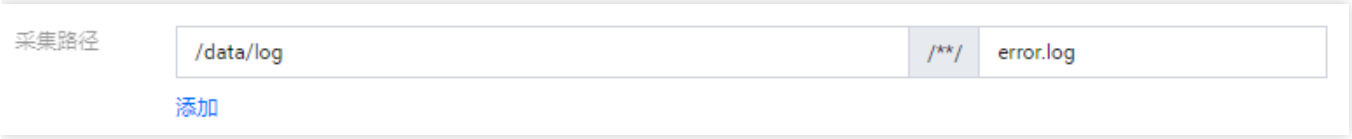

3. 在“机器组管理”步骤，勾选需要与当前日志主题进行绑定的机器组，单击下一步。

## 步骤4：采集配置

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”。

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 ？<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>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 ？<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>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

> - 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。

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。

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。

- 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
  >
  > - 过滤规则 “不等于” 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<br>- 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 |

## 步骤5：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。
   - 索引状态：确认是否开启。
   - 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。
   - 键值索引：默认关闭，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，可将  设置为  。

     > 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。

# 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，单击检索分析，即可开始按照设定的查询条件检索日志。

# 多行全文模式

## 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或自建 K8s 已安装 LogListener。
- LogListener 版本为2.1.1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：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：创建日志主题

1. 单击创建日志主题。
2. 在弹出的对话框中，输入相关信息，单击确定，即可新增日志主题。

### 步骤3：机器组管理

1. 日志主题创建成功后，进入该日志主题管理页面。
2. 选择采集配置页签，在 LogListener 采集配置中单击新增，并在服务器及应用栏中选择多行全文 - 文件日志。
3. 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。

### 步骤4：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： [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 ？<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>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 ？<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>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 |

| 序号 | 目录前缀表达式 | 文件名表达式 | 说明 |
| --- | --- | --- | --- |
| 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
>
> - 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。

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。




   - 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。




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 htt
p://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
>
> - 过滤规则 “不等于” 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

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<br>- 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。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：<br>Loglistener 2.8.8及以上版本才可以配置合并解析失败日志。<br><br>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 | 开启/关闭 |

| 名称 | 描述 | 配置项 |
| --- | --- | --- |
| | 匹配失败的单条多行日志被拆分至多条日志 | |

### 步骤5：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。
   - 索引状态：确认是否开启。

     > 注意
     >
     > 检索必须开启索引配置，否则无法检索。

   - 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。
   - 键值索引：默认关闭，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，可将  设置为  。
3. 单击提交，完成采集配置。

## 相关操作

检索日志，请参见 检索分析 文档。

# 单行-完全正则模式

## 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或自建 K8s 已安装 LogListener。
- LogListener 版本为2.2.2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：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：新增日志主题

1. 单击创建日志主题。
2. 在弹出的对话框中，输入相关信息，单击确定，即可新增日志主题。

## 步骤3：机器组管理

1. 日志主题创建成功后，进入该日志主题管理页面。
2. 选择采集配置页签，在 LogListener 采集配置中单击新增，并在服务器及应用栏中选择单行完全正则 - 文件日志。
3. 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。

## 步骤4：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：[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 ？<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>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 ？<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>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 |
| --- | --- | --- | --- |

| 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
>
> - 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。

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

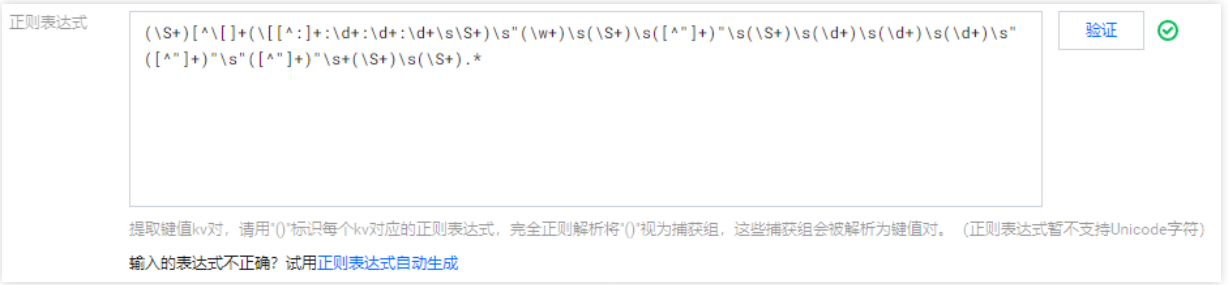
1. 在采集配置页面，将“提取模式”设置为单行-完全正则，并在“日志样例”文本框中，输入日志样例。




2. 根据如下规则，定义正则表达式。

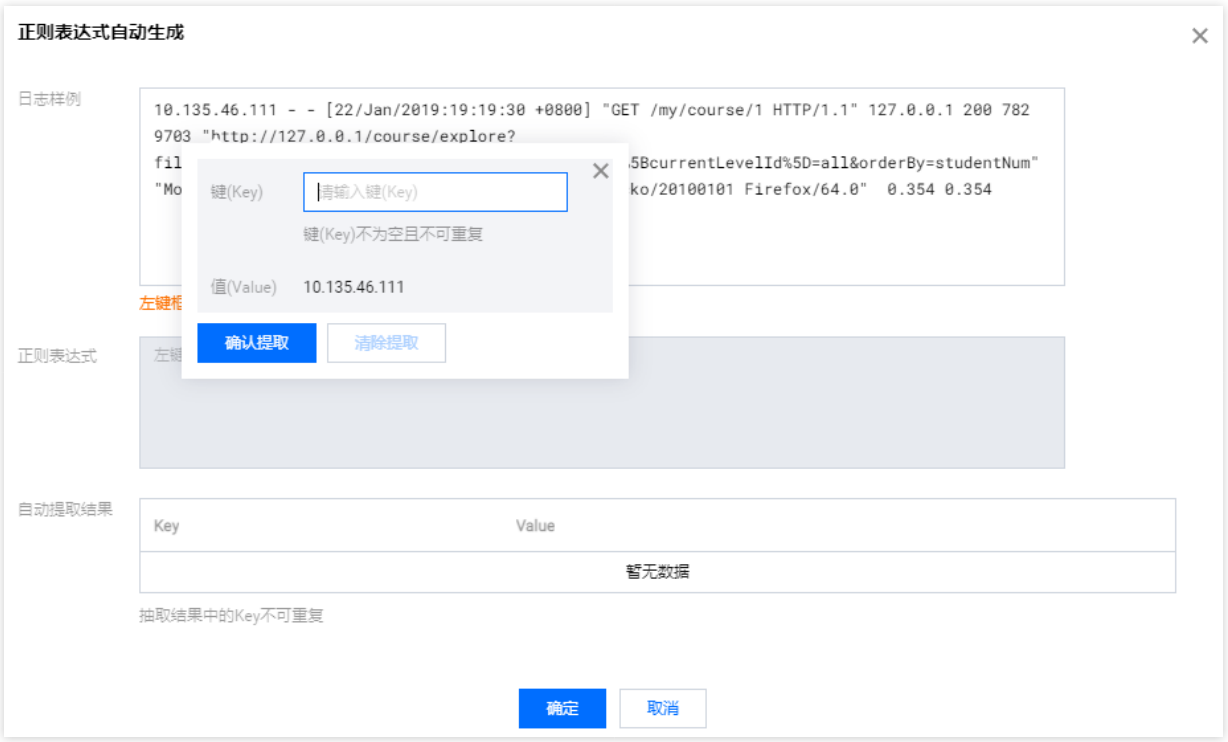
   系统有手动模式和自动模式两种方式定义正则表达式。您可手动输入表达式提取 key-value 进行验证，也可单击正则表达式自动生成切换为自动模式。系统会根据您选择的模式以及定义好的正则表达式，提取 key-value 进行正则表达式的验证。

   - 手动模式：




     a. 在“正则表达式”的文本框中，输入正则表达式。
     b. 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（单击正则表达式自动生成进行切换）：
     a. 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。

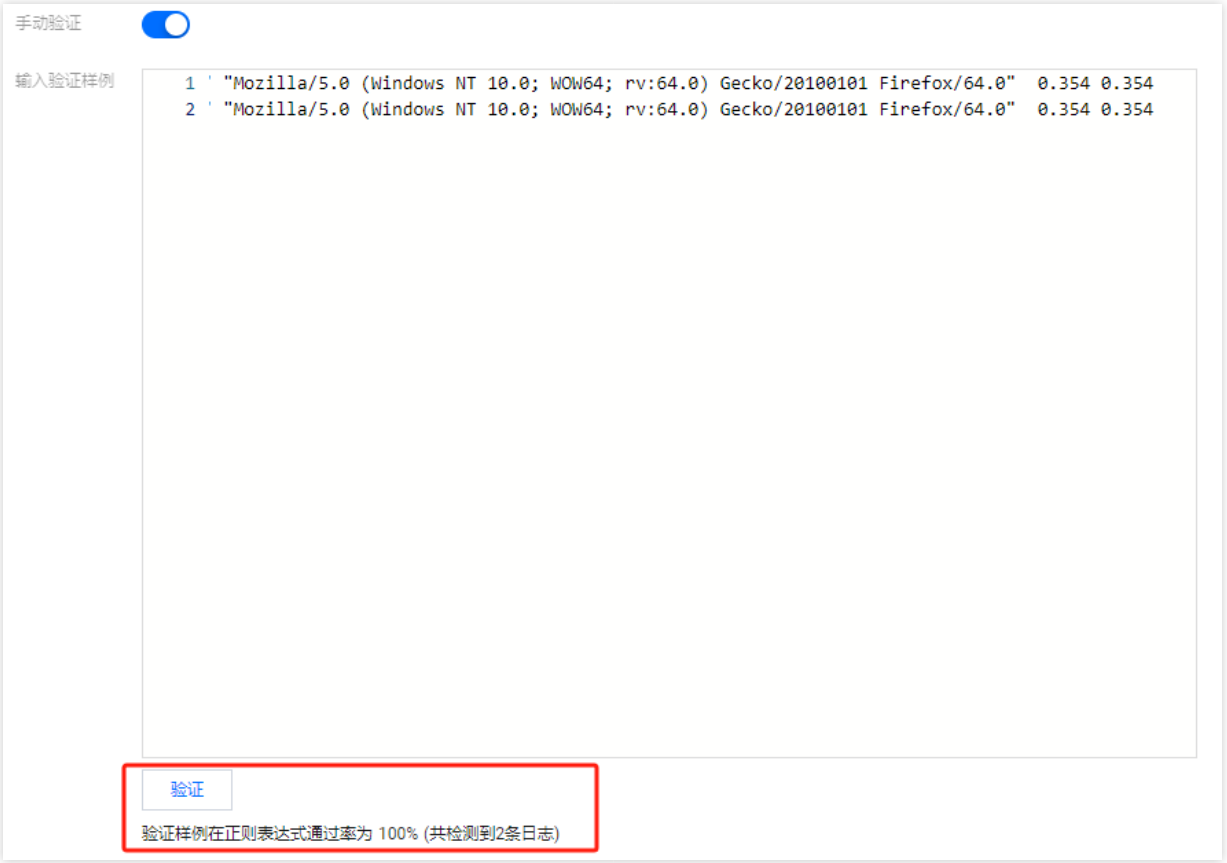
b. 重复 自动模式的步骤（1），直到提取完所有的 key-value 对。

c. 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。

手动验证

1. 当您的日志数据复杂时，可以将"手动验证"设置为  ，即可开启手动验证。
2. 输入多个日志样例，单击验证。系统将验证样例正则表达式的通过率。




配置采集时间

- 日志时间单位为：毫秒。
- 日志的时间属性有如下方式：
  - 日志采集时间：默认作为日志的时间属性。日志的时间属性由日志服务 CLS 采集该条日志的时间决定。
  - 指定日志字段：填写原始时间戳的时间键以及对应的时间解析格式。日志的时间属性由原始日志中时间戳决定。
    时间格式解析规则填写的示例如下：
    - 示例1：
      日志样例原始时间戳： 10/Dec/2017:08:00:00.000 ，解析格式为： %d/%b/%Y:%H:%M:%S.%f 。
    - 示例2：
      日志样例原始时间戳： 2017-12-10 08:00:00.000 ，解析格式为： %Y-%m-%d %H:%M:%S.%f 。

- 示例3：
  日志样例原始时间戳： 12/10/2017, 08:00:00.000 ，解析格式为： %m/%d/%Y, %H:%M:%S.%f 。

> 注意
>
> 日志时间支持以毫秒为单位，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
>
> - 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

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<br>- 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 |

### 步骤5：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。
   - 索引状态：确认是否开启。

     > 注意
     >
     > 检索必须开启索引配置，否则无法检索。

   - 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。
   - 键值索引：默认关闭，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，可将  设置为  。
3. 单击提交，完成采集配置。

## 相关操作

检索日志，请参见 检索分析 文档。

# 多行-完全正则模式

## 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或自建 K8s 已安装 LogListener。
- LogListener 版本为2.4.5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
```

## 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：新增日志主题

1. 单击创建日志主题。
2. 在弹出的对话框中，输入相关信息，单击确定，即可新增日志主题。

### 步骤3：机器组管理

1. 日志主题创建成功后，进入该日志主题管理页面。
2. 选择采集配置页签，在 LogListener 采集配置中单击新增，并在服务器及应用栏中选择多行完全正则 - 文件日志。
3. 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。

### 步骤4：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： [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 ？<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>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 ？<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>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 |

| 序号 | 目录前缀表达式 | 文件名表达式 | 说明 |
| --- | --- | --- | --- |
| 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。

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

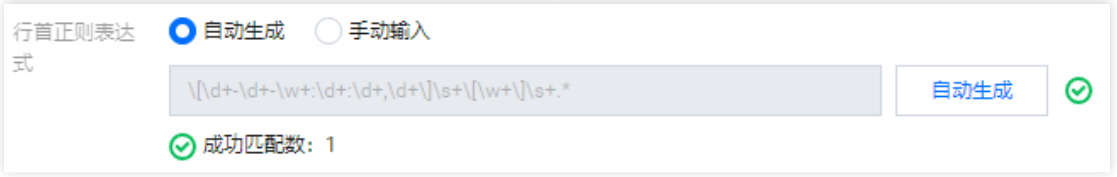
1. 在“采集配置”页面，将“提取模式”设置为多行-完全正则，并在“日志样例”文本框中，输入日志样例。




2. 根据如下规则，定义首行正则表达式。
   您可选择自动生成或者手动输入两种方式定义行首正则表达式，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：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。



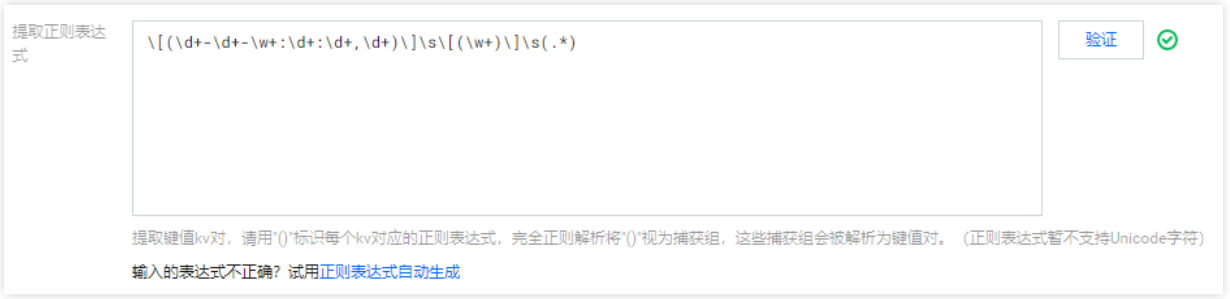

- 手动输入：在文本框中，手动输入行首正则表达式，单击验证，系统将判断表达式是否通过。




3. 提取正则表达式

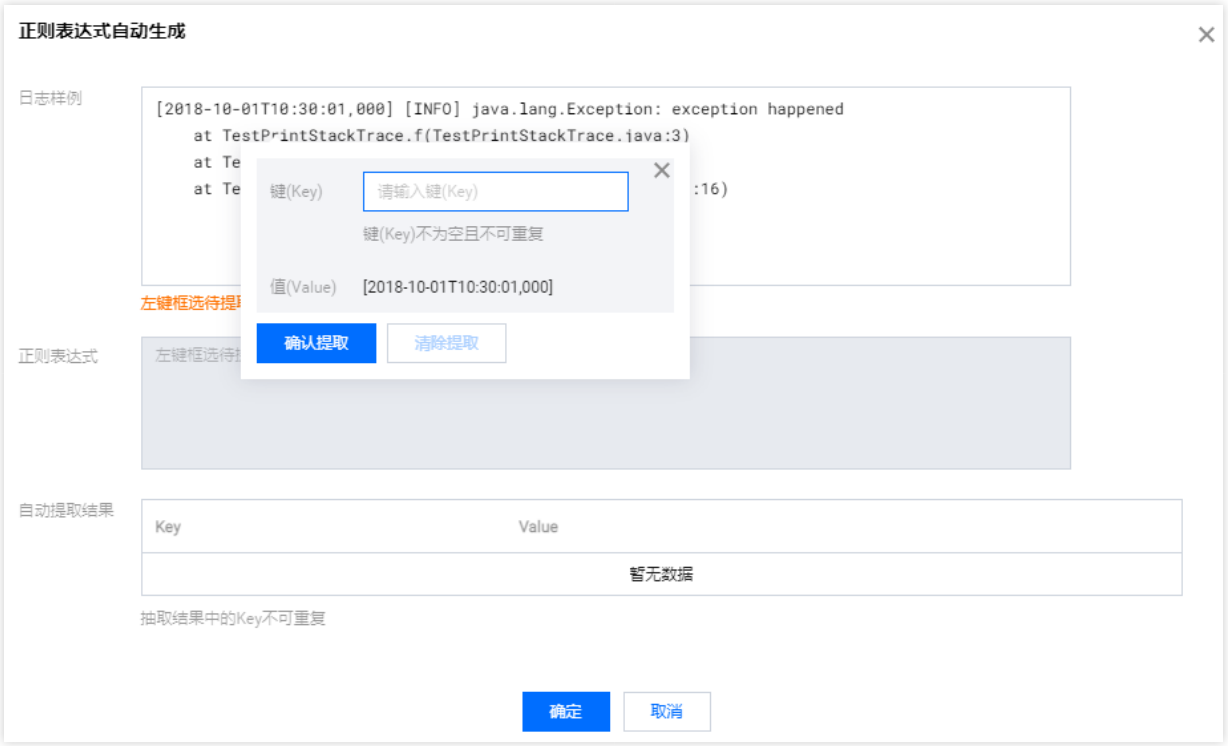
系统有手动模式和自动模式两种方式定义正则表达式。您可手动输入表达式提取 key-value 进行验证，也可单击正则表达式自动生成切换为自动模式。系统会根据您选择的模式以及定义好的正则表达式，提取 key-value 进行正则表达式的验证。

- 手动模式：




  a. 在"正则表达式"的文本框中，输入正则表达式。
  b. 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（单击正则表达式自动生成进行切换）：
  a. 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。




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。

b. 重复自动模式的步骤1，直到提取完所有的 key-value 对。

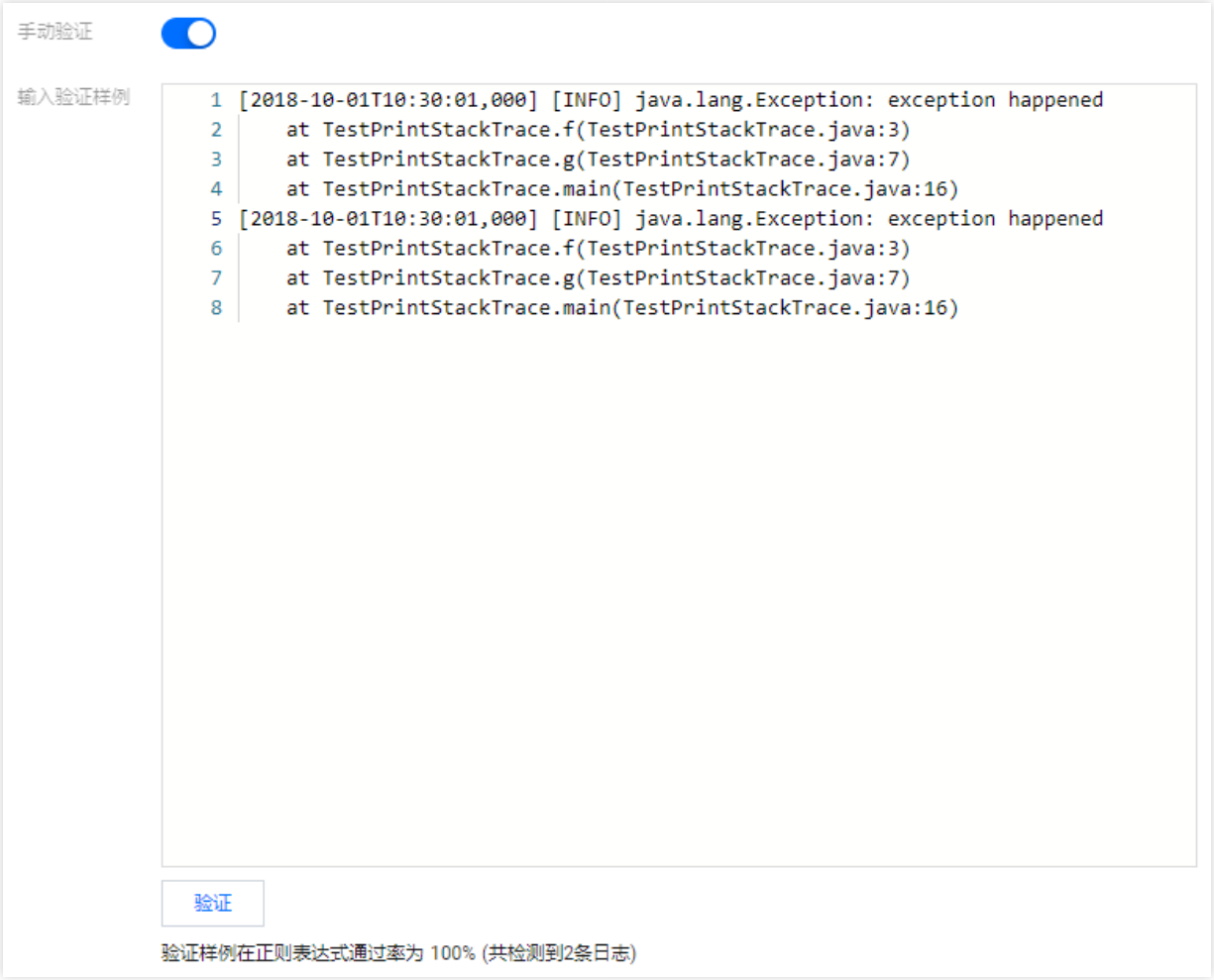
c. 单击确定，系统将根据提取好的 key-value 对自动生成完整的正则表达式。

> 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。

手动验证



1. 当您的日志数据复杂时，可以将"手动验证"设置为 ，即可开启手动验证。
2. 输入多个日志样例，单击验证。系统将验证样例正则表达式的通过率。




配置采集时间

- 日志时间单位为：毫秒。
- 日志的时间属性有如下方式：
  - 日志采集时间：默认作为日志的时间属性。日志的时间属性由日志服务 CLS 采集该条日志的时间决定。
  - 指定日志字段：填写原始时间戳的时间键以及对应的时间解析格式。日志的时间属性由原始日志中时间戳决定。
    时间格式解析规则填写的示例如下：
    - 示例1：
      日志样例原始时间戳： 10/Dec/2017:08:00:00.000 ，解析格式为： %d/%b/%Y:%H:%M:%S.%f 。
    - 示例2：
      日志样例原始时间戳： 2017-12-10 08:00:00.000 ，解析格式为： %Y-%m-%d %H:%M:%S.%f 。
    - 示例3：
      日志样例原始时间戳： 12/10/2017, 08:00:00.000 ，解析格式为： %m/%d/%Y, %H:%M:%S.%f 。

> 注意
>
> 日志时间支持以毫秒为单位，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
>
> - 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

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<br>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<br>- 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。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：<br>Loglistener 2.8.8及以上版本才可以配置合并解析失败日志。<br><br>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5：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。
   - 索引状态：确认是否开启。

     > 注意
     >
     > 检索必须开启索引配置，否则无法检索。

   - 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。
   - 键值索引：默认关闭，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，可将  设置为  。
3. 单击提交，完成采集配置。

## 相关操作

检索日志，请参见 检索分析 文档。

# JSON 模式

## 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或自建 K8s 已安装 LogListener。
- LogListener 版本为2.4.5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：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：创建日志主题

1. 单击创建日志主题。
2. 在弹出的对话框中，输入相关信息，单击确定，即可新增日志主题。

## 步骤3：机器组管理

1. 日志主题创建成功后，进入该日志主题管理页面。
2. 选择采集配置页签，在 LogListener 采集配置中单击新增，并在服务器及应用栏中选择JSON - 文件日志。
3. 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。

## 步骤4：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： [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 |
| 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。

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

在采集配置页面，将“提取模式”设置为 JSON。

配置采集时间

- 日志时间单位为：毫秒。
- 日志的时间属性有如下方式：
  - 日志采集时间：默认作为日志的时间属性。日志的时间属性由日志服务 CLS 采集该条日志的时间决定。
  - 指定日志字段：填写原始时间戳的时间键以及对应的时间解析格式。日志的时间属性由原始日志中时间戳决定。
    时间格式解析规则填写的示例如下：
    - 示例1：
      日志样例原始时间戳： 10/Dec/2017:08:00:00.000 ，解析格式为： %d/%b/%Y:%H:%M:%S.%f 。
    - 示例2：
      日志样例原始时间戳： 2017-12-10 08:00:00.000 ，解析格式为： %Y-%m-%d %H:%M:%S.%f 。
    - 示例3：
      日志样例原始时间戳： 12/10/2017, 08:00:00.000 ，解析格式为： %m/%d/%Y, %H:%M:%S.%f 。

> 注意
>
> 日志时间支持以毫秒为单位，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
>
> - 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

通过勾选，选择您需要定义的高级配置。
JSON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<br>- 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 |

## 步骤5：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。
   - 索引状态：确认是否开启。

     > 注意
     >
     > 检索必须开启索引配置，否则无法检索。

   - 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\\ ，确认是否需要开启。
   - 键值索引：默认开启，您可按需进行字段类型、分词符以及是否开启统计分析的配置。若您需要关闭键值索引，可将  设置为  。

3. 单击提交，完成采集配置。

# 相关操作

检索日志，请参见 检索分析 文档。

# 分隔符模式

## 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或自建 K8s 已安装 LogListener。
- LogListener 版本为2.4.5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 ::: 64
7 ::: 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：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：创建日志主题

1. 单击创建日志主题。
2. 在弹出的对话框中，输入相关信息，单击确定，即可新增日志主题。

### 步骤3：机器组管理

1. 日志主题创建成功后，进入该日志主题管理页面。
2. 选择采集配置页签，在 LogListener 采集配置中单击新增，并在服务器及应用栏中选择分隔符 - 文件日志。
3. 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。

### 步骤4：采集配置

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： [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 ？<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>- 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 ？ ，<br>- * 表示匹配多个任意字符<br>- ？表示匹配单个任意字符<br>- 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 |

| 序号 | 目录前缀表达式 | 文件名表达式 | 说明 |
| --- | --- | --- | --- |
| 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
>
> - 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。

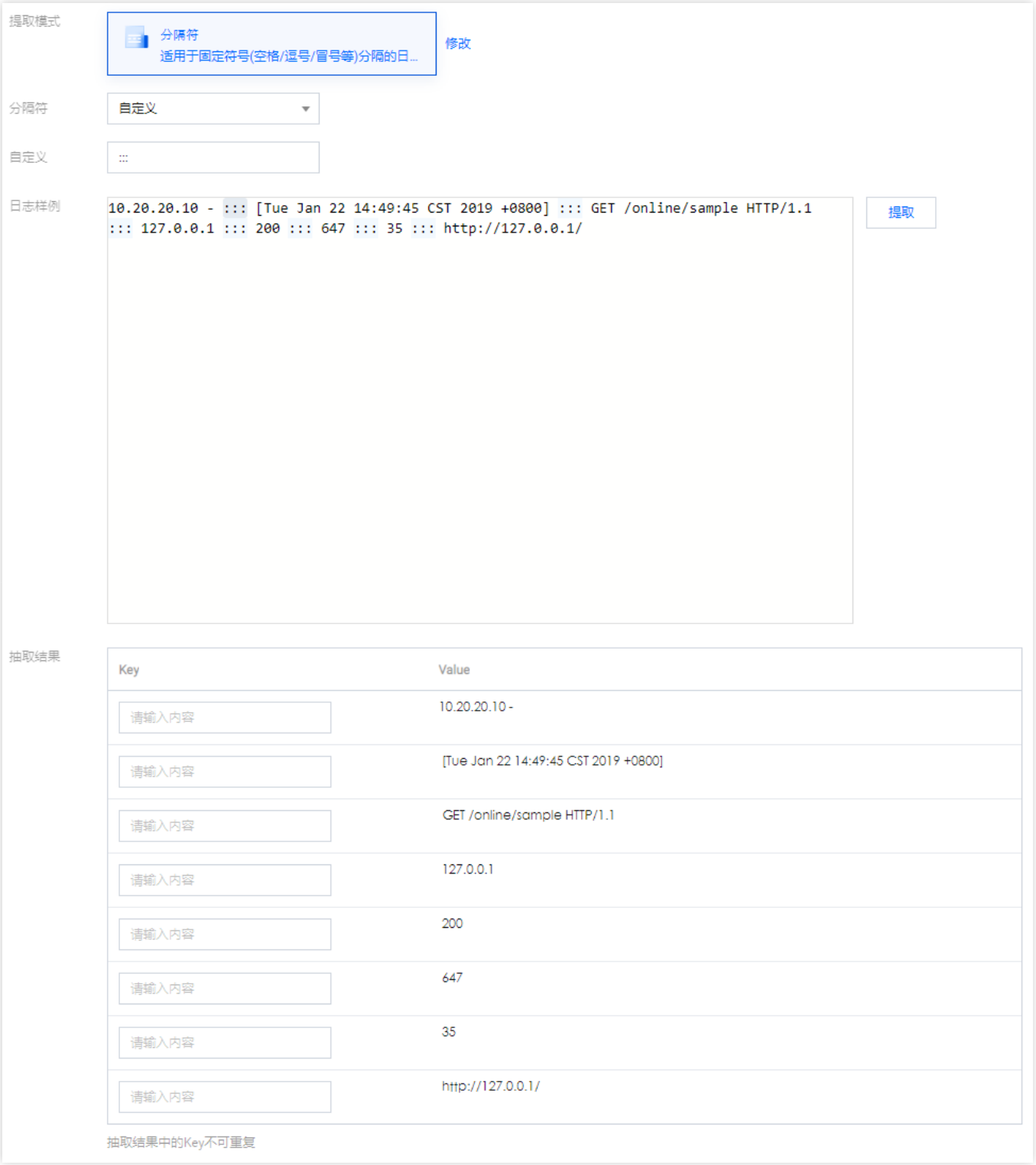
编码模式

- 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。




配置采集时间

- 日志时间单位为：毫秒。
- 日志的时间属性有如下方式：
  - 日志采集时间：默认作为日志的时间属性。日志的时间属性由日志服务 CLS 采集该条日志的时间决定。
  - 指定日志字段：填写原始时间戳的时间键以及对应的时间解析格式。日志的时间属性由原始日志中时间戳决

定。

时间格式解析规则填写的示例如下：

- 示例1：
  日志样例原始时间戳： 10/Dec/2017:08:00:00.000 ，解析格式为： %d/%b/%Y:%H:%M:%S.%f 。
- 示例2：
  日志样例原始时间戳： 2017-12-10 08:00:00.000 ，解析格式为： %Y-%m-%d %H:%M:%S.%f 。
- 示例3：
  日志样例原始时间戳： 12/10/2017, 08:00:00.000 ，解析格式为： %m/%d/%Y, %H:%M:%S.%f 。

> 注意
>
> 日志时间支持以毫秒为单位，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
  >
  > - 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

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<br>- 自定义：自定义日志文件的超时时 |

| 名称 | 描述 | 配置项 |
| --- | --- | --- |
| | | 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 |

### 步骤5：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。
   - 索引状态：确认是否开启。

     > 注意
     >
     > 检索必须开启索引配置，否则无法检索。

   - 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。
   - 键值索引：默认关闭，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，可将  设置为  。
3. 单击提交，完成采集配置。

## 相关操作

检索日志，请参见 检索分析 文档。

# 组合解析模式

## 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 或 自建 K8s 已安装 LogListener。
- LogListener 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：

```
157139445,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"]
          }
        },
        {
          "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 |

| 插件功能 | 插件名称 | 功能 |
| --- | --- | --- |
| 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 |
| processor_multiline_fullregex | 是 | ExtractRegex | 是 | 定义提取到多行日志后的提取正则 |
| processor_multiline_fullregex | 是 | ExtractKeys | 是 | 定义提取键值 |
| processor_fullregex | 是 | ExtractRegex | 是 | 定义提取正则 |
| processor_fullregex | 是 | 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 |
| processor_json | 是 | 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 |
| processor_split_delimiter | 是 | KeepSource | 否 | 最终键值名称中，是否保留 SourceKey |

| 插件名称 | 是否支持子项解析 | 插件参数 | 是否必须 | 功能 |
| --- | --- | --- | --- | --- |
| processor_split_delimiter | 是 | Delimiter | 是 | 指定分隔符（单/多字符） |
| processor_split_delimiter | 是 | 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 |
| processor_split_key_value | 否 | KeepSource | 否 | 最终键值名称中，是否保留 SourceKey |
| processor_split_key_value | 否 | 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 |
| processor_timeformat | 否 | 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：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：创建日志主题

1. 单击创建日志主题。
2. 在弹出的对话框中，输入相关信息，单击确定。

## 步骤3：机器组管理

1. 日志主题创建成功后，进入该日志主题管理页面。

2. 选择采集配置页签，在 LogListener 采集配置中单击新增，并在服务器及应用栏中选择任意数据源。
3. 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。

## 步骤4：采集配置

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 |
| 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。

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

在采集配置页面，将“提取模式”设置为组合解析。

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 |
| --- | --- | --- |

| 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<br>- 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 |

# 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，单击检索分析，即可开始按照设定的查询条件检索日志。

# 配置时间格式

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 |

| 参数格式 | 说明 | 示例 |
| --- | --- | --- |
| %f | 毫秒时间 | 0.123 |
| %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 |

| 时间表示示例 | 时间提取格式 |
| --- | --- |
| 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 |

# 采集 Nginx 访问日志

## 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1：创建日志主题

1. 登录日志服务控制台。
2. 在左侧导航栏中，单击日志主题，进入日志主题管理页面。
3. 选择日志主题的地域，单击创建日志主题。
4. 在弹出的创建日志主题窗口中，填写相关信息，单击确定。

### 步骤2：机器组管理

1. 日志主题创建成功后，进入该日志主题管理页面。
2. 选择采集配置页签，在 LogListener 采集配置中单击新增，并在服务器及应用栏中选择Nginx日志。

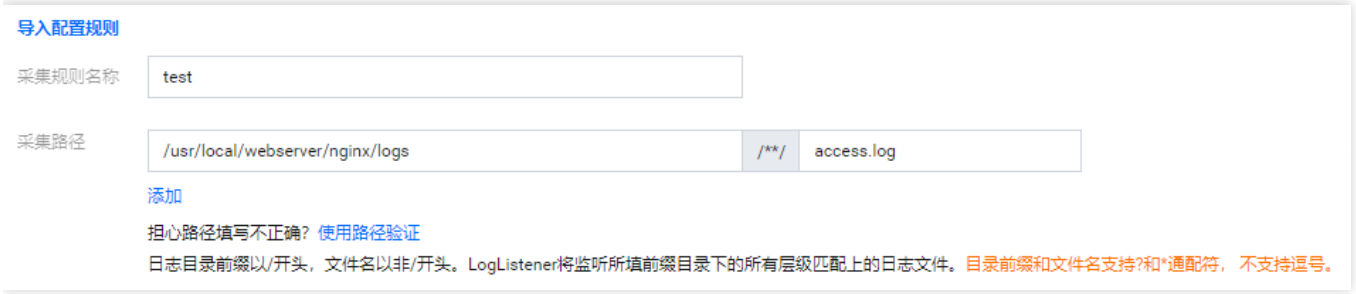



3. 在“机器组管理”步骤，勾选需要与当前日志主题进行绑定的机器组，单击下一步。

## 步骤3：增加采集配置

### 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”。
日志采集路径格式： [目录前缀表达式]/**/[文件名表达式] 。
例如，采集本地路径为 /usr/local/webserver/nginx/logs/access.log 的日志文件。




### 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。

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"';
```

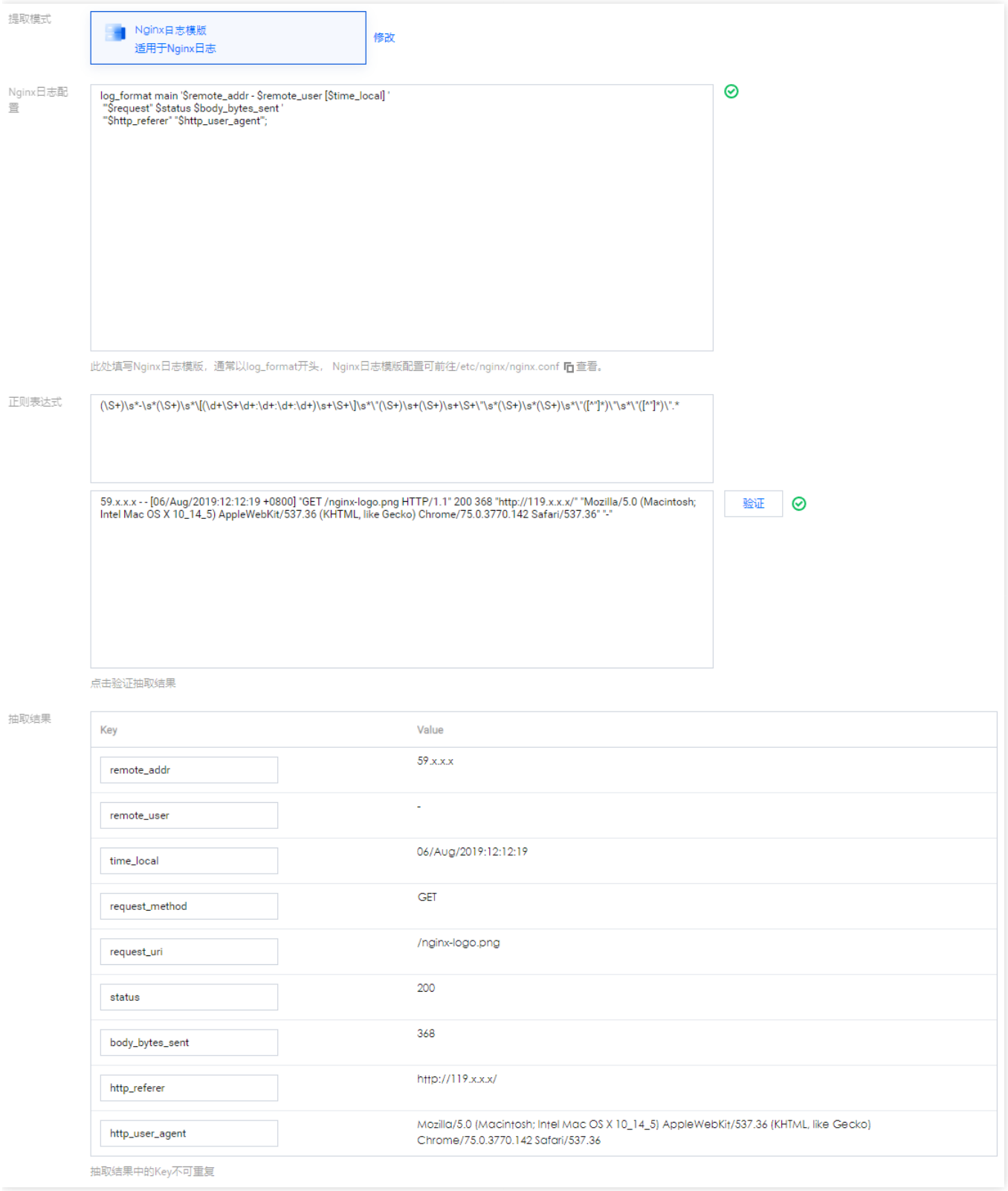
CLS 将根据上面的 Nginx 日志模板自动生成日志提取正则表达式，例如：

```
(\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/" "Mozill
a/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.142 S
afari/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：




## 配置采集时间

- 日志时间单位为：毫秒。
- 日志的时间属性有如下方式：
  - 日志采集时间：默认作为日志的时间属性。日志的时间属性由日志服务 CLS 采集该条日志的时间决定。
  - 指定日志字段：填写原始时间戳的时间键以及对应的时间解析格式。日志的时间属性由原始日志中时间戳决

定。

时间格式解析规则填写的示例如下：

- 示例1：
  日志样例原始时间戳： 10/Dec/2017:08:00:00.000 ，解析格式为： %d/%b/%Y:%H:%M:%S.%f 。
- 示例2：
  日志样例原始时间戳： 2017-12-10 08:00:00.000 ，解析格式为： %Y-%m-%d %H:%M:%S.%f 。
- 示例3：
  日志样例原始时间戳： 12/10/2017, 08:00:00.000 ，解析格式为： %m/%d/%Y, %H:%M:%S.%f 。

> 注意
>
> 日志时间支持以毫秒为单位，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

样例日志使用Nginx 日志模板模式解析后，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。

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

通过勾选，选择您需要定义的高级配置。
Nginx 日志模板提取模式下，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<br>- 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 |

## 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。

- 索引状态：确认是否开启。

  > 注意
  >
  > 检索必须开启索引配置，否则无法检索。

- 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\\ ，确认是否需要开启。
- 键值索引：默认开启，您可按需进行字段类型、分词符以及是否开启统计分析的配置。若您需要关闭键值索引，可将  设置为  。

3. 单击提交，完成采集配置。

## 步骤5：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。

# 使用 Kafka 协议上传日志

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。

- 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。
- 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。

## 配置方式

使用 Kafka 协议上传日志至 CLS Kafka 生产端时，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。 |

| 参数 | 说明 |
| --- | --- |
| 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 ${SecurityId}#${SecurityKey} 。例如：XXXXXXXXXXXXXX#YYYYYYYY。<br>- 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。<br>注意：<br>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。 详情请参考 管理日志主题。 |

# 服务入口

> 注意：
>
> 本文档以城市1地域为例，内外网域名需使用不同端口标识，其他地域请替换地址前缀。

host（服务入口）填写样例：

| 地域 | 网络类型 | 端口号 | 服务入口 |
| --- | --- | --- | --- |
| 城市1 | 内网 | 9095 | tcloud-cls-kafka-access.{{regionName}}.{{domainMain}} |

# 示例

## Beat 调用示例

filebeat/winlogbeat 配置

```
output.kafka:
  enabled: true
  hosts: ["tcloud-cls-kafka-access.{{regionName}}.{{domainMain}}:9095"] # TODO 服务地址；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: "${SecurityId}#${SecurityKey}"
```

## Logstash 调用示例

### logstash 配置

```
output {
  kafka {
    topic_id => "${ClstopicID}"
    bootstrap_servers => "tcloud-cls-kafka-access.{{regionName}}.{{domainMain}}:${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
='${ClslogsetID}' password='${securityID}#${securityKEY}';"
    codec => json
  }
}
```

## 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 ${SecurityId}#${SecurityKey}，若要匿名写入，格式为 topic_id#${日志主题 ID}
    config.Net.SASL.Password = "${SecurityId}#${SecurityKey}"
    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；内网端口9095
    producer, err := sarama.NewSyncProducer([]string{"tcloud-cls-kafka-access.{{regionName}}.{{domain
Main}}: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；内网端口9095
        bootstrap_servers=["tcloud-cls-kafka-access.{{regionName}}.{{domainMain}}:9095"],
        security_protocol='SASL_PLAINTEXT',
        sasl_mechanism='PLAIN',
        # TODO 日志集 ID
        sasl_plain_username='${logsetID}',
        # TODO 格式为 ${SecurityId}#${SecurityKey}, 若要匿名写入，格式为 topic_id#${日志主题 ID}
        sasl_plain_password='${SecurityId}#${SecurityKey}',
        api_version=(0, 11, 0),
        # TODO 配置压缩方式
        compression_type="${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, TimeoutExce
ption {
        // 0.配置一系列参数
        Properties props = new Properties();
        // TODO 使用时
        props.put("bootstrap.servers", "tcloud-cls-kafka-access.{{regionName}}.{{domainMain}}: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securityID和securityKEY的组合securityID#securityKEY,格式为 ${SecurityId}#${SecurityKey},
        // 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='${SecurityId}#${SecurityKey}';");

        // 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 "tcloud-cls-kafka-access.{{regionName}}.{{domainMain}}:${port}"
// USERNAME 为日志集 ID
#define USERNAME "${logsetID}"
// PASSWORD 格式为 ${SecurityId}#${SecurityKey}, 若要匿名写入，格式为 topic_id#${日志主题 ID}
#define PASSWORD "${SecurityId}#${SecurityKey}"
// 日志主题ID
#define TOPIC "${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->partiti
on);
    }
}

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_KAF
KA_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_CO
NF_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_O
K) {
        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_O
K) {
        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
            {
              //  Kafka 填写，注意内网端口9095
                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 ${SecurityId}#${SecurityKey},
                // 若要匿名写入，格式为 topic_id#${日志主题 ID}
                SaslPassword = "${SecurityId}#${SecurityKey}",
                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}" : $"D
elivery 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}");
            }
        }
    }
}
```

# 匿名日志采集

CLS 日志服务支持免密通过 HTTP 协议将日志上传至 CLS。

## 使用场景

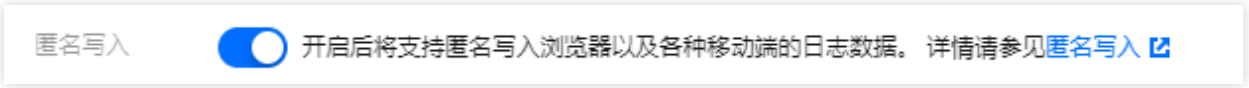
- 用户使用的浏览器、操作系统、分辨率等信息。
- 用户浏览行为记录（例如：用户在网站上的点击行为等信息）。
- 用户在 App 中的停留时间、是否活跃等信息。

## 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。
3. 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。
4. 在日志主题创建或编辑弹窗中，找到匿名写入配置项， 并开启开关。




### 步骤二：上传日志

开通匿名写入后，您可以通过以下方法上传日志到日志主题中。

方法1：通过 HTTP GET 请求上传日志

```
curl --request GET 'http://data.tcloud-cls-cgi.{{regionName}}.{{domainMain}}:8028/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。例如：ap-city |
| ${topic_id} | 是 | 日志主题 ID |

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 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2：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。

  请求语法

```
POST http://data.tcloud-cls-cgi.{{regionName}}.{{domainMain}}:8028/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。例如：ap-city |
| ${topic_id} | 是 | 日志主题 ID |

示例

```
POST /tracklog?topic_id={topic_id} HTTP/1.1
Host:data.tcloud-cls-cgi.{{regionName}}.{{domainMain}}:8028
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"
```

```
}
```

# 使用API上传日志

## 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: data.tcloud-cls-cgi.agent.{{regionName}}.{{domainMain}}:80
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：

- 内网域名格式形如 data.tcloud-cls-cgi.agent.{{regionName}}.{{domainMain}}:80 ，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。
  region 字段是日志服务区域简称，例如城市1区域填写 ap-city1 。

```
ap-city1 - 城市1
ap-city2 - 城市2
ap-city3 - 城市3
ap-city4 - 城市4
...
```

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）。
  示例：

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: data.tcloud-cls-cgi.agent.{{regionName}}.{{domainMain}}:80
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。
  示例：

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: data.tcloud-cls-cgi.agent.{{regionName}}.{{domainMain}}:80
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf

<LogGroupList 的 PB 格式打包内容>
```

# 签名方法

签名方法请参见“API文档>接口签名”。

# 请求

请求行

```
POST /structuredlog
```

请求头

x-cls-hashkey 请求头表示日志根据 hashkey 路由写入到 CLS 对应范围的主题分区，以保证写入到的每个主题分区是严格有序的，便于消费时有序消费。

| 字段名 | 类型 | 位置 | 是否必选 | 含义 |
| --- | --- | --- | --- | --- |
| x-cls-hashkey | string | header | 否 | 根据 hashkey 写入相应范围的主题分区 |

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 |

| 字段名 | 类型 | 位置 | 是否必选 | 含义 |
| --- | --- | --- | --- | --- |
| 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，表示有多个 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 |

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“API文档>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 ，解压并安装。示例版本为 protobuf 2.6.1，环境为 CentOS 7.3 系统。

   2. 解压 protobuf-2.6.1.tar.gz 压缩包至 /usr/local 目录并进入该目录，执行命令如下：

      ```
      [root@VM_0_8_centos]# 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：

      ```
      [root@VM_0_8_centos protobuf-2.6.1]# ./configure
      [root@VM_0_8_centos protobuf-2.6.1]# make && make install
      [root@VM_0_8_centos protobuf-2.6.1]# 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：

```
[root@VM_0_8_centos protobuf-2.6.1]# protoc --version
liprotoc 2.6.1
```

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 protoc 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，如下所示：

```
[root@VM_0_8_centos protobuf-2.6.1]# protoc --cpp_out=./ ./cls.proto
[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。

# 数据导入
# 导入 COS 数据

## 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。
- 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。

## 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，单击日志主题，进入日志主题页面。
3. 选择需要投递的日志主题，进入日志主题管理页面。
4. 选择采集配置页签，在对象存储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流量。 | 列表选择 | 是 |
| 对象存储存储桶 | 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。 | 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/ 无压缩。 | 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。

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：
    - 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。
    - 提取模式：可选择以下几种提取模式。

| 提取模式 | 说明 |
| --- | --- |
| JSON | 结构化提取 |

| 提取模式 | 说明 |
| --- | --- |
| | 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"}<br><br>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<br>body_sent: 23<br>http_host: 127.0.0.1<br>method: POST<br>referer: http://127.0.0.1/my/course/4<br>remote_ip: 10.135.46.111<br>request: POST /event/dispatch HTTP/1.1<br>response_code: 200<br>responsetime: 0.232<br>time_local: 22/Jan/2019:19:19:34 +0800<br>upstreamhost: unix:/tmp/php-cgi.sock<br>upstreamtime: 0.232<br>url: /event/dispatch<br>xff: - | |
| 分隔符 | 结构化提取<br>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/<br><br>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 -<br>bytes: 35<br>host: 127.0.0.1<br>length: 647<br>referer: http://127.0.0.1/<br>request: GET /online/sample HTTP/1.1<br>status: 200<br>time: [Tue Jan 22 14:49:45 CST 2019 +0800] | |

| 提取模式 | 说明 |
| --- | --- |
| 单行-全文日志 | 非结构化提取<br>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。<br>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<br><br>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 | |

- 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。

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：
   - 索引状态：确认是否需要开启索引以使用日志检索等分析功能。
   - 全文索引：
     - 全文分词符：默认为 @&()='",;:<>[]{}/ \n\t\r ，确认是否需要修改。
     - 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：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。

> 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

# 检索分析
# 检索分析概述

## 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。

### 高级功能

- 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。
- 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。
- 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。

## 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。

# 规格与限制

检索分析的规格与限制，请参见 规格与限制。

# 语法规则

## 语法规则（Lucene）

| 保留关键字 | 说明 |
| --- | --- |
| SOURCE | 查询某个 SOURCE 源的日志，支持通配符，例如 __SOURCE__:"127.0.0.*" |
| FILENAME | 查询来源为某个文件的日志，支持通配符 ，例如 __FILENAME__:"/var/log/access.*" |
| 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 |
| TO | 范围 逻辑操作符，例如 request_time:[0.1 TO 1.0] |
| "" | 双引号，引用一个短语词组（短语当作一个整体词组），例如 name:"john Smith" |
| ： | 冒号，表示作用于的 key 字段，即键值检索，例如 level:ERROR |
| * | 通配符查询，匹配零个、单个、多个字符，例如 host:www.test*.com |
| ? | 通配符查询，匹配单个字符，例如 host:www.te?t.com |
| () | 分组操作符，控制逻辑运算优先级，例如（ERROR OR WARNING) AND pid:1234 |
| > | 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 |
| [] | 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 |
| + | 逻辑操作符，类似 AND， +A 表示 A 一定存在，例如 +level:ERROR +pid:1234 |
| - | 逻辑操作符，类似 NOT， -A 表示 A 不存在，例如 +level:ERROR -pid:1234 |
| \|\| | 或逻辑，类似 OR，例如 level:ERROR \|\| level:WARNING |

| 保留关键字 | 说明 |
| --- | --- |
| && | 与逻辑，类似 AND，例如 level:ERROR && pid:1234 |
| ! | 非逻辑，类似 NOT，例如 level:ERROR !pid:1234 |
| / | 正则表达式标识符，/${regExp}/，例如 /[mb]oat/ 表示搜索包含 moat 或 boat 的结果 |
| *exists* | _exists_:key，返回 key 不为空的值，例如 _exists_:userAgent 表示搜索 userAgent 字段有值的结果 |
| ~ | 相似搜索，例如 level:errro~ ，可以命中 level 为 error 关键字的结果 |

说明：

- 语法操作符区分大小写，例如 AND、OR 表示检索逻辑运算符，而 and、or 视为普通词组。
- 多个检索语句用空格连接时，视为"或"逻辑，例如 warning error 表示包含 warning 或 error 关键字的结果。
- 语法中的字符均为保留字符，若检索关键字中包含这些语法保留字符，均需要转义。
- 使用键值检索时（形如 key：value），键名（key）必须在日志主题的键值索引配置项里面。

# 检索语句示例

| 检索场景 | 检索语句 |
| --- | --- |
| 检索包含 ERROR 关键字的日志 | 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.") 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} |

| 检索场景 | 检索语句 |
| --- | --- |
| 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，即 message 字段有值的日志 | message:* 或者 _exists_:message |
| 检索不包含 message 字段的日志 | NOT _exists_:message |
| 检索 message 字段为空的日志 | message:"" |

# 索引配置

## 操作场景

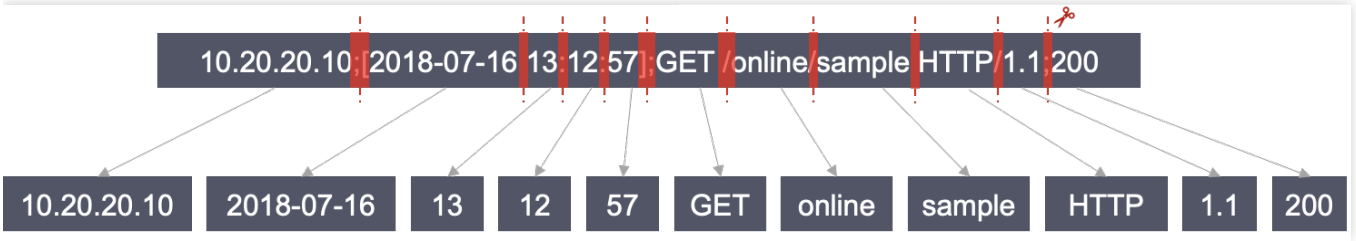
索引配置是使用日志服务（Cloud Log Service，CLS）进行检索分析的必要条件，只有开启索引才能对日志进行检索分析。而且不同的索引规则也会产生不同的日志检索分析效果，本文档会详细介绍索引的配置规则及原理。

## 功能介绍

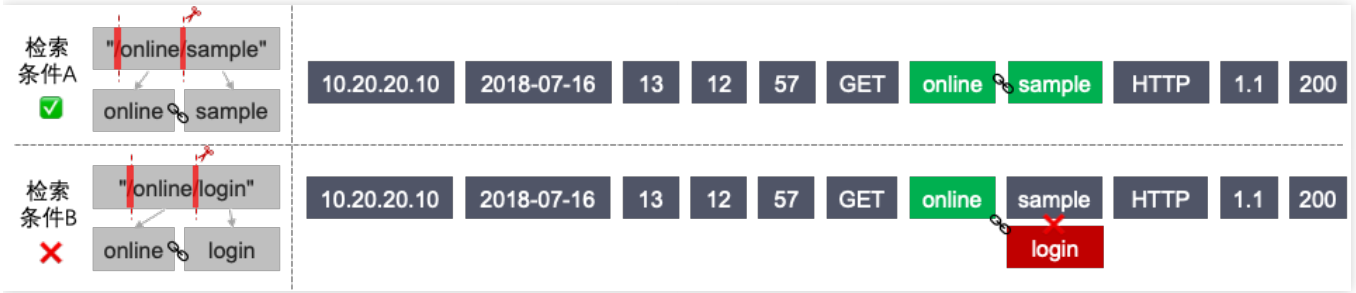
索引配置的核心是对原始日志进行“分词”，以快速且便捷地根据特定的检索条件检索日志。同时在索引配置中还可以针对特定的字段“开启统计”，便于使用 SQL 对日志进行统计分析。
检索一个较长的日志时，通常只使用其中一部分内容进行检索。例如：需要检索出包含 sample 的如下完整日志：

```
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。<br>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。
> - 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， 开启全文索引后，将会在字段前加上@符号。

# 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。 |
| 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

整体配置

| 配置项 | 功能描述 |
| --- | --- |

| 配置项 | 功能描述 |
| --- | --- |
| 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。 |

## 字段配置

| 配置项 | 功能描述 |
| --- | --- |
| 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。<br>- 仅支持字母、数字及 _-./@ ，且不能以 _ 开头( __CONTENT__ 字段例外)。<br>- 不允许同时包含 JSON 父子级字段，例如 a 及 a.b 。 |
| 字段类型 | 字段的数据类型，包括 text、long、double 三种类型。<br>- text 类型支持使用通配符进行模糊检索，不支持范围比较。<br>- 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。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，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，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。<br>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 |

| 内置保留字段 | 说明 | 键值索引配置 |
| --- | --- | --- |
| __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 |

# 高级设置

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" 。<br>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-city 。<br>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-city 。低频日志主题不支持键值检索，此时将无法检索这些字段。<br>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），详见 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名称，进入日志主题管理页。
4. 在索引配置页签中**，单击编辑**，进入编辑索引配置页面。
    - 索引状态：不开启时无法检索日志。
    - 全文索引 ：配置项说明详见全文索引。

- 键值索引：已开启全文索引时，键值索引不会产生额外成本。
  - 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。
  - 字段列表：配置项说明详见键值索引。

5. 单击确定，保存配置信息。

### 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名称，进入日志主题管理页。
4. 在索引配置页签中**，单击导入配置规则**。
5. 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，保存当前日志主题的索引配置。

## 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？

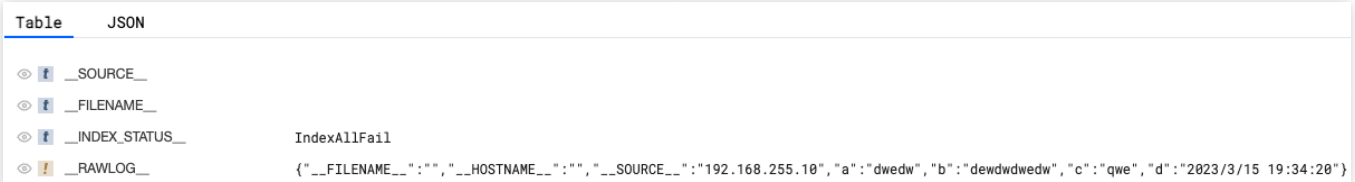
日志创建索引过程中，如果原始日志格式异常或索引配置与原始日志不匹配，可能会导致日志创建索引异常，此时 CLS 为了确保日志不丢失，会将原始日志存储在 __RAWLOG__ 中，作为兜底的异常处理方式。 __INDEX_STATUS__ 则为创建索引异常的原因。

- __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。

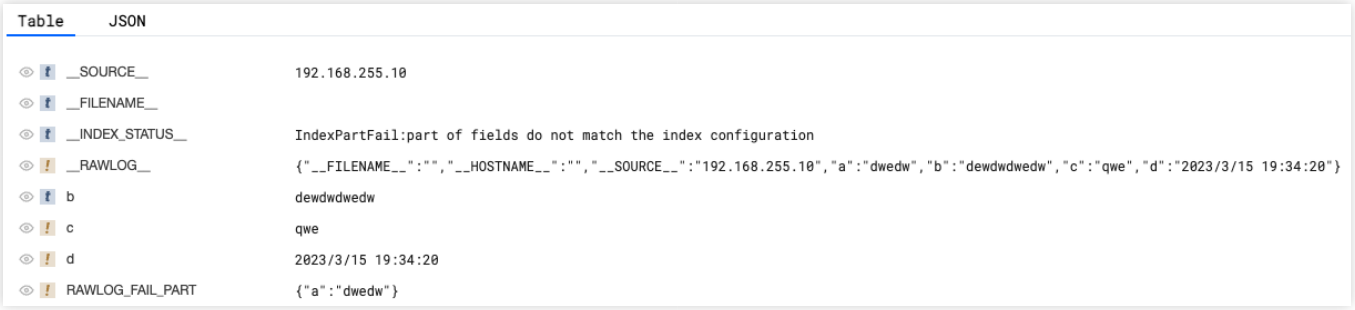
创建索引异常包含两类情况：

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。

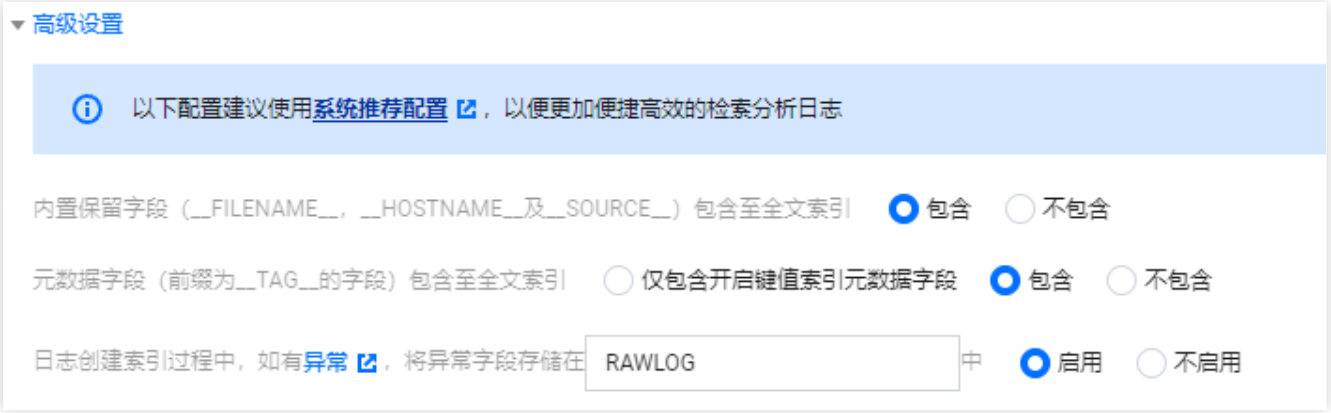



2. 日志中部分字段未成功创建索引，部分字段成功创建索引，此时日志中既具备 __RAWLOG__ 字段，又具备创建索引

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（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

## 概述

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（不产生写流量及索引存储）。
  数据量较大时会消耗较长时间，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。
- 仅支持通过控制台重建写流量500GB以内的时间范围的日志。超过该限制时建议您缩小需要重建索引的日志时间范围。
- 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。

# SQL语法
# SELECT 语法

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，也可以再在 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

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

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）。<br>例如 status 列的序号为1，所以下面两个语句为等同关系。<br>- * \| SELECT status, count(*) AS PV GROUP BY status<br>- * \| 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：

  ```
  * |
  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分钟的时间粒度统计 PV 及 UV：
  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
```

# ORDER BY 语法

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

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 |

# WHERE 语法

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 后缀的为 .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'
```

# HAVING 语法

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。

# 嵌套子查询

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。

# JOIN 语法

JOIN 语法用于连接多个表。日志服务支持同日志主题内多个查询结果进行连接，或连接对象存储中的 CSV 文件、MySQL 数据库。

## 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 |
|---|---|---|---|---|
| 168.132.25.66 | "GET /class/111.html HTTP/1.1" | 3 | Michael Johnson | Male |
| 163.168.25.19 | "GET /NoBUG/666 HTTP/1.1" | 17 | Logan Thompson | Male |
| 125.168.66.37 | "GET /class/114.html HTTP/1.1" | 4 | Sarah Davis | Female |
| 86.168.19.168 | "GET /NoBUG/666 HTTP/1.1" | 4 | Sarah Davis | Female |
| 19.10.131.132 | "GET /NoBUG/666 HTTP/1.1" | 6 | Emily Anderson | Female |
| 125.19.187.66 | "GET /NoBUG/666 HTTP/1.1" | 23 | Benjamin Rodriguez | Male |
| 132.123.10.53 | "GET /class/113.html HTTP/1.1" | 17 | Logan Thompson | Male |
| 168.10.25.132 | "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于 JOIN 语法一起使用的方式。

```
* |
select
  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 | 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" |

# UNION 语法

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

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

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

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 |

# SQL函数
# 函数概览

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(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。 |

| 函数名称 | 说明 |
| --- | --- |
| 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(number,n) | 对比当前时间周期内的计算结果 number 与 n 秒之前时间周期内的计算结果，返回数组，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]。 |

# 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。<br>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。<br>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。 |

| 函数名称 | 说明 |
| --- | --- |
| 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。 |
| 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。 |

| 函数名称 | 说明 |
| --- | --- |
| 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）。 |
| current_timezone() | 返回当前时区。 |
| 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。 |

| 函数名称 | 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。 |
| 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。 |
| now() | 返回当前时间时间戳（非Unix格式）与 current_timestamp 函数同等用法。 |
| 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_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。 |
| 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。 |

# 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。 |

| 函数名称 | 说明 |
| --- | --- |
| 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。 |

| 函数名称 | 说明 |
| --- | --- |
| 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, lambda_expression) | 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。 |
| 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。 |

| 函数名称 | 说明 |
| --- | --- |
| 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。 |
| 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。 |

| 函数名称 | 说明 |
| --- | --- |
| 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。

# 通用聚合函数

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 |

| 函数语句 | 说明 | 示例 |
| --- | --- | --- |
| 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

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 |

| a | NOT a |
| --- | --- |
| NULL | NULL |

# 算数及比较运算符

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 |

| 运算符 | 描述 | 示例 |
| --- | --- | --- |
| >= | 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 。<br>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。 | NULL IS NOT DISTINCT FROM NULL |
| LEAST | 语法： LEAST(x, y...)<br>返回 x,y...中的最小值。 | LEAST(1,2,3) |
| GREATEST | 语法： GREATEST(x, y...)<br>返回 x,y...中的最大值。 | GREATEST(1,2,3) |
| ALL | 语法： x expression operator ALL ( subquery )<br>x 满足所有条件时，返回 true，operator 支持 <、>、<=、>=、=、<>、!= 。 | 示例1：21 < ALL (VALUES 19, 20, 21)<br>示例2：* \| SELECT 200 = ALL(SELECT status) |
| ANY / SOME | 语法： x expression operator ANY ( subquery ) 或 x expression operator SOME ( subquery )<br>x 满足任意一个条件时，返回 true，operator 支持 <、>、<=、>=、=、<>、!= 。 | 示例1：'hello' = ANY (VALUES 'hello', 'world')<br>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 |

# 估算函数

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 函数 | 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

### 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)
```

### 示例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

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
   )
)
```

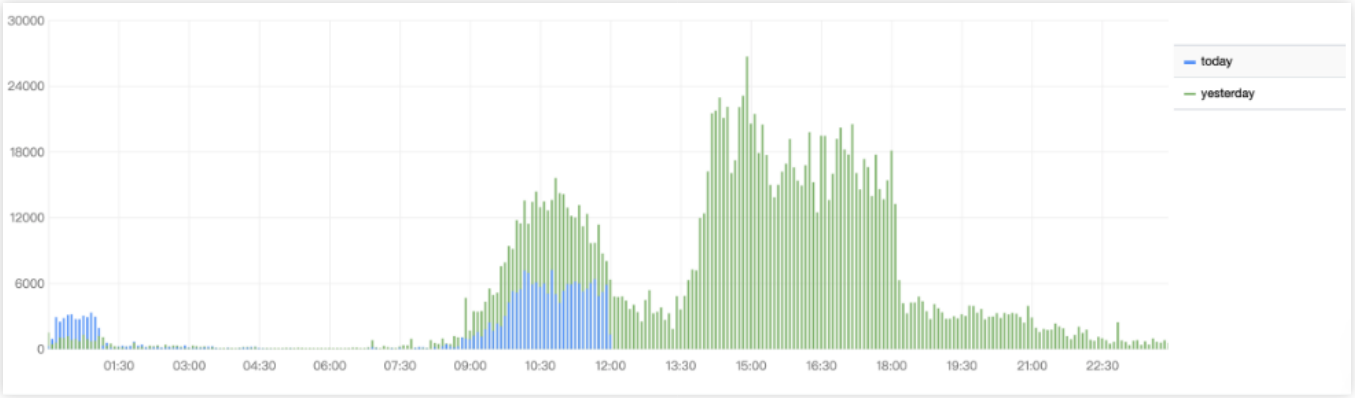
示例2：计算今天每5分钟网站访问量变化趋势与昨天同时间段的对比

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

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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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 | 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。<br>* \| select concat_ws('#',split('cloud.city.com/product/cls', '/')) |

| 函数名称 | 说明 | 示例 |
| --- | --- | --- |
| | 型。<br>注意：此函数中 array(varchar) 参数为一个数组，不是字符串。 | |
| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| select position('G' IN http_method) |

| 函数名称 | 说明 | 示例 |
| --- | --- | --- |
| 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’。<br>* \| 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’。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| SELECT strpos(host, 'org') |

| 函数名称 | 说明 | 示例 |
| --- | --- | --- |
| 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。<br>* \| 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。<br>* \| 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。<br>* \| SELECT substr(remote_user, 2, 5) |
| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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。<br>* \| 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
```

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 DES
  C 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

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。<br>- 返回值格式：YYYY-MM-DD，例如：2021-05-21。<br>- 返回值类型：DATE 。 | * \| select current_date |
| current_time | 返回当前时间。<br>- 返回值格式：HH:MM:SS.Ms Time zone，例如：17:07:52.143+08:00。<br>- 返回值类型：TIME。 | * \| select current_time |
| current_timestamp | 返回当前时间戳。<br>- 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]。<br>- 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）。<br>- 返回值类型：VARCHAR，例如： Asia/Shanghai。 | * \| select current_timezone() |
| localtime | 返回本地时间。<br>- 返回值格式：HH:MM:SS.Ms，例如：19:56:36。<br>- 返回值类型：TIME。 | * \| select localtime |
| localtimestamp | 返回本地的日期和时间。<br>- 返回值格式：YYYY-MM-DD HH:MM:SS.Ms，例如：2021-07-15 19:56:26.908。<br>- 返回值类型：TIMESTAMP。 | * \| select localtimestamp |

| 函数名称 | 说明 | 示例 |
| --- | --- | --- |
| now() | 返回当前日期和时间，与current_timestamp 函数同等用法。<br>- 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]。<br>- 返回值类型：TIMESTAMP。 | * \| select now() |
| last_day_of_month(x) | 返回月份最后一天。<br>- 返回值格式：YYYY-MM-DD，例如：2021-05-31。<br>- 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。<br>- 返回值格式：YYYY-MM-DD，例如：2021-05-31。<br>- 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。<br>- 返回值格式：HH:MM:SS.Ms Time zone，例如：2021-07-15T17:10:56.735+08:00[Asia/Shanghai]。<br>- 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。<br>- 返回值格式：YYYY-MM-DD HH:MM:SS.Ms，例如：2017-05-17 01:41:15.000。<br>- 返回值类型：TIMESTAMP 。<br>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：<br>* \| select from_unixtime(1494985275)<br>示例2：<br>* \| 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。<br>- 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]。<br>- 返回值类型：TIMESTAMP。 | 示例1：<br>* \| select from_unixtime(1494985275, 'Asia/Shanghai')<br>示例2：<br>* \| 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。<br>返回值类型：LONG。例如：1626347592。 | * \| select to_unixtime(cast(__TIMESTAMP__ as timestamp)) |

| 函数名称 | 说明 | 示例 |
| --- | --- | --- |
| to_milliseconds(interval) | 以毫秒为单位返回间隔的时间值。<br>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。<br>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。<br><br>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。<br>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
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
```

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。<br>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。<br><br>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。<br>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：<br>- 0：将缺失的值补全为0。<br>- null：将缺失的值补全为 null。<br>- last：将缺失的值补全为上一个时间点的值。<br>- next：将缺失的值补全为下一个时间点的值。<br>- avg：将缺失的值补全为前后两个时间点的平均值。 |

示例

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
```

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 gro
up 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)) 。

| 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')<br>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')<br>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 |

| unit | 说明 |
| --- | --- |
| 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。<br>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。<br>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 |
| --- | --- |
| %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(000000999000)，~~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。 |

| Format | 说明 |
| --- | --- |
| %r | 时间。12小时制，格式为 hh:mm:ss AM/PM 。 |
| %S | 秒。取值范围为00 - 59。<br>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 fomat 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

IP 函数可用于判断 IP 地址属于内网还是外网，也可用于分析 IP 地址所属的国家、省份、城市。本文介绍 IP 函数的基本语法及示例。

## IP 地址函数

> 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；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。
> - IP 目前仅支持 IPv4 地址，IPv6 暂不支持。
> - 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_domain(KEY) | 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。 | * \| SELECT ip_to_domain(ip) |
| 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 | 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 | 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 | * \| SELECT ip_to_city_code(ip) |

| 函数名称 | 说明 | 示例 |
| --- | --- | --- |
| | 及国外城市。 | |
| ip_to_city_geo | 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。 | * \| 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) |

# 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。

| 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[[192.168.1.0, 192.168.1.255]]。 | * \| SELECT ip_subnet_range(concat(ip,'/24')) |
| is_subnet_of | 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) |

| 函数名 | 说明 | 示例 |
| --- | --- | --- |
| is_prefix_subnet_of | 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')) |

# 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，可参考如下查询和分析语句。

  ```
  * | SELECT ip_to_province(ip) AS province, count(*) as PV where ip_to_domain(ip)!='intranet' GRO UP 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
  ```

# URL 函数

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.abc.com/#/project/dashboard-demo/categoryList') | /project/dashboard-demo/categoryList<br> |
| url_extract_host(url) | 提取出 URL 中的 host，结果为 varchar 类型。 | * \| select url_extract_host('https://console.abc.com/cls') | console.abc.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.abc.com/cls?region=ap-city1','region') | ap-city1 |
| url_extract_path(url) | 提取出 URL 中的 path，结果为 varchar 类型。 | * \| select url_extract_path('https://console.abc.com/cls?region=ap-city1') | /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.abc.com:80/cls?region=ap-city1') | 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.abc.com:80/cls?region=ap-city1') | https |
| url_extract_query(url) | 提取出 URL 中的 query，结果为 varchar 类型。 | * \| select url_extract_query('https://console.abc.com:80/cls?region=ap-city1') | region=ap-city1 |
| url_encode(value) | 对 value 进行转义编码，使之能应用在 URL_query 中。<br>- 字母不会被解码。<br>- .-*_不会被编码。<br>- 空格被解码为+。<br>- 其他字符被解码为 UTF8 格式。 | * \| select url_encode('https://console.abc.com:80/cls?region=ap-city1') | https%3A%2F%2Fconsole.abc.com%3A80%2Fcls%3Fregion%3Dap-city1 |
| url_decode(value) | 对 URL 进行解码。 | * \| select url_decode('https%3A%2F%2Fconsole.abc.com%3A80%2Fcls%3Fregion%3Dap-city1') | https://console.abc.com:80/cls?region=ap-city1 |

# JSON 函数

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。<br>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。 |

## 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。<br>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。 |

### 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 | JSON 路径，格式为 $.store.book[0].title。 |

### 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

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 函数 | 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 函数 | 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 函数 | 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。 |

| 参数 | 说明 |
| --- | --- |
| 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。 |

| 参数 | 说明 |
| --- | --- |
| 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

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 函数 | 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_max(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) | 把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 函数 | 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。 |
| 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 |
|---|---|

| 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, '指定区域','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,'指定区域','sh'],'/','region')

- 查询和分析结果

  region/指定区域/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 |
| --- | --- |

| 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。 |

| 参数 | 说明 |
| --- | --- |
| 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]
```

# 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。<br>当参数值为日期和时间表达式时，step 格式如下：<br>- interval 'n' year to month，表示间隔为 n 年。<br>- 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

  [1628733298,1628733358,1628733418,1628733478,1628733538,1628733598,1628733658,1628733718,1628733778,1628733838,1628733898,1628733958,1628734018,1628734078]

# 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。<br>- 如果 start 为负数，则从末尾开始。<br>- 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。 |

| 参数 | 说明 |
| --- | --- |
| 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

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。<br>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。<br>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。<br>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。<br>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

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。 |

### 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 |
| --- | --- |

| 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

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。<br>例如 * \| SELECT cosine_similarity(MAP(ARRAY['x','y'], ARRAY[1.0,0.0]), MAP(ARRAY['x','y'], ARRAY[0.0,1.0]))<br>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。 |

| 函数名称 | 说明 |
| --- | --- |
| 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。<br>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。<br>例如 * \| select timeCost,width_bucket(timeCost,array[10,100,1000]) |

| 函数名称 | 说明 |
| --- | --- |
| 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

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

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

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

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：<br>x -> x + 1<br>(x, y) -> x + y<br>x -> regexp_like(x, 'a+')<br>x -> x[1] / x[2]<br>x -> if(x > 0, x, -x)<br>x -> coalesce(x, 0)<br>x -> cast(x AS JSON)<br>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

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。<br>使用示例：<br>rows between current row and 1 following：当前行及后一行。<br>rows between 1 preceding and current row：当前行及前一行。<br>rows between 1 preceding and 1 following：前一行至后一行（共三行）。<br>rows between current row and unbounded following：当前行及后续所有行。<br>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。<br>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 |

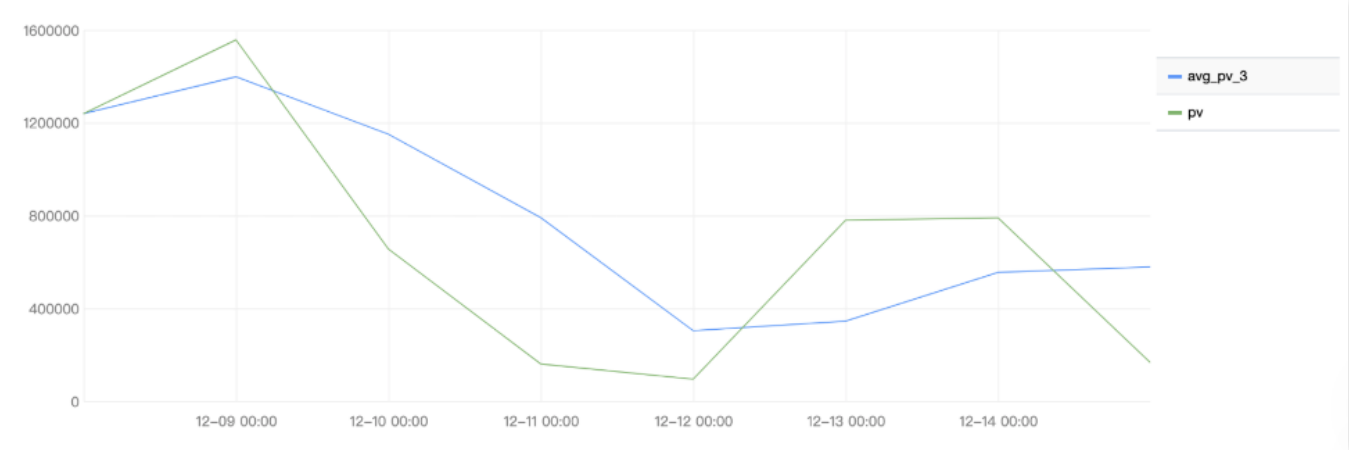
示例2：查询应用吞吐量的3日移动平均值变化趋势

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

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 |
| --- | --- | --- |

| 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(Geometry, Geometry) | boolean | 当两个实体有共同内部点时，返回 true。 |
| 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。 |

| 函数 | 返回值类型 | 说明 |
| --- | --- | --- |
| 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。 |

| 函数 | 返回值类型 | 说明 |
| --- | --- | --- |
| 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。 |
| 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。 |

# 快速分析

## 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。

# 多主题检索

## 操作场景

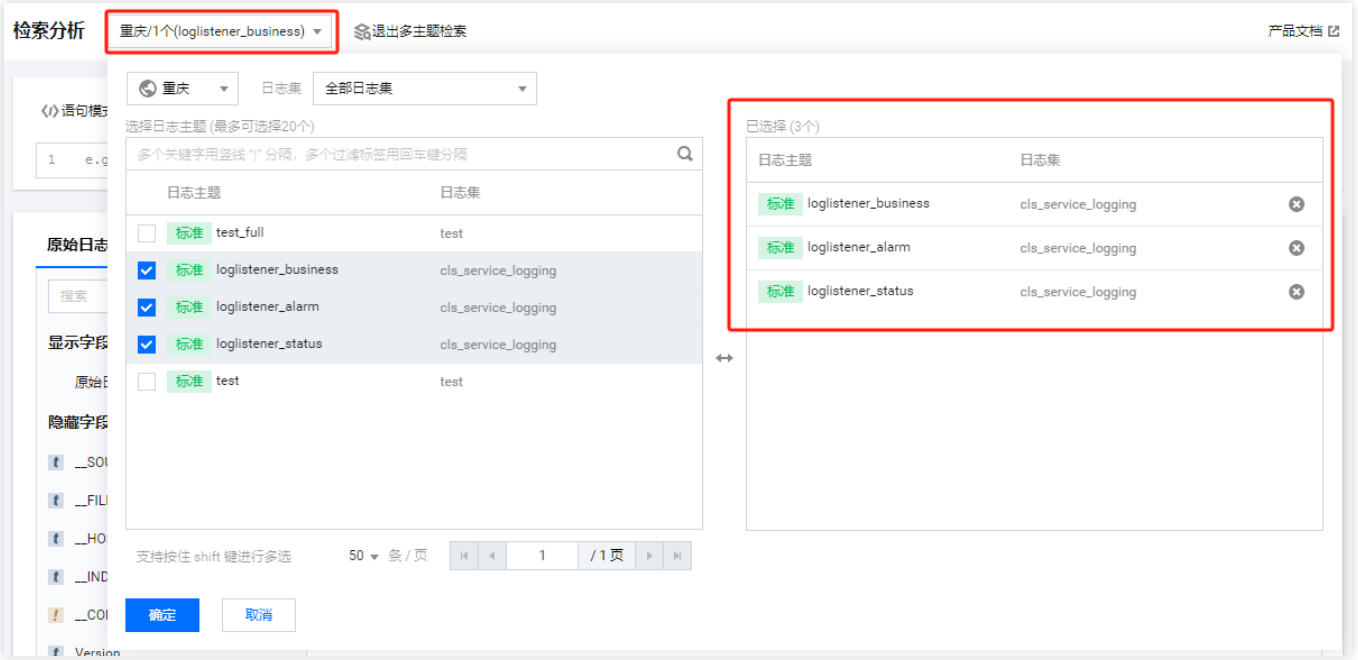
多主题检索用于集中检索分布在多个日志主题中的日志，例如通过 RequestID 跨日志主题检索相关日志。本文为您介绍如何使用多主题检索。

## 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20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. 单击顶部的多主题检索，进入多主题检索模式。
4. 选择需要同时检索的多个日志主题，单击确定。




5. 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。

# 自定义跳转

## 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。
自定义跳转支持如下操作类型：

| 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 |
| 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. 输入相应配置信息，单击保存。
    - 跳转类型：根据需要选择对应的跳转类型。
    - 配置名称：自定义跳转的名称。
    - 新窗口打开：是否跳转至新窗口。
    - 其他配置项：与跳转类型有关。
        - 打开外部 URL
          URL：需要打开的 URL 地址，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。
        - 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。
          检索分析语句：输入检索分析语句，可使用变量动态获取日志字段值等信息。

### 编辑/删除自定义跳转

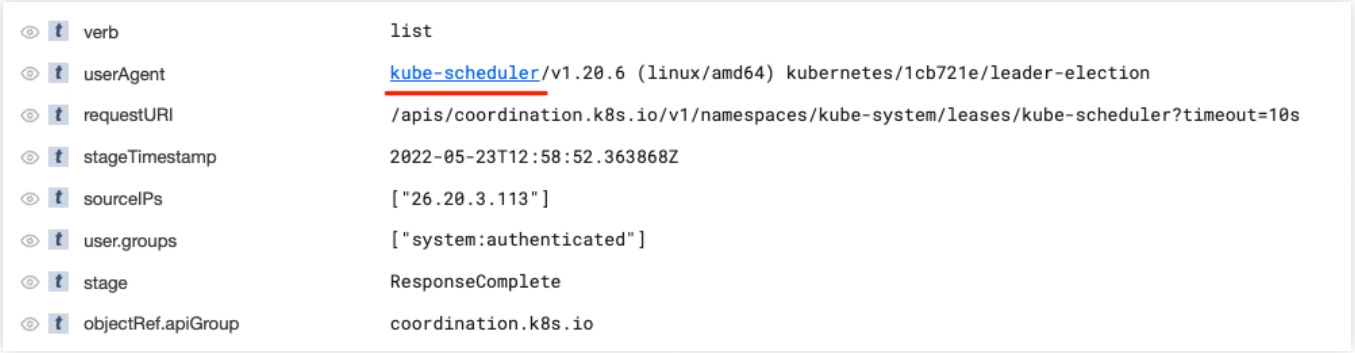
1. 登录日志服务控制台。
2. 在左侧导航栏中，单击检索分析，进入检索分析页面。

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. 在弹框中单击自定义跳转配置名称，单击名称右侧 可删除该配置，在右侧表单中可更新该配置。最后单击应用保存。

## 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。




- 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 。

# 上下文检索

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。

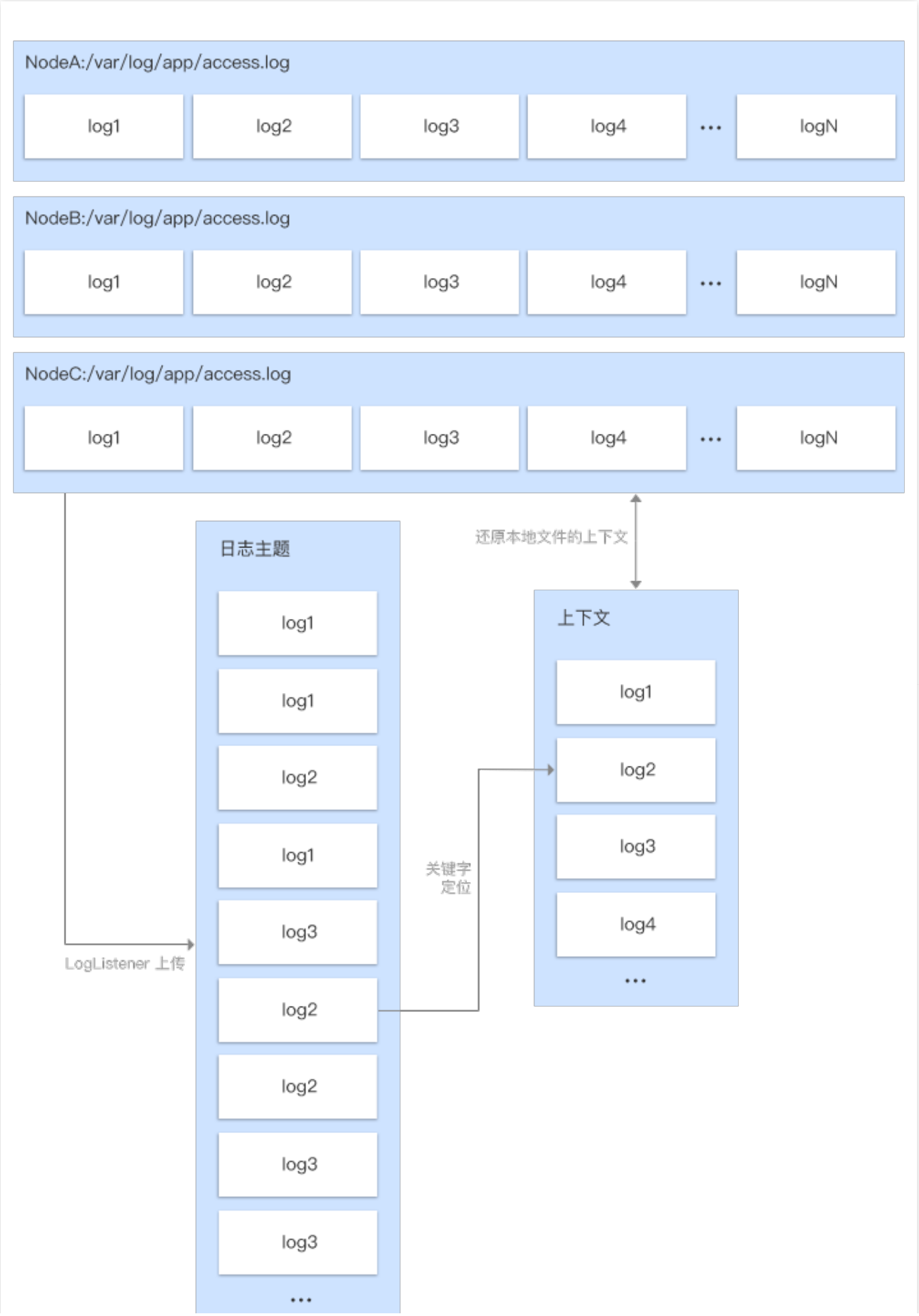
## 前提条件

- 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。

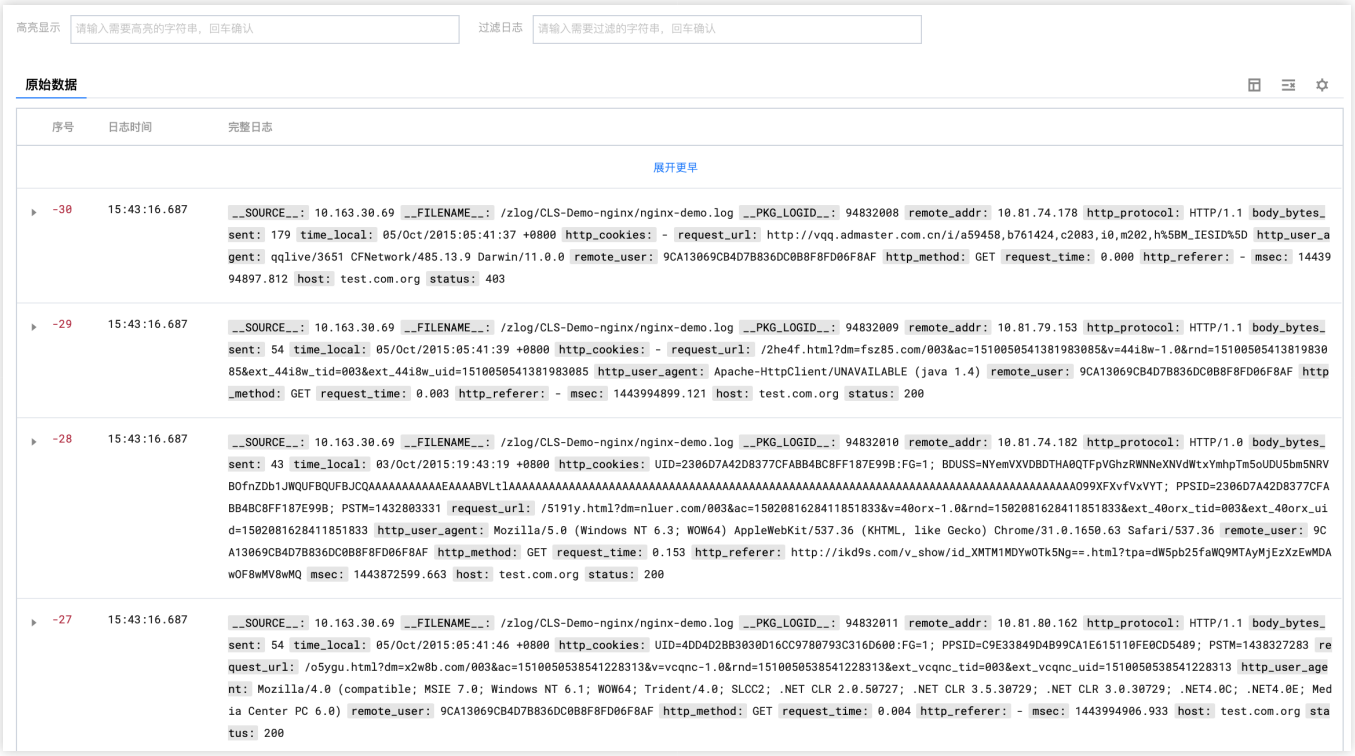

NodeA:/var/log/app/access.log
log1
log2
log3
log4
logN
NodeB:/var/log/app/access.log
log1
log2
log3
log4
logN
NodeC:/var/log/app/access.log
log1
log2
log3
log4
logN
还原本地文件的上下文
日志主题
log1
log1
log2
log1
log3
log2
log2
log3
log3
LogListener 上传
关键字
定位
上下文
log1
log2
log3
log4

# 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、日志集和日志主题。
4. 输入检索分析语句，选择时间范围， 单击检索分析。



5. 在原始日志页签下，找到目标日志的日志时间，单击 ，进入上下文检索分析页面。
6. 在上下文检索分析页面，系统仅检索出目标日志的前10条日志数据（上文）和后10条日志数据（下文）。


高亮显示
请输入需要高亮的字符串，回车确认
过滤日志
请输入需要过滤的字符串，回车确认
原始数据
序号
日志时间
完整日志
展开更早
-30
15:43:16.687
__SOURCE__: 10.163.30.69 __FILENAME__: /zlog/CLS-Demo-nginx/nginx-demo.log __PKG_LOGID__: 94832008 remote_addr: 10.81.74.178 http_protocol: HTTP/1.1 body_bytes_sent: 179 time_local: 05/Oct/2015:05:41:37 +0800 http_cookies: - request_url: http://vqq.admaster.com.cn/i/a59458,b761424,c2083,i0,m202,h%5BM_IESID%5D http_user_agent: qqlive/3651 CFNetwork/485.13.9 Darwin/11.0.0 remote_user: 9CA13069CB4D7B836DC0B8F8FD06F8AF http_method: GET request_time: 0.000 http_referer: - msec: 1443994897.812 host: test.com.org status: 403
-29
15:43:16.687
__SOURCE__: 10.163.30.69 __FILENAME__: /zlog/CLS-Demo-nginx/nginx-demo.log __PKG_LOGID__: 94832009 remote_addr: 10.81.79.153 http_protocol: HTTP/1.1 body_bytes_sent: 54 time_local: 05/Oct/2015:05:41:39 +0800 http_cookies: - request_url: /2he4f.html?dm=fsz85.com/003&ac=151005054138198308 5&v=44i8w-1.0&rnd=151005054138198308 5&ext_44i8w_tid=003&ext_44i8w_uid=1510050541381983085 http_user_agent: Apache-HttpClient/UNAVAILABLE (java 1.4) remote_user: 9CA13069CB4D7B836DC0B8F8FD06F8AF http_method: GET request_time: 0.003 http_referer: - msec: 1443994899.121 host: test.com.org status: 200
-28
15:43:16.687
__SOURCE__: 10.163.30.69 __FILENAME__: /zlog/CLS-Demo-nginx/nginx-demo.log __PKG_LOGID__: 94832010 remote_addr: 10.81.74.182 http_protocol: HTTP/1.0 body_bytes_sent: 43 time_local: 03/Oct/2015:19:43:19 +0800 http_cookies: UID=2306D7A42D8377CFABB4BC8FF187E99B:FG=1; BDUSS=NYemVXVDBDTHA0QTFpVGhzRWNNeXNVdWtxYmhpTm5oUDU5bm5NRVBOfnZDb1JWQUFBQUFBJCQAAAAAAAAAAAEAAAABVLtlAAAAAAAAAAA

# 仪表盘
# 仪表盘概述

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。

## 功能特性

### 仪表盘功能

仪表盘功能如下表所示：

| 功能名称 | 功能描述 |
| --- | --- |
| 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。<br>- 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。 |
| 图表自定义时间 | 配置自定义时间后，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。 |
| 预置仪表盘 | 针对云产品日志，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。 |
| 订阅仪表盘 | 支持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。 |
| 导出仪表盘 | 支持导出仪表盘的 Json 文件，可用于跨账号的仪表盘文件导入。 |

### 图表功能

图表类型如下表所示：

| 图表类型 | 使用场景 |
| --- | --- |
| 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。 |
| 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。 |
| 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。 |
| 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。 |

| 图表类型 | 使用场景 |
| --- | --- |
| 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。 |
| 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。 |
| 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。 |
| 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。 |
| 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。 |
| 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。 |
| 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。 |

### 其他

其他功能如下表所示：

| 功能名称 | 功能描述 |
| --- | --- |
| 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。 |
| 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。 |
| 添加图表到仪表盘 | 支持将图表分析结果保存到仪表盘，进行长期监控。 |
| 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。 |

## 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

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以其相关操作。

## 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，进入仪表盘列表页面。
3. 在自定义仪表盘页面，单击创建仪表盘。
4. 选择创建仪表盘的方式，单击下一步。
    - 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。
    - 从模板库创建（推荐）：CLS 提供了丰富的仪表盘模版，可以根据自己的场景与日志内容选择合适的模版一键启用，自动生成仪表盘。
5. 输入仪表盘相关配置信息，单击保存，创建成功。

## 配置项说明

| 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。 |

# 添加图表

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。

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. 填写检索分析语句，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 | 图表名称。 |
| 目标仪表盘 | 图表将要添加到的仪表盘类别。<br>- 选择已有仪表盘，会添加仪表盘到已有仪表盘中。<br>- 选择新建仪表盘，则需要新建一个仪表盘，并将图表添加进去。 |
| 仪表盘 | 仪表盘名称。 |

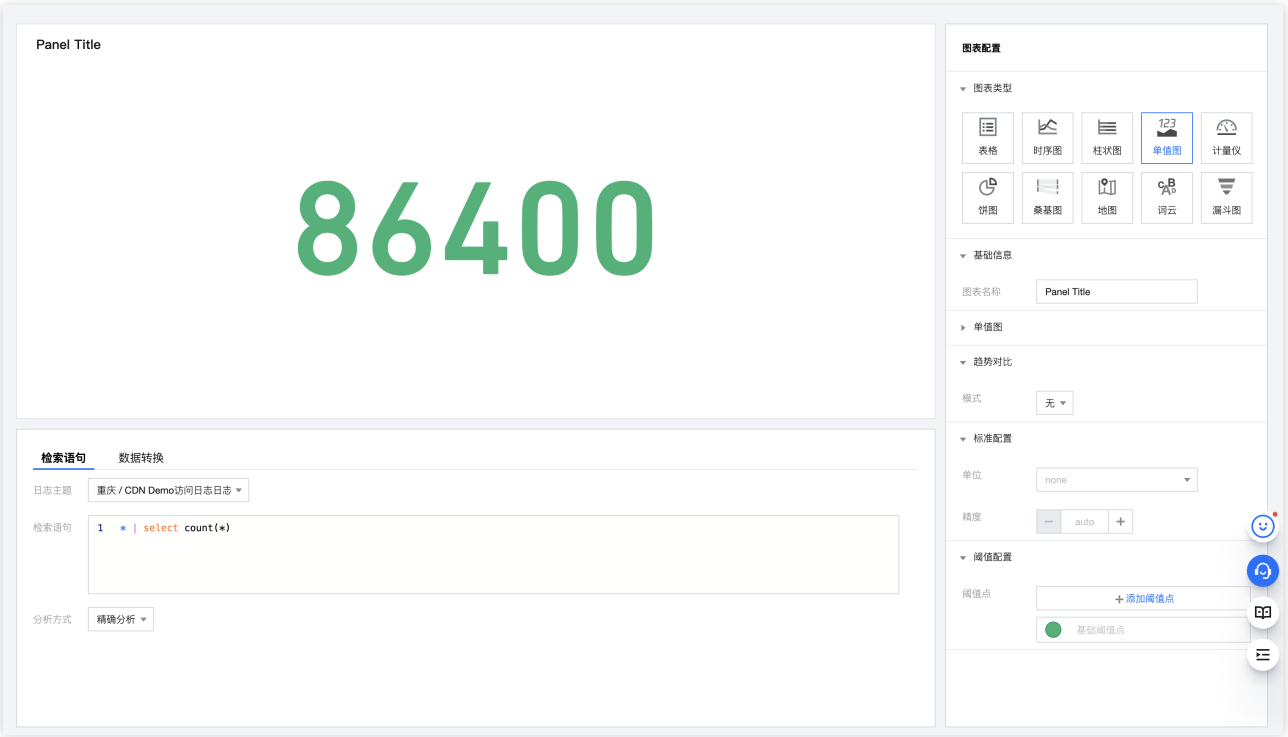
### 通过仪表盘添加图表

1. 登录日志服务控制台。
2. 在左侧导航栏中，单击仪表盘，进入仪表盘管理页面。
3. 单击目标仪表盘ID，进入编辑页面。

> 说明：
>
> 如果您没有仪表盘，请新建仪表盘。

4. 单击添加图表，图表编辑页面。
    - 自定义图表：通过自定义检索语句创建图表，则支持全部统计分析功能。

填写检索分析语句即可自动生成图表。



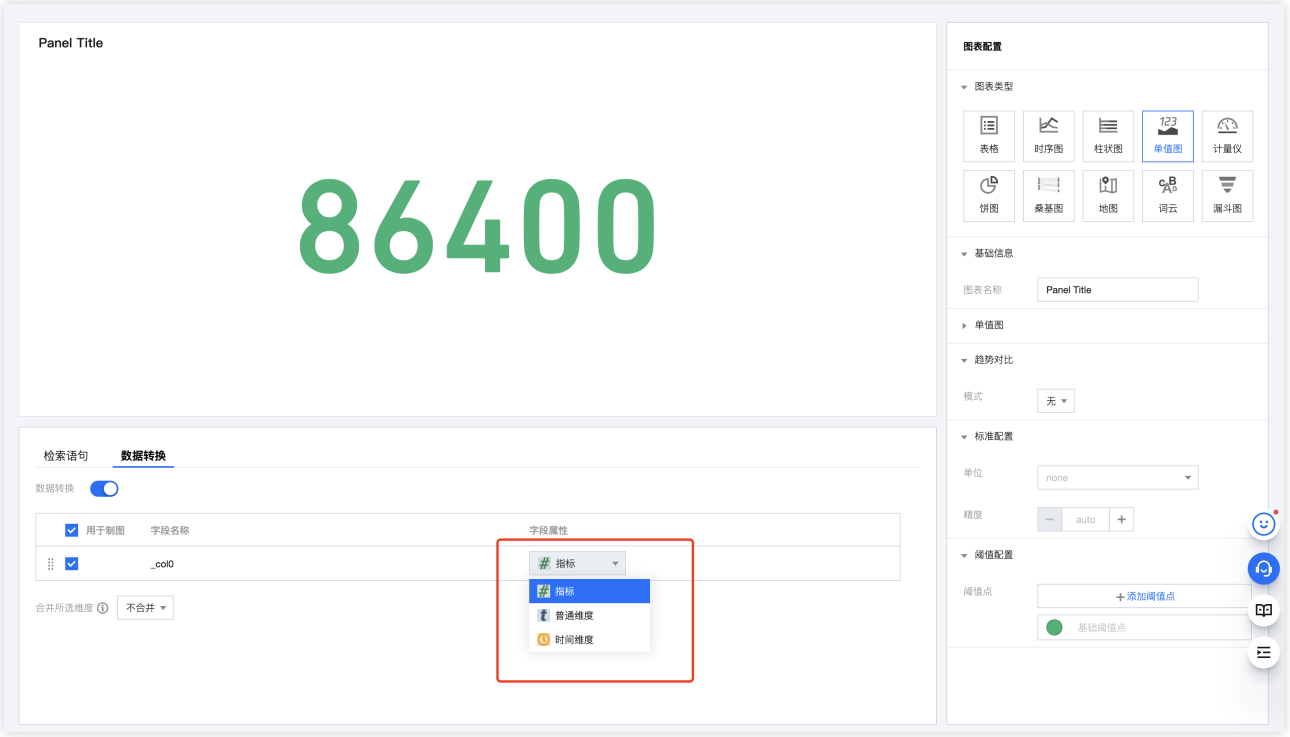

- 简易图表：通过可视化拖拽方式轻松创建图表，但统计分析功能较弱。




| 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。 |

| 图表元素 | 说明 |
| --- | --- |
| 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。但必要时用户可手动调整数据结构配置。




6. 单击保存。

# 配置图表
# 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。

## 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。 |
| 表格 | - 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。<br>- 最小列宽：配置表格的最小列宽。<br>- 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。<br>- 单元格样式：单元格显示样式，支持阈值高亮，需要配合阈值使用。<br>- 列过滤：开启后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。 |
| 自定义跳转 | 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，配合单元格样式配置，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，在表格 > 单元格样式配置中，选择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。 |

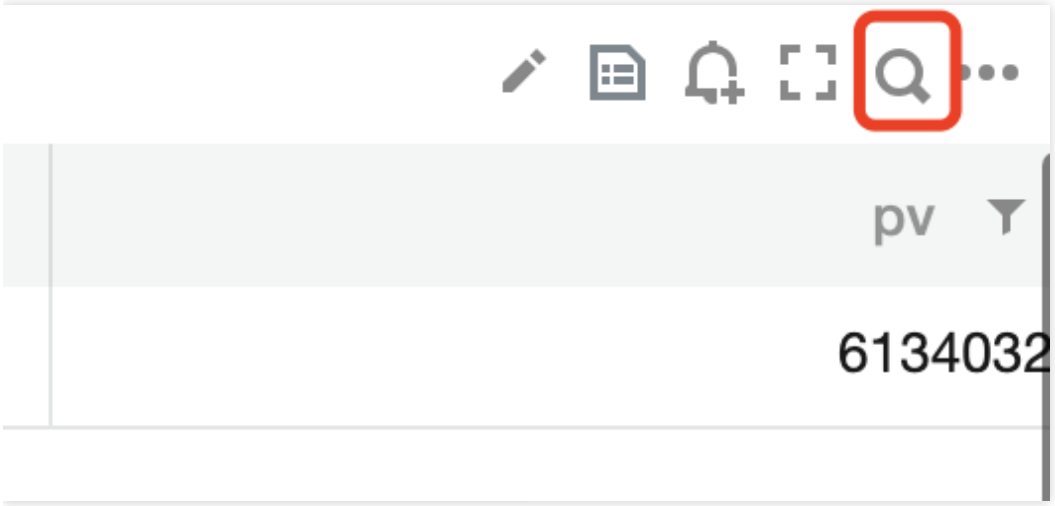
示例：当表格有多个指标时，如想把 Error 日志数、错误率同时显示在同一行，并带上单位。Error 日志数单位为 “次数”，错误率单位为 “%”，单位不同。此时选择添加字段配置，选择“Error 日志数”，单位配置为“次数”，选择“错误率”，单位配置为“%”，实现不同的字段配置不同的效果。




## 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 |

# 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。

## 图表配置

### 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。 |
| 提示框 | 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。 |
| 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。 |
| 自定义跳转 | 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。 |

### 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 | - 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。<br>- 线性插值：点与点之间的连线是否平滑处理。<br>- 线宽：控制线条的粗细。<br>- 填充：控制填充区域的透明度，为0时不填充。<br>- 显示点：显示数据点，无数据则不显示。<br>- 空值：控制时序序列点位上无数据存在时，该点位的处理，默认填充为0。<br>- 堆叠：控制是否将数据堆叠显示。 |

| 配置项 | 说明 |
| --- | --- |
| 坐标轴 | - 位置：显示/隐藏坐标轴，及坐标轴的位置。<br>- 最大值/最小值：控制坐标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。<br>- 坐标轴尺度：控制坐标轴的尺度，支持线性尺度与对数尺度。 |

## 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。<br>- 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。 |

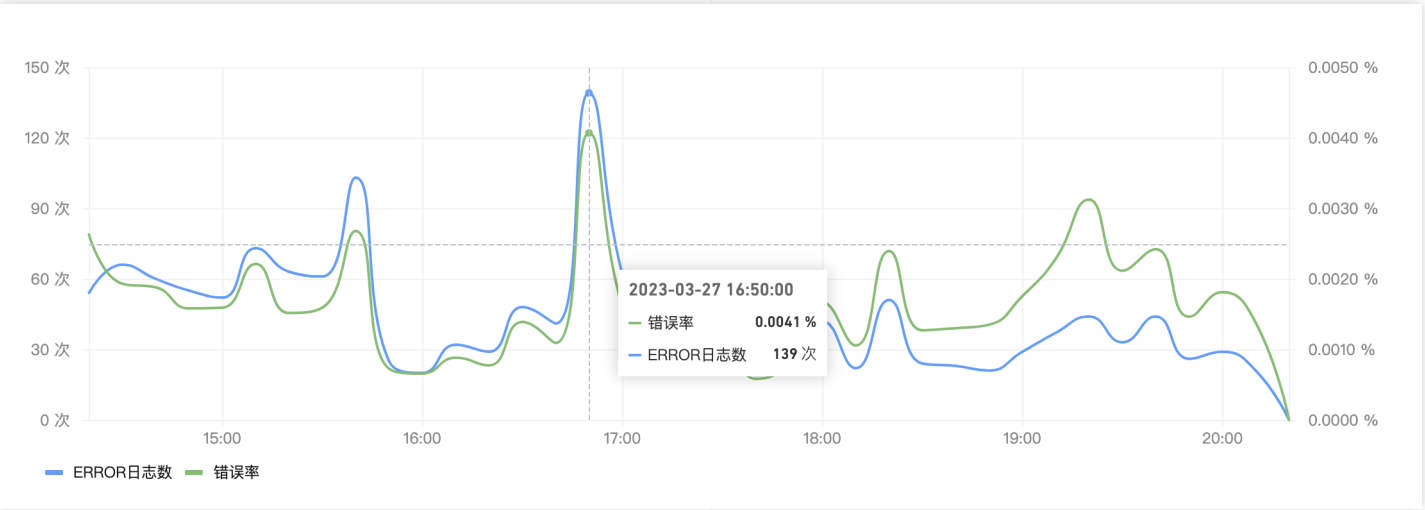
## 字段配置

| 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：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。

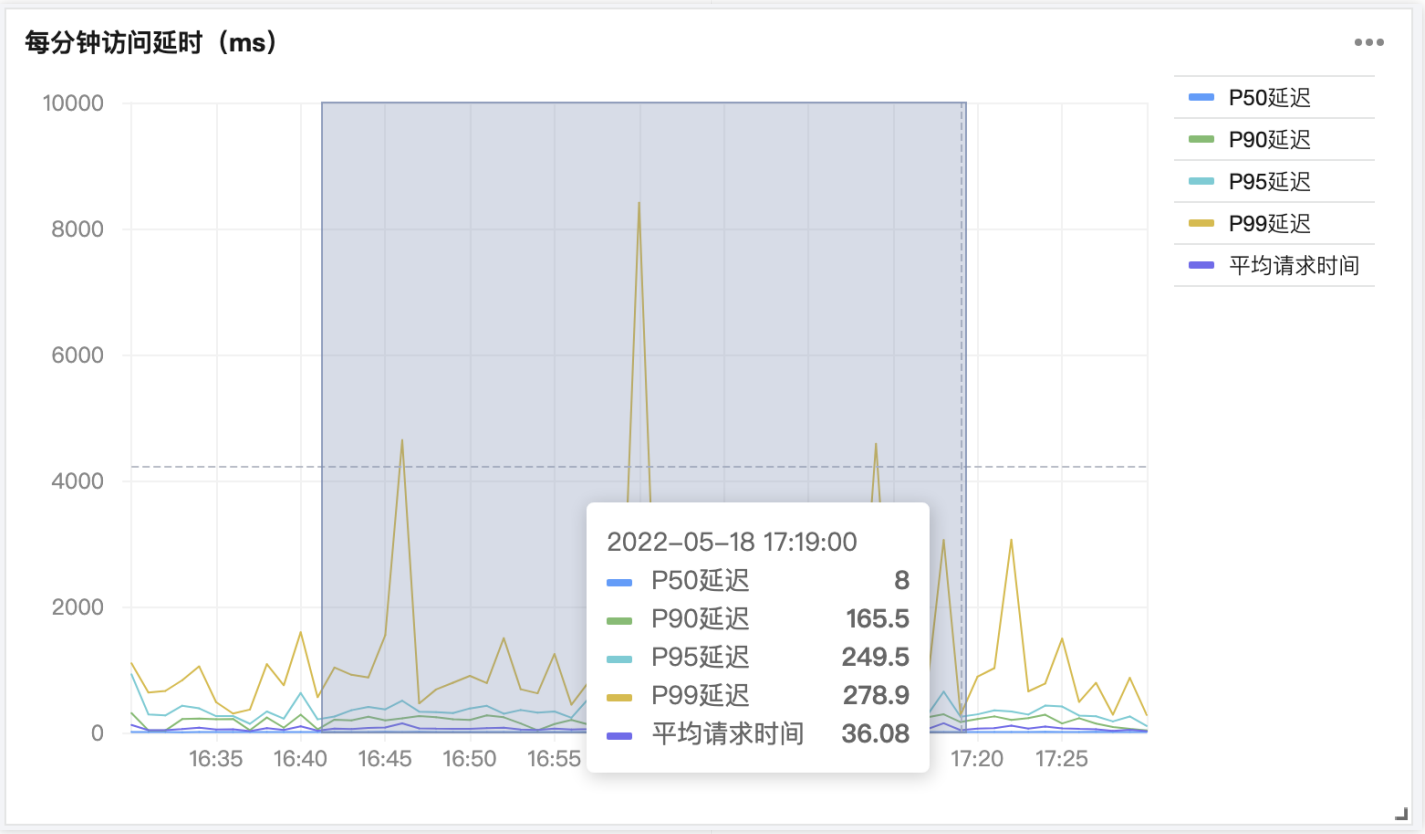



您可查看双坐标轴的效果：




# 图表操作

## 框选时间范围



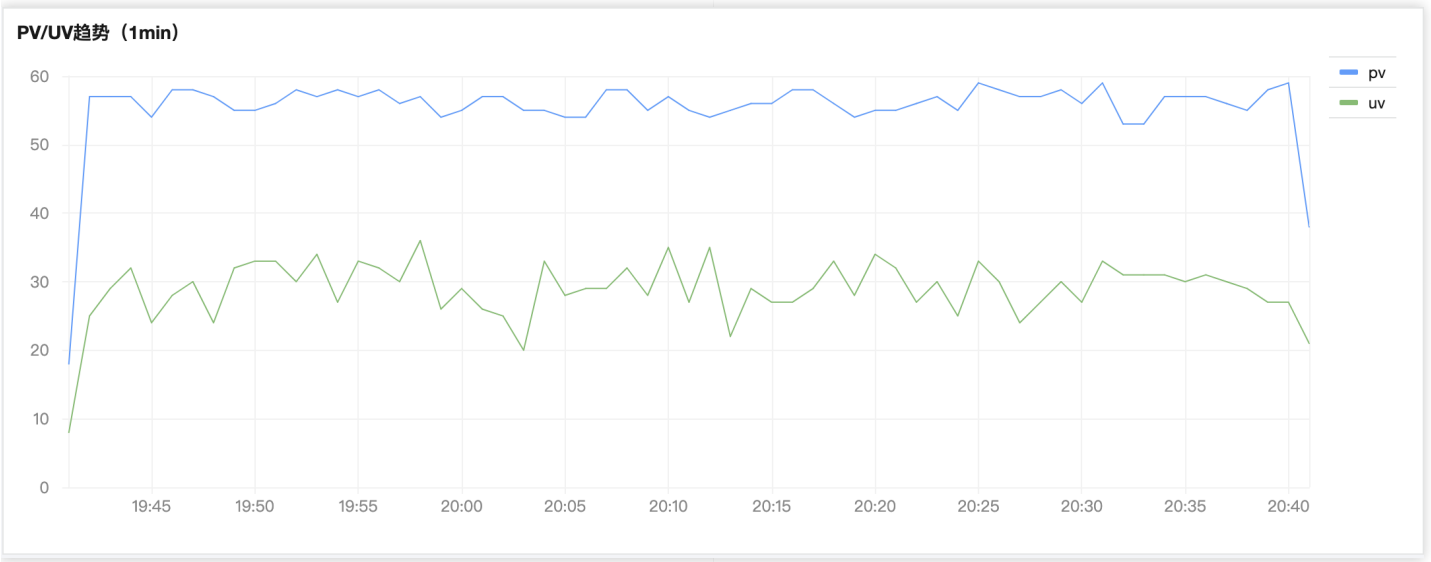

鼠标悬浮在图表上，长按后拖拽触发框选，时间范围将会使用框选区域的时间。适用于异常点的时间范围下钻等场景。

# 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。

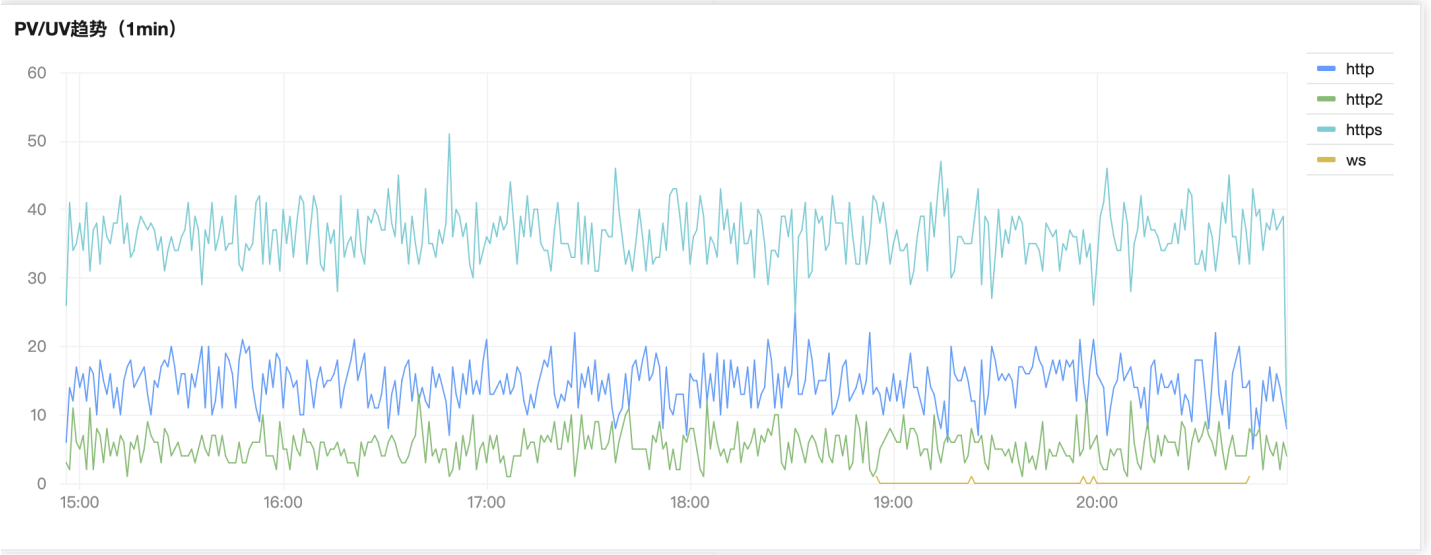
计算每分钟的 PV 与 UV：

```
* | 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
```



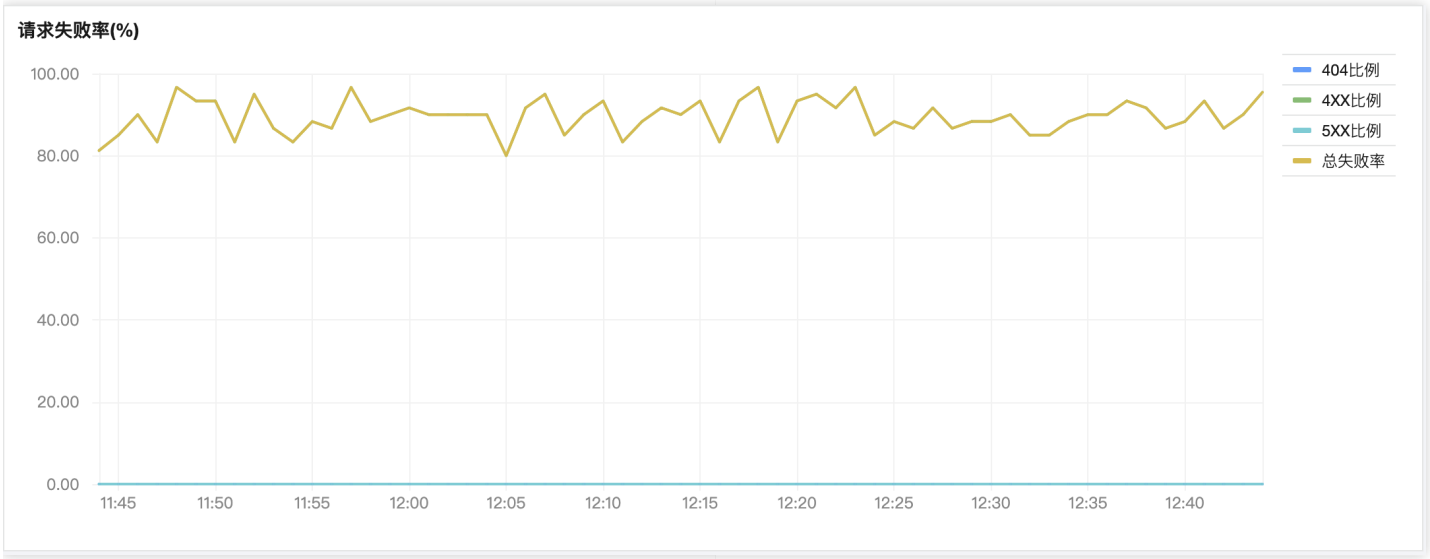

计算每分钟各协议类型的 PV：

```
* | 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
```



# 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，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。

## 图表配置

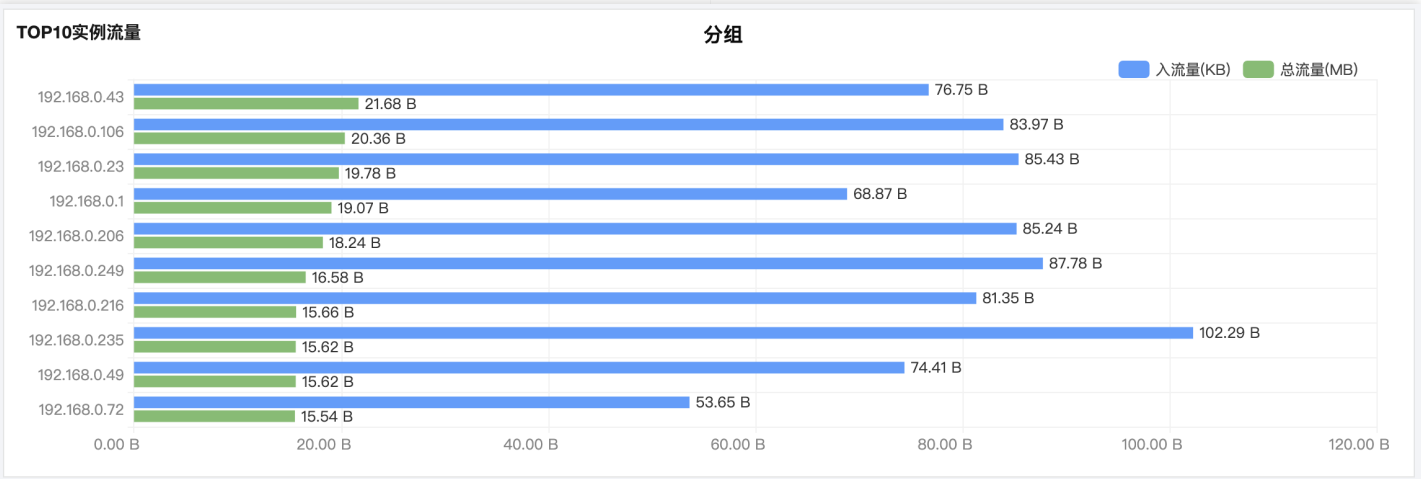
### 通用配置

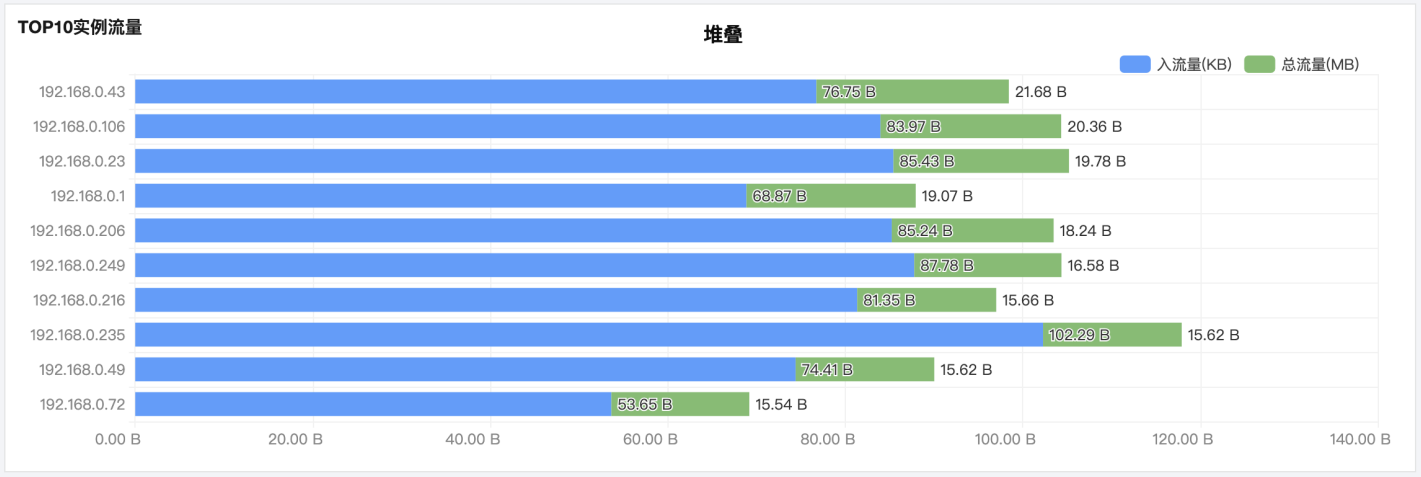
| 配置项 | 说明 |
| --- | --- |
| 基础信息 | 图表名称：设置图表的显示名称，可为空。 |
| 提示框 | 设置鼠标悬浮触发的气泡提示的内容样式。 |
| 标准配置 | 设置图表内所有指标类型的字段单位。详情请参见 单位配置。 |

### 柱状图配置

| 配置项 | 说明 |
| --- | --- |
| 柱状图 | - 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。<br>- 显示数值：控制是否显示每个条柱的数值标签。<br>- 条柱模式：支持条柱分组或堆叠显示。<br>- 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 | - 位置：显示/隐藏坐标轴，及坐标轴的位置。<br>- 最大值/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。 |

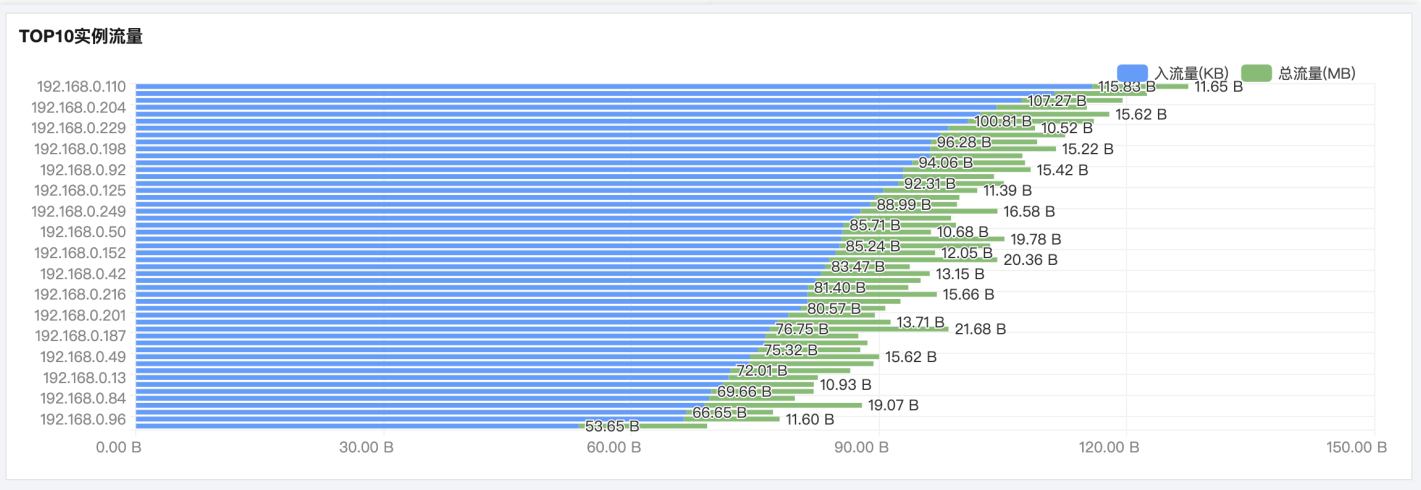
条柱模式示例：




# 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
```

# 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。

## 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。 |
| 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。 |

### 趋势对比

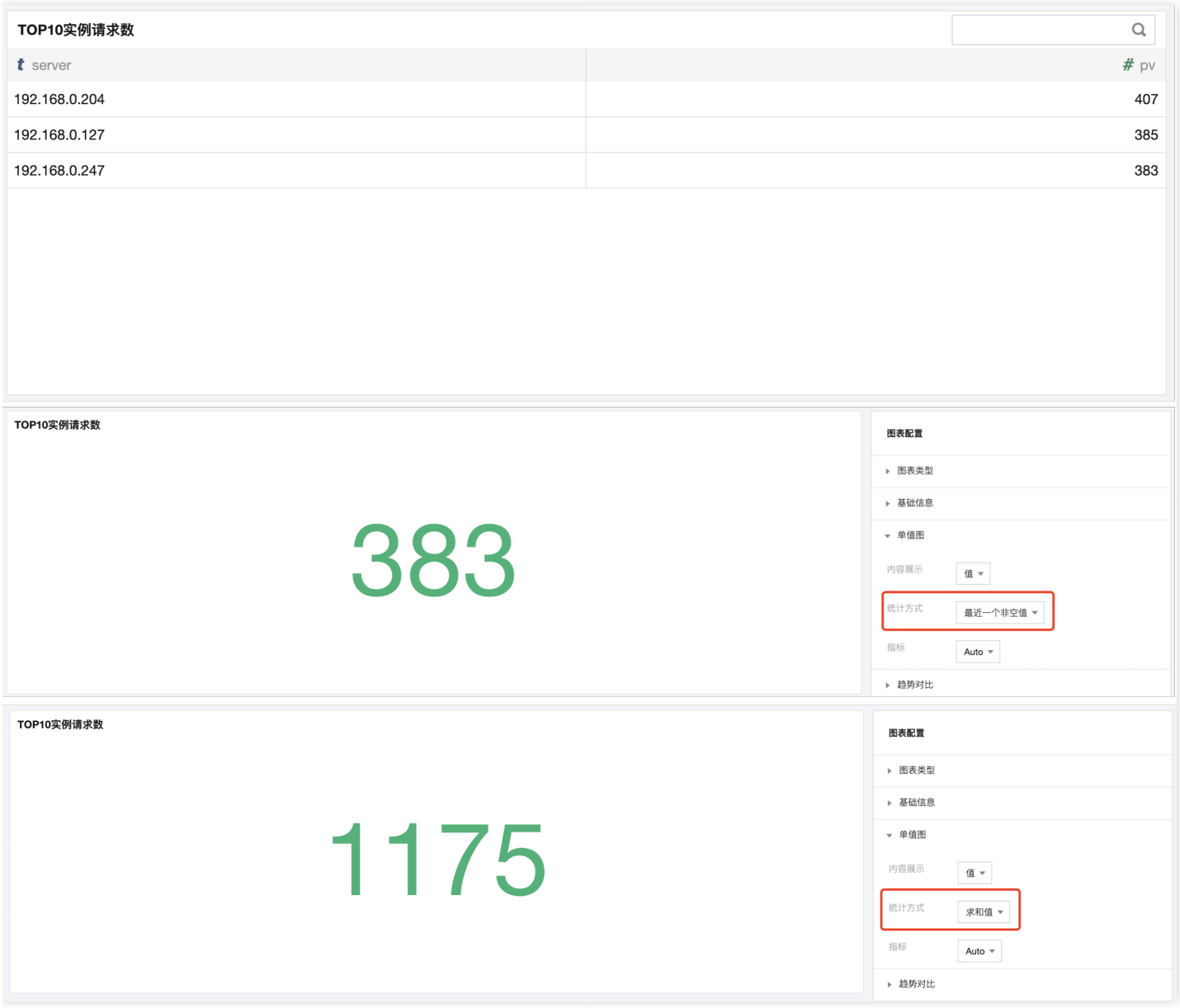
| 配置项 | 说明 |
| --- | --- |
| 趋势对比 | 开启趋势对比后，可以选择和 X 小时、天、月、年以前的同周期数据进行对比。趋势对比可以选择绝对值与百分比两类。 |




### 单值图配置

| 配置项 | 说明 |
| --- | --- |
| 单值图 | - 内容展示：控制是否显示单值图的指标名称。<br>- 统计方式：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。<br>- 指标：统计的目标指标，默认为自动，会选择返回的数据中排在第一的指标字段。<br>- 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3条数据，默认选择最近一个非空值，即排在末尾的值383显示。若修改为求和值，则会把3条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

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。

## 图表配置

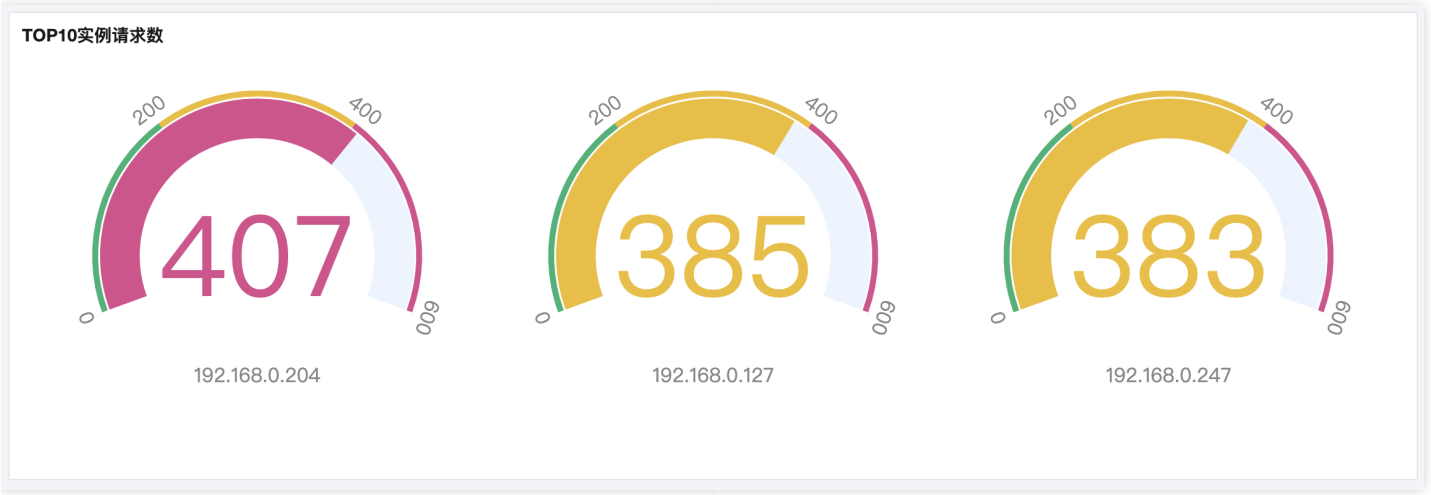
### 通用配置

| 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。 |
| 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。 |

### 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。<br>- 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。<br>- 显示阈值刻度：在图表上显示配置的阈值刻度。 |

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(%)"
```

# 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。

## 图表配置

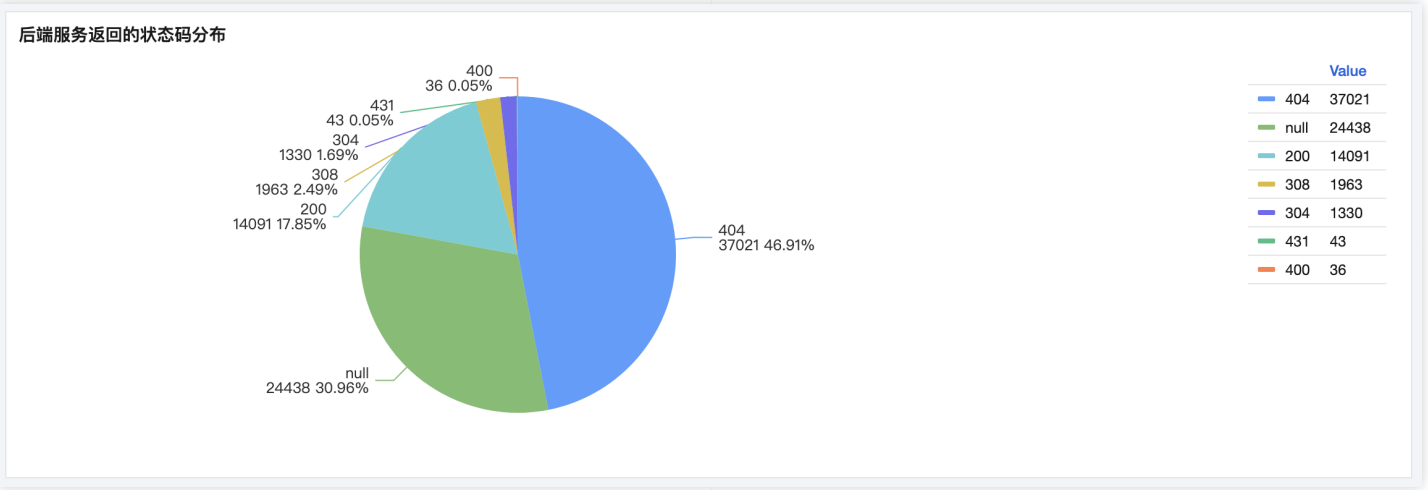
### 通用配置

| 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。 |
| 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。 |
| 自定义跳转 | 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。 |

### 饼图配置

| 配置项 | 说明 |
| --- | --- |
| 饼图 | - 展示方式：控制饼图的样式，实心为饼图，空心为环形图。<br>- 显示：扇区显示数值的方式。<br>- 选择所有值则会将每个单元格的数值单独作为一个扇区显示，支持配置显示的数值数量。<br>- 选择计算值则会将每列的数值聚合为一个扇区显示，支持配置聚合的统计方式。<br>- 指标：显示的指标字段，默认显示所有指标字段。<br>- 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。<br>- 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。<br>- 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。 |

标签示例：




# 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
```

# 桑基图

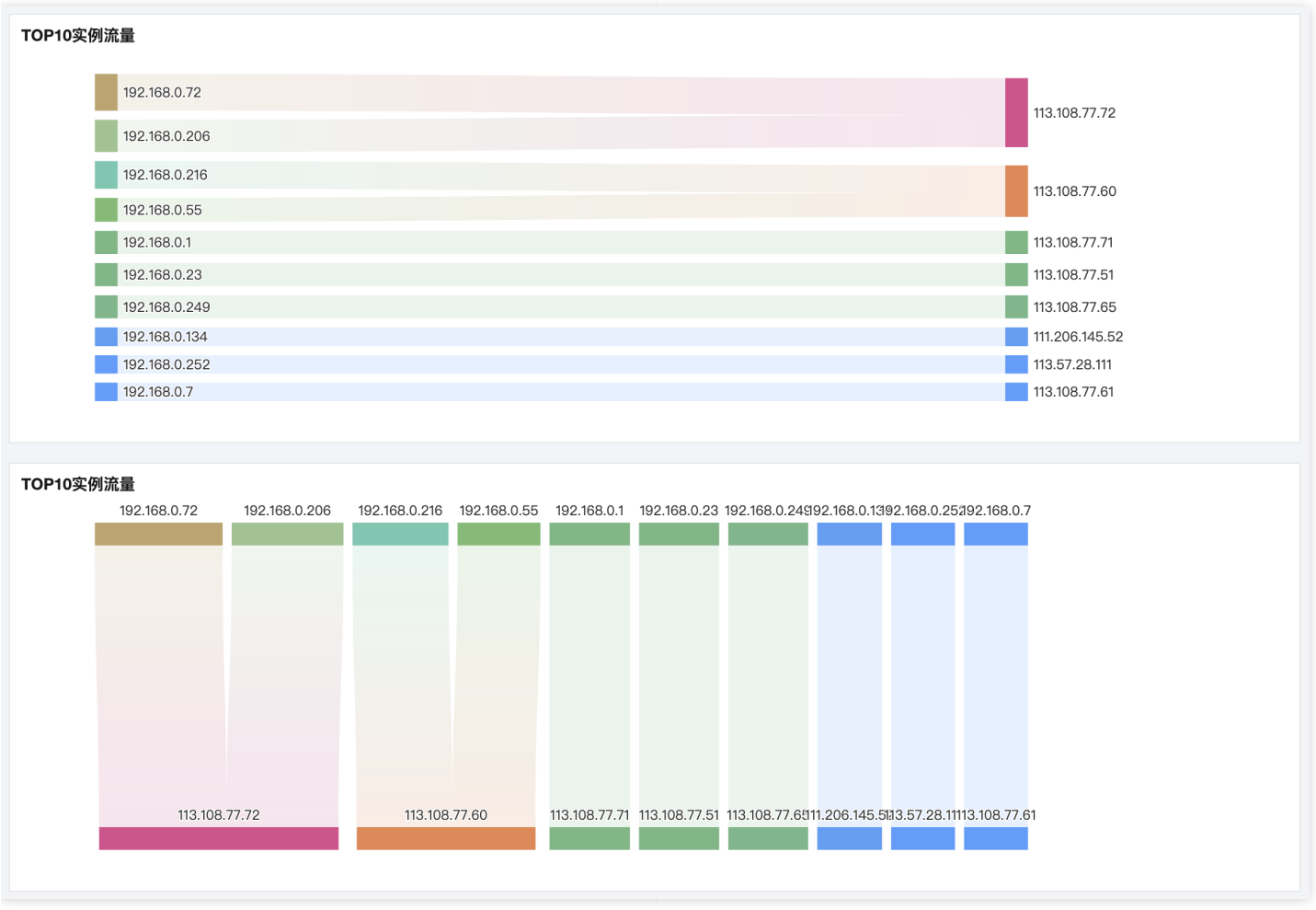
桑基图是一种特定类型的流图，用于描述一组值到另一组值的流向。防火墙目的与源 IP 流量统计等场景适用。

## 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。 |
| 桑基图 | 方向：设置桑基图的显示方向。 |

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
```

# 地图

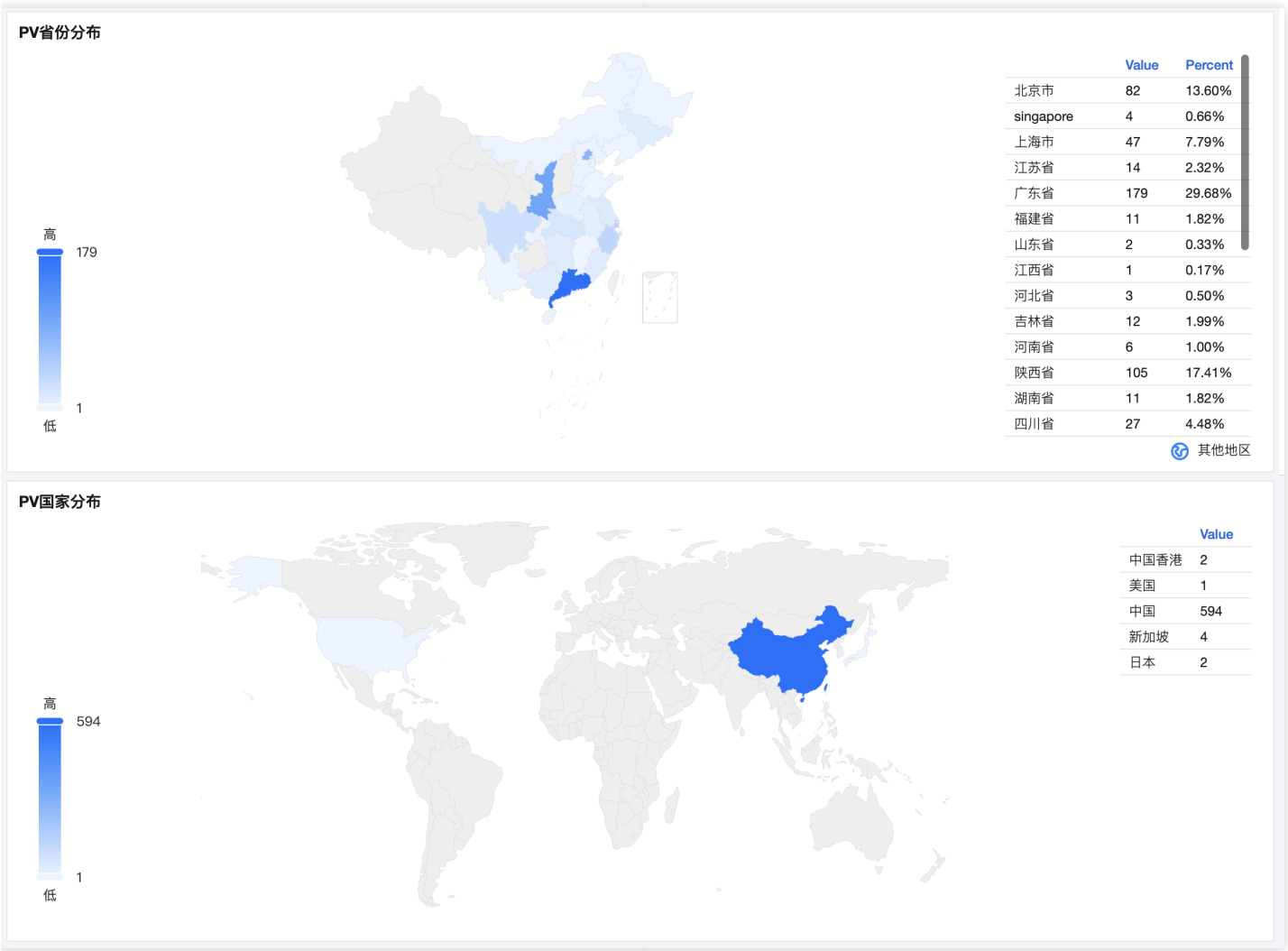
地图通过图形的位置来表现数据的地理位置，通常来展示数据在不同地理区域上的分布情况。攻击 IP 地理分布等地理位置统计场景适用。

## 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。 |
| 地图 | 区域定位：设置地图显示的区域范围，包含中国地图和世界地图两种。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。 |

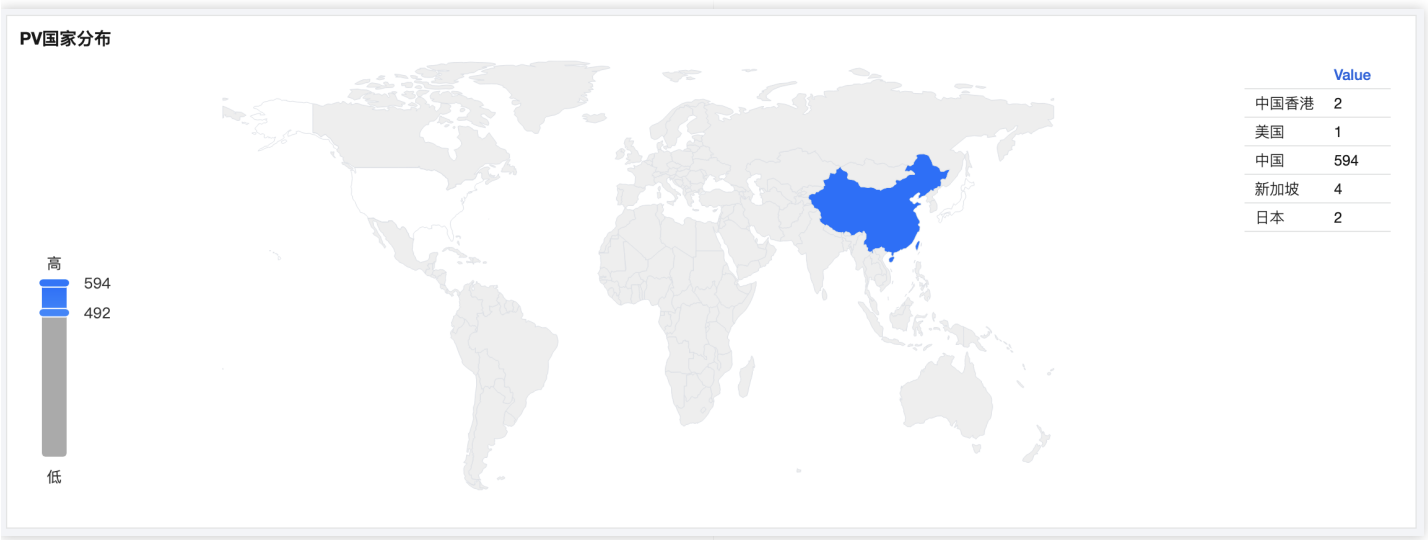
地图示例：




# 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。



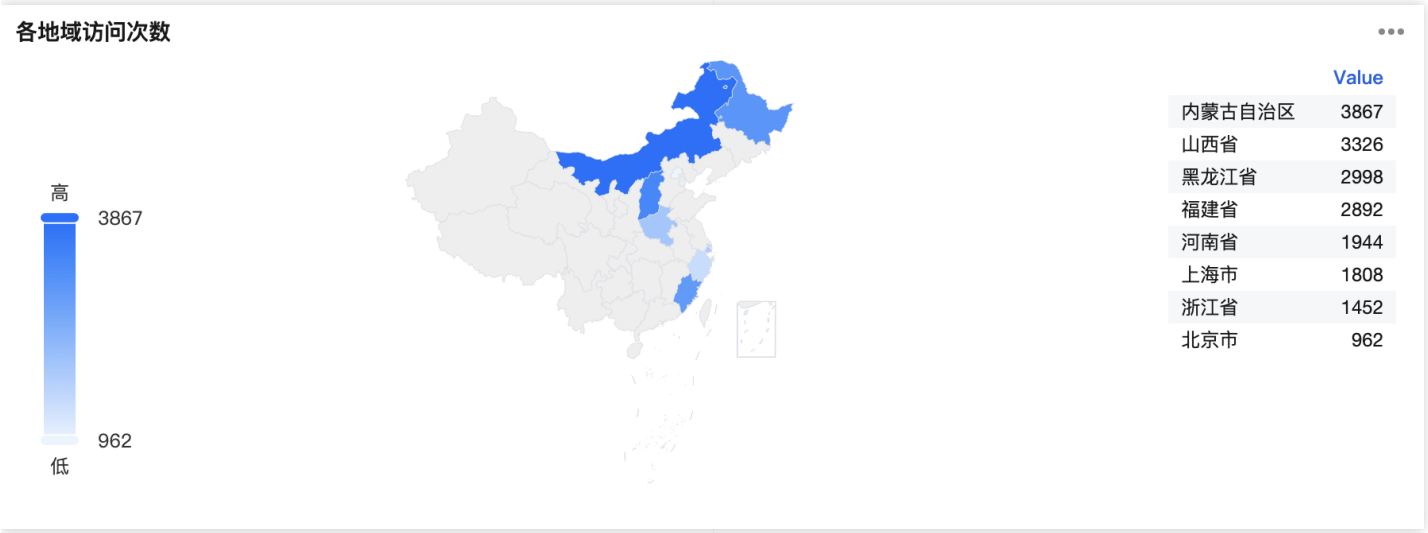

# 语句示例

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
```



# 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。

## 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。 |
| 词云 | - 显示词条数：控制最大显示多少条数据，取 TOPN，最大取100条。<br>- 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。 |

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

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。

## 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。 |
| 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。 |

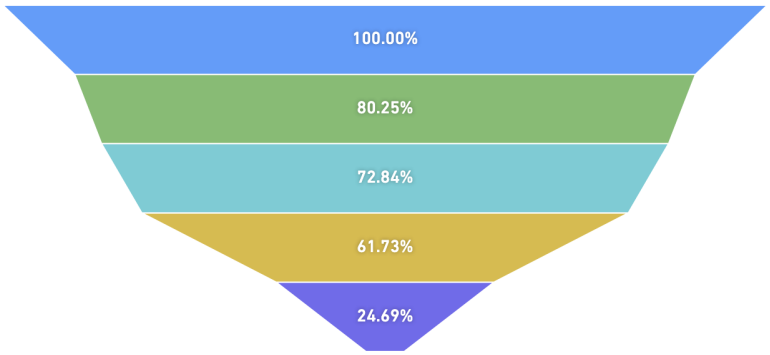
### 漏斗图配置

| 配置项 | 说明 |
| --- | --- |
| 漏斗图 | - 排序方式：漏斗图数值的排序。默认按照查询结果依次排列。<br>- 显示值：设置漏斗图每一层的标签显示形式，可以选择值或转化率。<br>- 标签：配置漏斗图标签的显示内容。<br>- 标签位置：配置标签在图表里的位置。<br>- 显示：漏斗图显示数值的方式<br>- 选择所有值则会将每个单元格的数值单独作为一层显示，支持配置显示的数值数量。<br>- 选择计算值则会将每列的数值聚合为一层显示，支持配置聚合的统计方式。<br>- 指标：显示的指标字段，默认显示所有指标字段。<br>- 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。 |

## 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
```




| | Value | Percent |
|---|---|---|
| /overview | 81 | 100.00% |
| /overview/books | 65 | 80.25% |
| /overview/books/detail | 59 | 72.84% |
| /overview/books/detail/like | 50 | 61.73% |
| /overview/books/detail/like/buy | 20 | 24.69% |

# 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，无需跳转到检索分析页面。

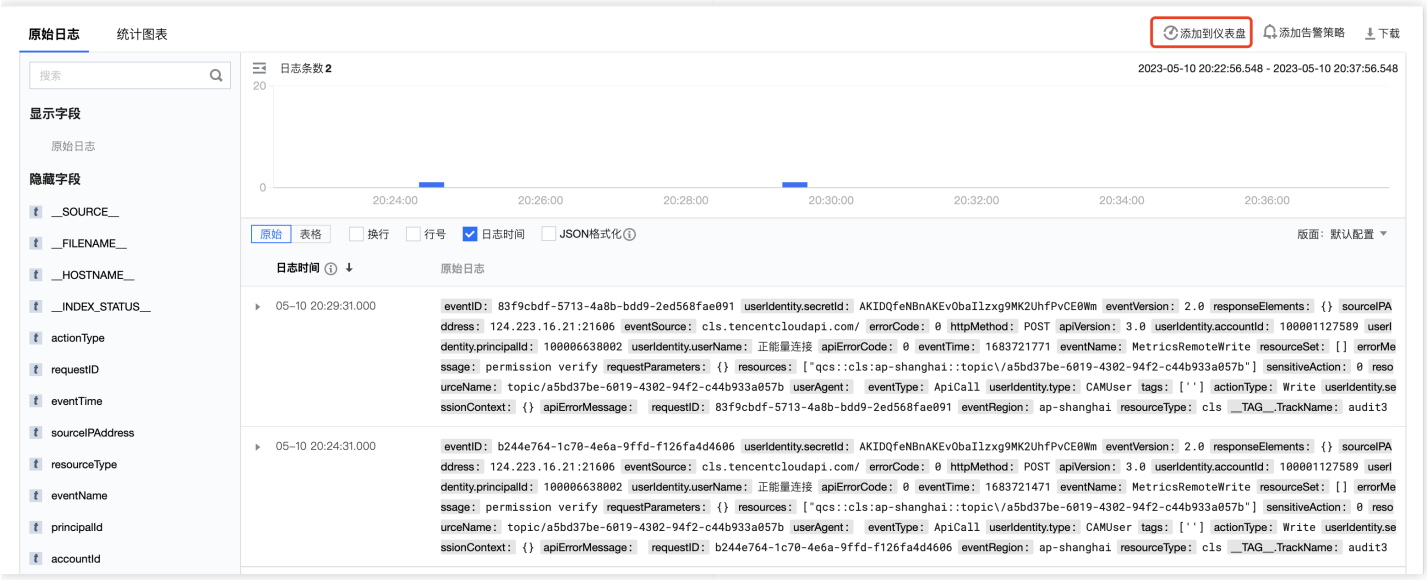
## 图表配置

| 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。<br>- 显示字段：选择需要展示的字段，为空时默认展示所有字段。<br>- 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。<br>- 行号：开启后显示日志行号。<br>- 日志时间：开启后显示日志时间。 |

## 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。



# 交互事件

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，创建一个表格类型的统计分析图表。
3. 在图表配置里选择自定义跳转，单击添加。
4. 填写跳转链接的相关配置，可在链接中引用变量，并设置是否在新窗口打开。

## 变量说明

| 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 |

## 自定义跳转案例

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。

**TOP10实例请求数**

| server | pv |
| --- | --- |
| 192.168.0.199 | 31 |
| 192.168.0.17 | 28 |
| 192.168.0.204 | 24 |
| 192.168.0.114 | 24 |
| 192.168.0.82 | 24 |
| 192.168.0.108 | 21 |
| 192.168.0.206 | 21 |
| 192.168.0.72 | 21 |
| 192.168.0.85 | 19 |
| 192.168.0.183 | 17 |

**TOP10实例请求失败率(%)**

| server | 失败率 |
| --- | --- |
| 192.168.0.110 | 12 % |
| 192.168.0.112 | 0 % |
| 192.168.0.114 | 0 % |
| 192.168.0.125 | 0 % |
| 192.168.0.127 | 0 % |
| 192.168.0.138 | 0 % |
| 192.168.0.152 | 0 % |
| 192.168.0.159 | 0 % |
| 192.168.0.104 | 0 % |
| 192.168.0.108 | 0 % |

**TOP10实例平均请求延...**

| server | 平均访问延时 |
| --- | --- |
| 192.168.0.152 | 113 |
| 192.168.0.17 | 96 |
| 192.168.0.114 | 88.54 |
| 192.168.0.108 | 80 |
| 192.168.0.31 | 79 |
| 192.168.0.125 | 78 |
| 192.168.0.210 | 78 |
| 192.168.0.191 | 78 |
| 192.168.0.233 | 78 |
| 192.168.0.19 | 78 |

选择检索其他日志主题，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。

**TOP10实例请求数**

| server | pv |
| --- | --- |
| 192.168.0.199 | 31 |
| 192.168.0.17 | 28 |
| 192.168.0.204 | 24 |
| 192.168.0.114 | 24 |
| 192.168.0.82 | 24 |
| 192.168.0.108 | 21 |
| 192.168.0.206 | 21 |
| 192.168.0.72 | 21 |
| 192.168.0.85 | 19 |
| 192.168.0.183 | 17 |

**TOP10实例请求失败...**

| server | 失败率 |
| --- | --- |
| 192.168.0.110 | 12 % |
| 192.168.0.11 | 0 % |
| 192.168.0.114 | 0 % |
| 192.168.0.125 | 0 % |
| 192.168.0.127 | 0 % |
| 192.168.0.138 | 0 % |
| 192.168.0.152 | 0 % |
| 192.168.0.159 | 0 % |
| 192.168.0.104 | 0 % |
| 192.168.0.108 | 0 % |

错误日志检索

**TOP10实例平均请求延...**

| server | 平均访问延时 |
| --- | --- |
| 192.168.0.152 | 113 |
| 192.168.0.17 | 96 |
| 192.168.0.114 | 88.54 |
| 192.168.0.108 | 80 |
| 192.168.0.31 | 79 |
| 192.168.0.125 | 78 |
| 192.168.0.210 | 78 |
| 192.168.0.191 | 78 |
| 192.168.0.233 | 78 |
| 192.168.0.19 | 78 |

# 数据转换

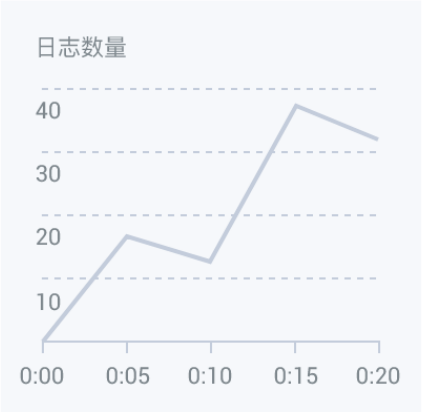
数据转换支持用户对检索结果进行二次处理，包括字段类型修改、选择制图字段、合并分组等。可以在不修改 SQL 语句的情况下，满足制图的需求。

## 开启数据转换

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. 选择统计图表页签，将下方的数据转换设置为  。
4. 在仪表盘图表编辑页面，选择数据转换页签，开启数据转换。




## 使用数据转换的情况




数据字段不匹配，请修改检索语句或进行手动数据转换

当前时序图只支持显示：

1个 时间维度，多个 # 指标

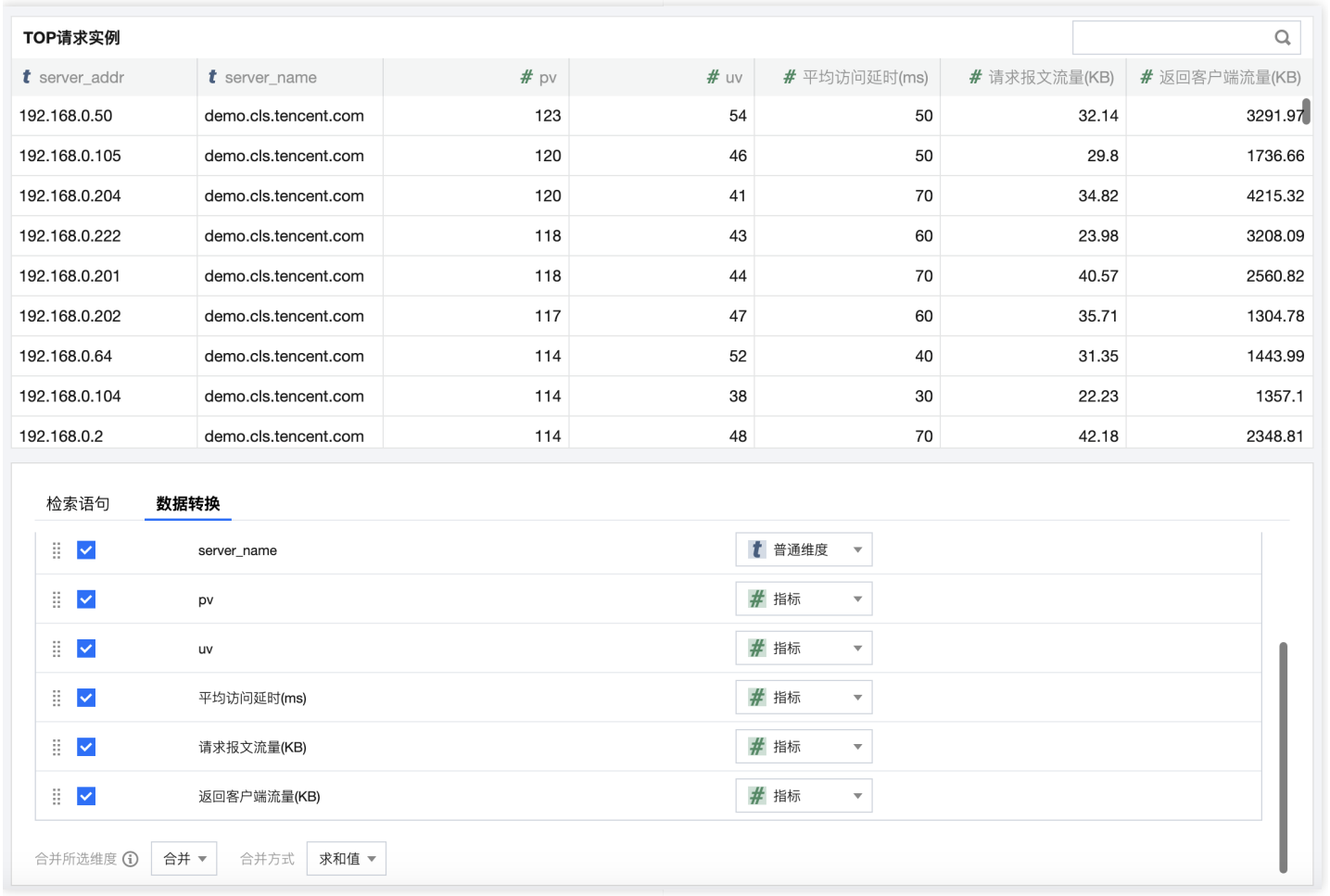
或

1个 时间维度，多个 t 普通维度，多个 # 指标

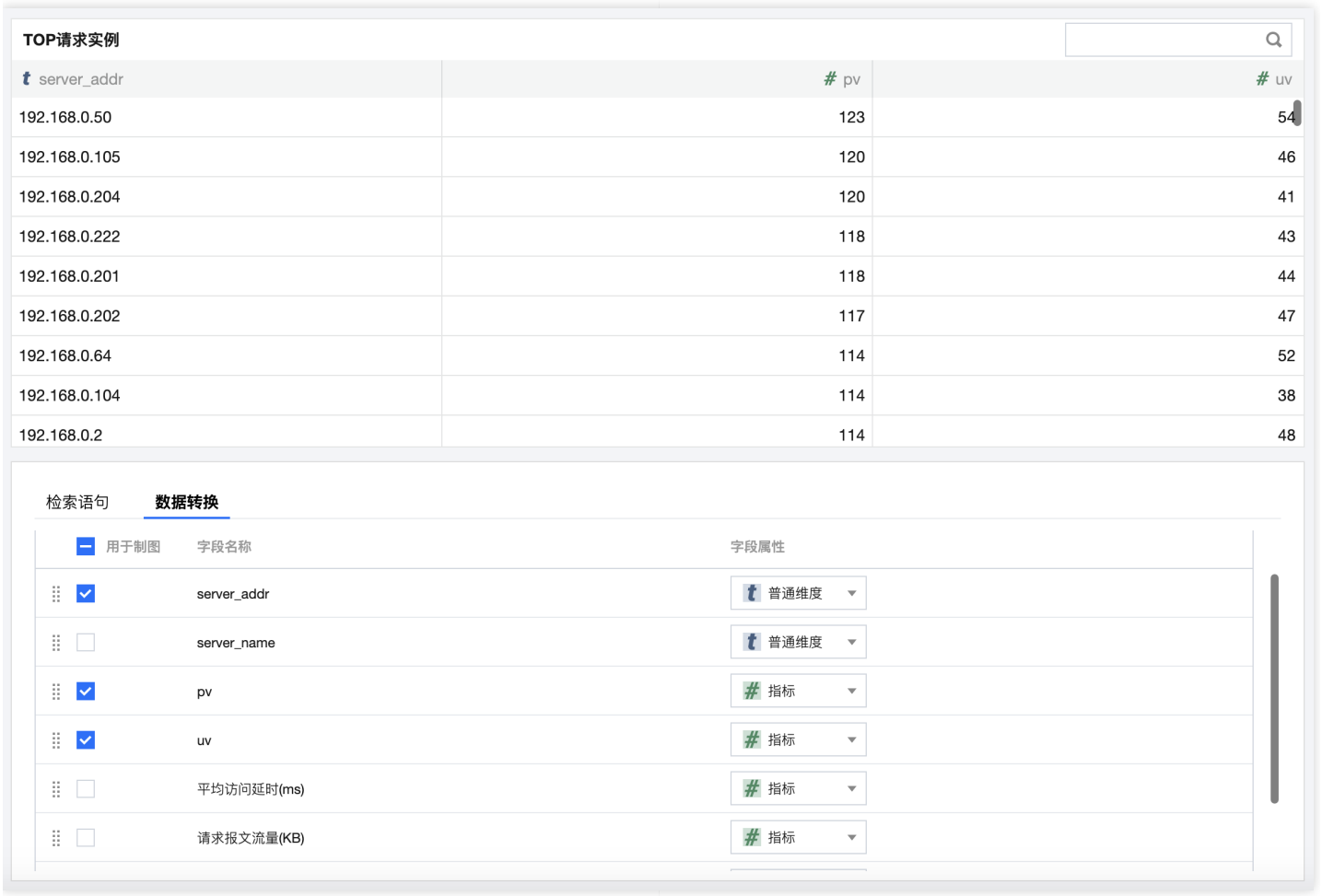
执行一个 SQL 分析语句后，想要在不同的图表类型上做可视化。因为不同的图表需要的字段数量与属性不同，在不匹配时就会出现以上情况。此时，用户可以通过修改 SQL，使其适配图表要求。也可以通过数据转换配置，对已有结果进行二次处理，满足制图需求。

# 使用数据转换

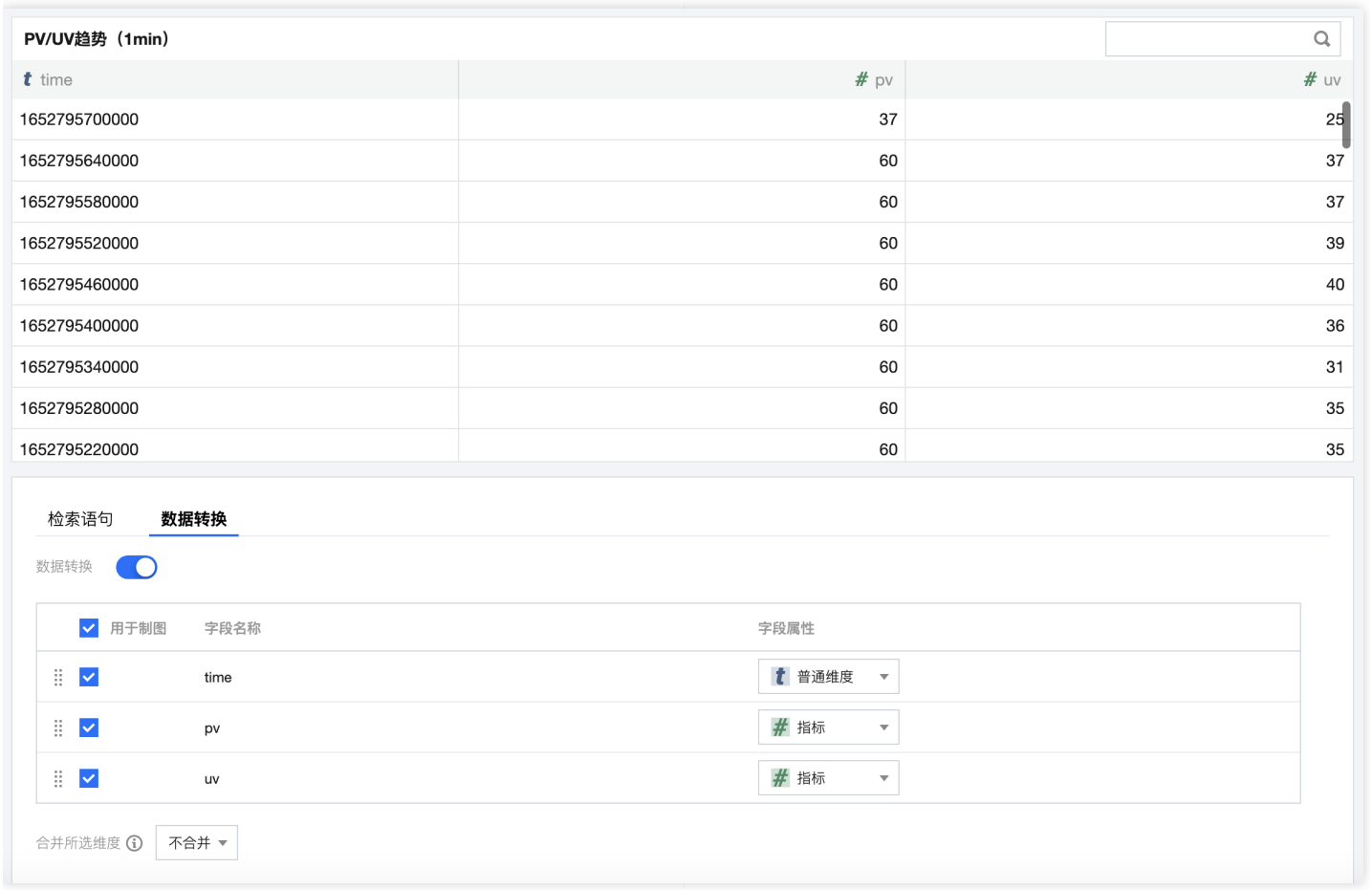
## 选择制图字段




通过 SQL 检索出了以上列表中的结果，此时通过勾选字段列表中的字段，可以控制图表选中指定字段来制图，等价于 select。以下为选中部分字段后的结果：


TOP请求实例
server_addr
pv
uv
192.168.0.50 123 54
192.168.0.105 120 46
192.168.0.204 120 41
192.168.0.222 118 43
192.168.0.201 118 44
192.168.0.202 117 47
192.168.0.64 114 52
192.168.0.104 114 38
192.168.0.2 114 48
检索语句
数据转换
用于制图
字段名称
字段属性
server_addr 普通维度
server_name 普通维度
pv 指标
uv 指标
平均访问延时(ms) 指标
请求报文流量(KB) 指标

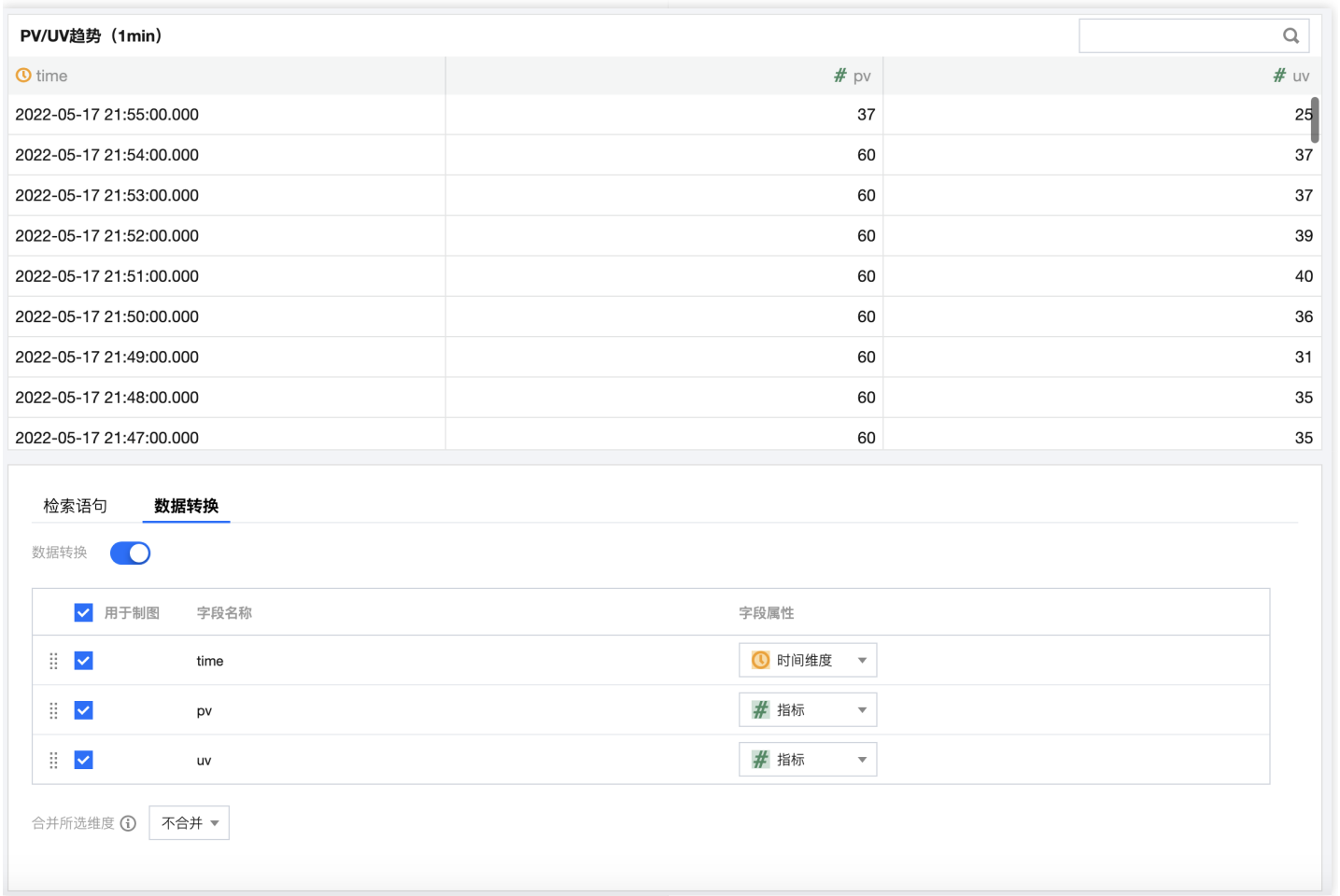
# 转换字段类型



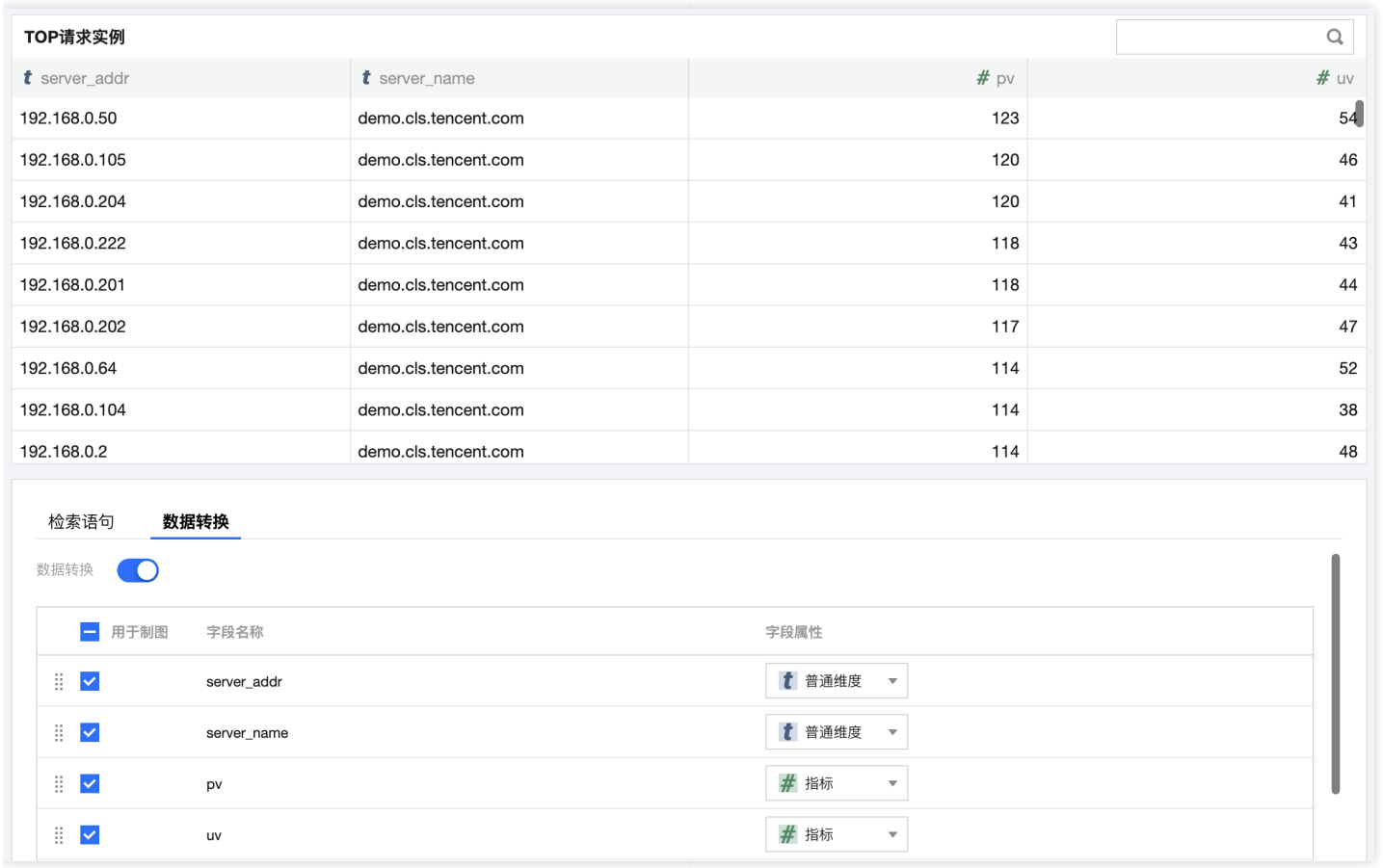

SQL 统计的结果中，每个字段都有一个默认的类型，图表依据类型识别 #数值类型 的字段为指标， t字符串类型 的字段为普通维度， 时间类型 的字段作为时间维度。并将其与制图所需的字段属性对应。例如在时序图中，就需要时间维度字段作为 X 轴，指标字段作为 Y 轴。

上图示例中的 time 字段被当做了一个普通维度，此时，无法满足时序图的字段属性要求。而通过修改 time 字段属性为时间类型，会发现 time 字段转变为时间格式（等价于 CAST 函数），此时即可使用时序图。

常见用户使用 histogram 函数处理字段，其结果可以是非标准的时间格式，因此若默认为普通维度，无法使用时序图。需要提前通过 cast 函数转换为时间类型，或者通过数据转换修改字段属性为时间维度。


PV/UV趋势（1min）
time
# pv
# uv
2022-05-17 21:55:00.000 37 25
2022-05-17 21:54:00.000 60 37
2022-05-17 21:53:00.000 60 37
2022-05-17 21:52:00.000 60 39
2022-05-17 21:51:00.000 60 40
2022-05-17 21:50:00.000 60 36
2022-05-17 21:49:00.000 60 31
2022-05-17 21:48:00.000 60 35
2022-05-17 21:47:00.000 60 35
检索语句
数据转换
数据转换
用于制图
字段名称
字段属性
time
时间维度
pv
指标
uv
指标
合并所选维度
不合并

# 合并分组

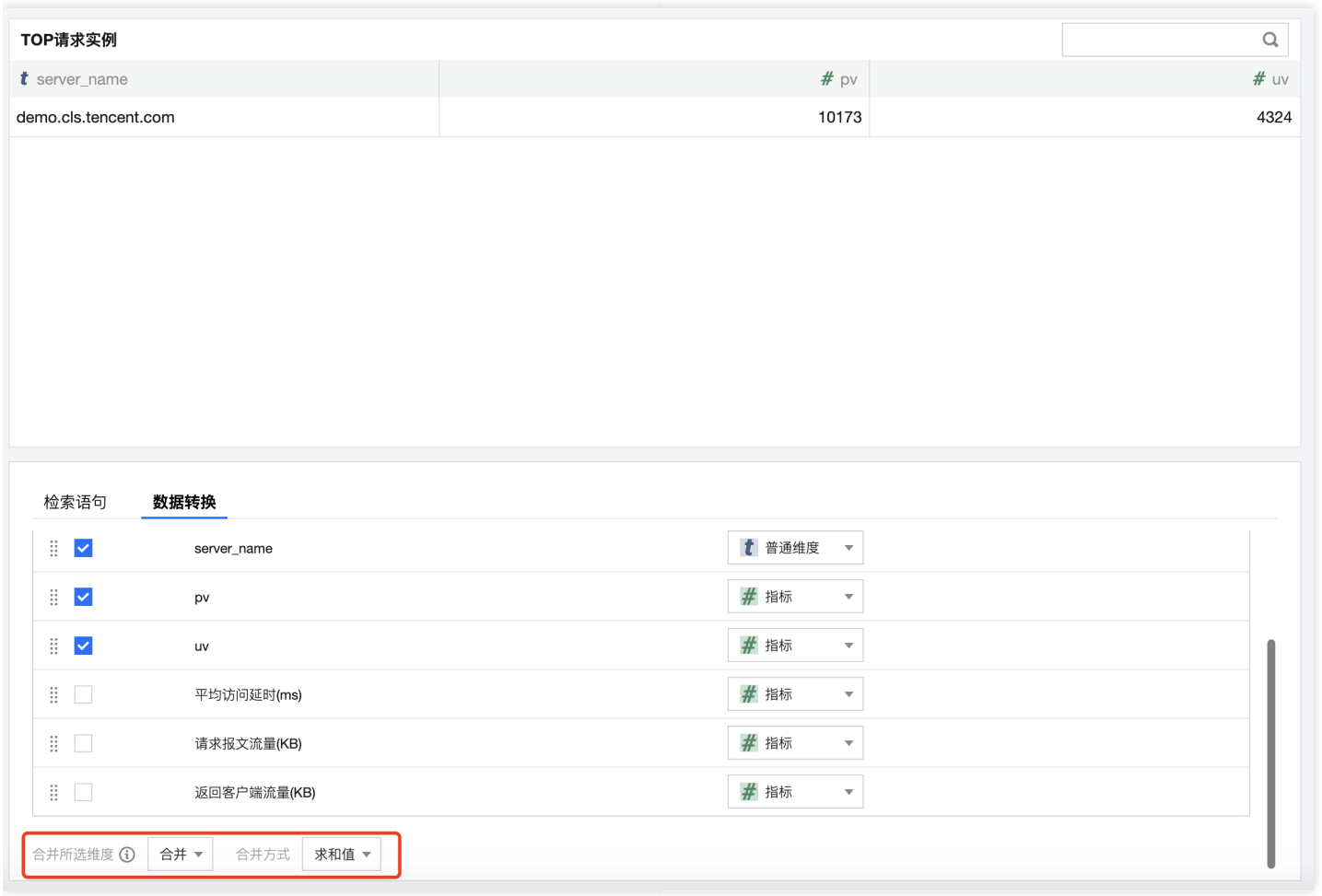


TOP请求实例

| server_addr | server_name | pv | uv |
|---|---|---|---|
| 192.168.0.50 | demo.cls.tencent.com | 123 | 54 |
| 192.168.0.105 | demo.cls.tencent.com | 120 | 46 |
| 192.168.0.204 | demo.cls.tencent.com | 120 | 41 |
| 192.168.0.222 | demo.cls.tencent.com | 118 | 43 |
| 192.168.0.201 | demo.cls.tencent.com | 118 | 44 |
| 192.168.0.202 | demo.cls.tencent.com | 117 | 47 |
| 192.168.0.64 | demo.cls.tencent.com | 114 | 52 |
| 192.168.0.104 | demo.cls.tencent.com | 114 | 38 |
| 192.168.0.2 | demo.cls.tencent.com | 114 | 48 |



通过 SQL 检索出以上结果，按照 "server_addr" 与 "server_name" 进行分组，统计每个分组的 PV、UV。此时，想要在当前的结果上对 "server_name" 进行合并分组统计，可以通过隐藏 "server_addr" 字段，然后合并所选的普通维度 "server_name"，等价于 Group By 函数。得到的结果如下：


TOP请求实例
server_name
pv
uv
demo.cls.tencent.com
10173
4324
检索语句
数据转换
server_name
普通维度
pv
指标
uv
指标
平均访问延时(ms)
指标
请求报文流量(KB)
指标
返回客户端流量(KB)
指标
合并所选维度
合并
合并方式
求和值

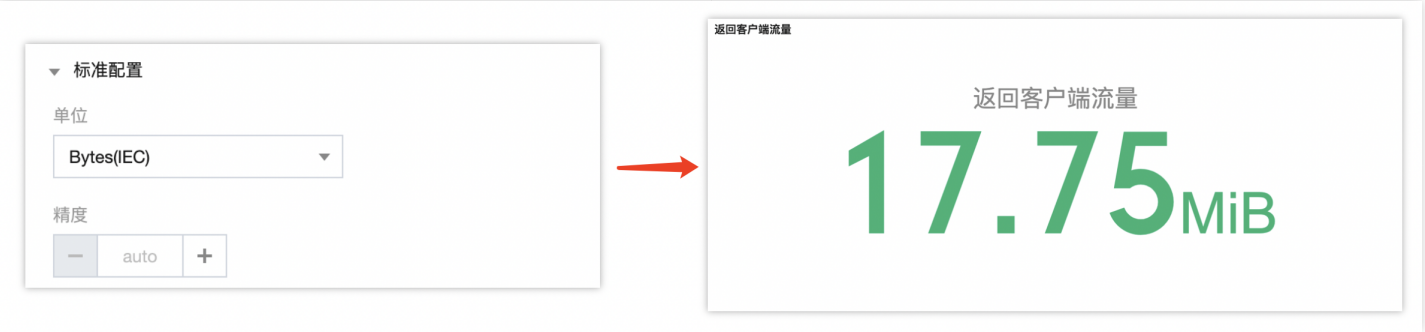
# 单位配置

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。


标准配置
单位
Custom: 字节
精度
auto
返回客户端流量
返回客户端流量
18616608字节

# 图表自定义时间

## 操作场景

仪表盘图表支持自定义时间配置。与自定义时间相对的是仪表盘全局时间。仪表盘图表创建时，默认使用仪表盘全局时间，其数据范围受仪表盘全局时间控制。修改仪表盘全局时间，所有关联图表的时间范围都会发生改变。而配置自定义时间后，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仪表盘，进入仪表盘页面。
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. 单击添加图表，选择自定义图表，进入图表编辑页面。
5. 单击图表时间范围，开启图表自定义时间。




6. 查看图表，图表时间与仪表盘时间不同。

# 过滤与变量

## 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 |

## 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，单击仪表盘，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，空值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。 |

| 表单元素 | 说明 |
| --- | --- |
| 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，单击仪表盘，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。

## 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，单击仪表盘，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 |
| --- | --- |
| 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，仅支持字母与数字。 |
| 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。 |
| 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，单击数据来源，勾选使用数据源变量，选择刚新建的模板变量。

8. 单击应用到仪表盘。

9. 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，单击仪表盘，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。



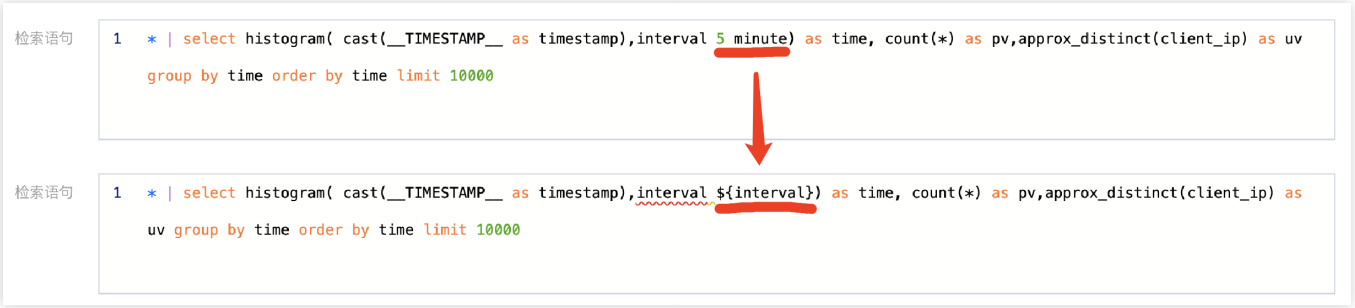

| 表单元素 | 说明 |
| --- | --- |
| 类型 | 变量的类别，不同的类别对应不同的配置项与应用场景，此处选择自定义变量。 |
| 变量名称 | 查询检索语句中变量的命名，仅支持字母与数字。通过${变量名称}的格式引用变量。 |
| 变量别名 | 在仪表盘上变量的显示名称，非必填项目，为空时自动使用变量名称为显示名称。 |
| 静态变量值 | 手动添加静态变量值，静态变量值将常驻显示，支持配置别名。 |
| 动态变量值 | 开启后，可以选择一个日志主题并输入检索分析语句，取检索分析的结果作为可选变量值。 |
| 默认值 | 默认使用变量值，为必填项。 |

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 ${interval} ，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，单击上方的时间颗粒度变量下拉框，修改时间颗粒度，发现插入该变量的图表发生对应的变更。

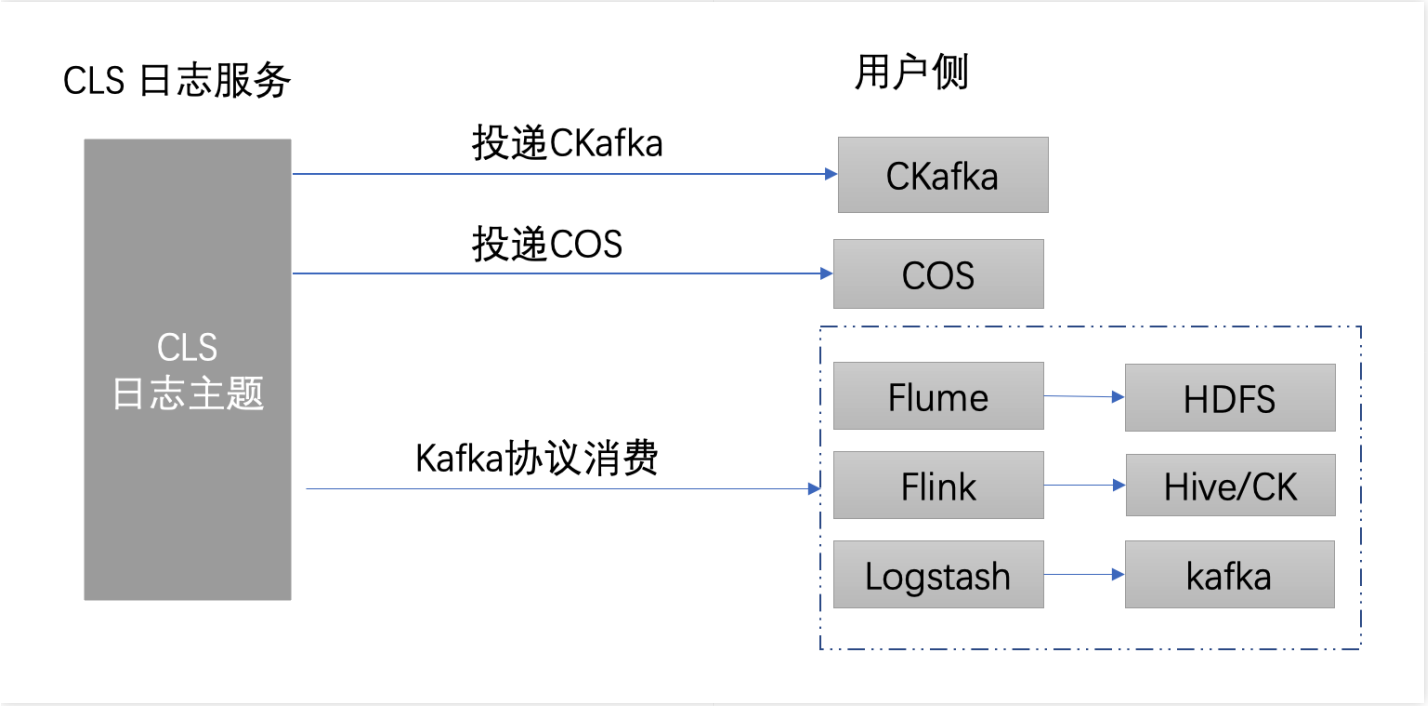
## 常见问题

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。

# 投递与消费
# 投递与消费概述

日志服务（Cloud Log Service，CLS）可以将日志主题中的数据投递到 COS、Ckafka，以满足归档或者计算的需求。
投递和消费都会产生读流量。
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。
- 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。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。这样可以无需 Kafka 实例，投递和消费的读流量基本相同。

## 功能特性

| 产品功能 | 说明 |
| --- | --- |
| 投递到 COS | CSV<br>可用于日志数据批处理。 |

| 产品功能 | 说明 |
| --- | --- |
| | 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）。<br>JSON<br>日志数据计算。<br>Parquet<br>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 | 投递数据格式：原文、JSON<br>使用场景：日志流处理 |
| Kafka 协议消费 | 投递数据格式：原文、JSON<br>使用场景：日志流处理 |

> 注意
>
> - 请提前准备 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 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。

# 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

## 操作场景

当您创建投递到 COS/CKafka 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 的权限。大部分用户通过控制台操作，系统会引导用户完成授权，小部分用户会直接使用 API，这时候需要手动去授权。手动授权之前，可以先通过以下步骤检查是否已经完成了对 CLS 服务角色的授权。

## 检查是否已经授权 CLS

1. 登录访问管理控制台。
2. 在左侧导航栏中，选择角色 > 角色管理。
3. 在角色管理页面中，查看是否存在 CLS_QcsRole 角色。可通过列表右上角的搜索框进行搜索。




4. 单击角色名称，进入角色详情页。
   - 在已关联的策略中，查看是否已具备 QcloudCOSAccessForCLSRole 及 QcloudCKAFKAAccessForCLSRole 权限。
   - 在角色载体中，查看角色载体是否关联受信主账号。

   若不具备角色及相关权限，则请参考下文进行创建。

## 操作步骤

### 授权 CLS 访问权限

您可通过以下方式授权CLS投递 COS/CKafka 权限：

通过控制台自动创建

您首次通过控制台创建投递到 COS/CKafka 的任务时，请根据页面指引创建角色和策略。

使用访问管理手动创建

1. 登录访问管理控制台。

2. 在左侧导航栏中，选择角色>角色管理。

3. 单击新建角色，输入角色信息后，单击下一步：配置角色策略。

4. 使用 clsrole 进行搜索，并在结果中勾选 QcloudCKAFKAAccessForCLSRole 及 QcloudCOSAccessForCLSRole 策略后，单击下一步：审阅。




5. 在审阅页面中，确认信息后，单击完成。

# 投递到对象存储
# 新建投递任务

## 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。
3. 子账号/协作者需要主账号授权。
4. 已授权给 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名称，进入日志主题管理页面。

4. 单击投递到对象存储页签，进入配置页面，单击添加投递配置。
   配置项说明如下：

| 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 |
| 对象存储存储桶 | 与当前日志主题同地域的存储桶作为投递目标存储桶 | 列表选择 | 必填 |
| 对象存储路径 | 对象存储存储桶的路径。默认按照/年/月/日/小时/，如/2022/7/31/14/ 在对象存储上来存储投递的日志文件，路径配置支持 strftime 的语法 ，例如： | 非/开头 | 可选 |

| 配置项 | 解释说明 | 规则 | 是否必填 |
| --- | --- | --- | --- |
| | - /%Y/%m/%d/生成的路径是/2022/7/31/<br>- /%Y%M%d/%H/生成的路径是/20220731/14/ | | |
| 文件命名 | - 投递时间命名，默认选项，例如202208251645_000_132612782.gz代表的是投递时间_日志主题分区_offset，Hive 可加载此文件<br>- 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 |
| 压缩格式 | 为了帮助用户节约读流量，我们将日志文件压缩后再投递到 COS，支持 Snappy\lzop\gzip | gzip\snappy\lzop | 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<br>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<br>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 | 必填 |

5. 单击下一步，进入高级配置，您可以选择数据的格式，以 JSON/CSV 进行投递。

| 投递数据格式 | 使用场景 |
| --- | --- |
| 按 CSV 格式投递 | - 可用于 DLC 入湖计算<br>- 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 |

## 投递格式为 JSON

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。

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。<br>例子：<br>- 日志原文： {"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} |

| 转义选项 | 说明 |
| --- | --- |
| | - 投递到 COS： {"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}}<br>注意：<br>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;<br>日志原文： {"a":123, "b":"123", "c":"-123", "d":"123.45", "e":{"e1":123,"f1":"123"}}<br>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。<br>例1：<br>- 日志原文： {"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}}<br>- 投递到 COS: {"a":"aa","b":"{\"b1\":\"b1b1\", \"c1\":\"c1c1\"}"}<br>例2：<br>- 日志原文： {"a":123, "b":"123", "c":"-123", "d":"123.45", "e":{"e1":123,"f1":"123"}}<br>- 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 | 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 |

# 投递日志原文

## 使用场景

如果您想把采集的日志原文，投递到对象存储，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、单竖线的情况 |
| 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。
   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、单竖线。 |
| 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

日志服务 CLS 提供了投递任务管理的功能，您可以查看每个时间点的投递任务，可根据时间、日志集、日志主题、投递任务名称，查看相关的投递任务。

## 查看投递任务

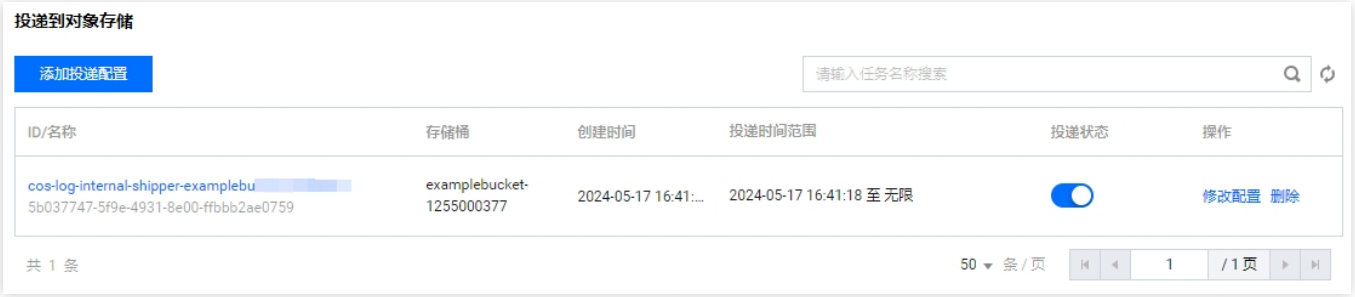
1. 登录日志服务控制台。
2. 在左侧导航栏中，选择投递任务管理，进入投递任务管理页面。
3. 在投递到对象存储页面，选择投递任务的地域、日志集、日志主题，查看投递任务列表。
4. 单击任务名称，即可查看任务的详情和监控信息。
    - 投递流量：投递对象存储成功的流量。当用户配置的文件大小或者投递间隔时间触发投递时，会有投递日志流量值，其余时间值为0。
    - 投递行数：投递对象存储成功的行数。当用户配置的文件大小或者投递间隔时间触发投递时，会有投递日志行数值，其余时间值为0。
    - 投递延时：Lantency 指的是日志数据到达 CLS 之后，等待被投递任务处理的时间，60秒内都属于正常。 该指标主要监控有无数据积压，日志是否被及时的投递到对象存储。
    - 投递任务异常记录：
        - 10001为 对象存储桶不存在，请检查您的对象存储桶的有效性。
        - 10002为 对象存储桶没有权限，请确认您有访问该对象存储桶的权限。
        - 10003为内部错误，请重试。

## 修改投递配置

1. 登录日志服务控制台。
2. 在左侧导航栏中，选择投递任务管理，进入投递任务管理页面。
3. 在投递到对象存储页面，选择投递任务的地域、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，即可修改投递配置。

## 关闭投递任务

1. 登录日志服务控制台。
2. 在左侧导航栏中，选择投递任务管理，进入投递任务管理页面。
3. 在投递到对象存储页面，选择投递任务的地域、日志集、日志主题，找到对应的投递任务，关闭投递状态。




4. 在弹窗中单击确定关闭，即可关闭日志投递。

# 删除投递任务

1. 登录日志服务控制台。
2. 在左侧导航栏中，选择投递任务管理，进入投递任务管理页面。
3. 在投递到对象存储页面，选择投递任务的地域、日志集、日志主题，找到对应的投递任务，单击右侧的删除。
4. 在弹窗中单击删除，即可删除日志投递。

# 配置告警

## 配置 CLS 投递 COS 任务的告警

1. 登录云监控控制台。
2. 在左侧导航栏中，选择告警配置 >告警策略。
3. 单击新建，根据实际情况，配置相关信息。
    - 策略名称：自定义，最多30个字符。
    - 策略类型：选择日志服务 / COS投递状态监控。
    - 告警对象：选择您投递对象存储的日志主题。
    - 触发条件：按需设置触发告警的判断条件。
4. 单击完成。

# 投递到 CKafka
# 创建投递任务

## 操作背景

您可以将日志主题的数据投递到 CKafka，然后用于您的实时流计算场景。如果您没有 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

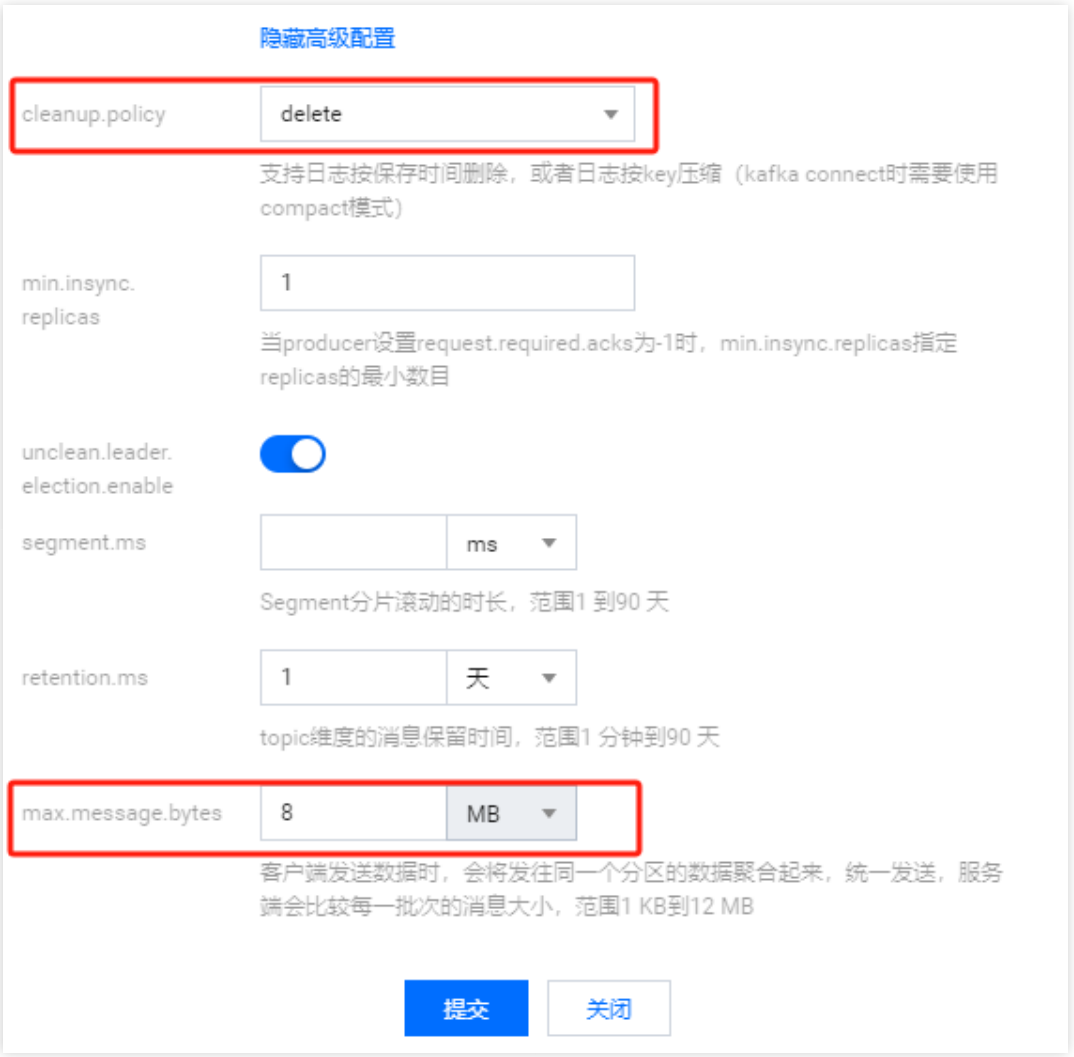
- 已开通消息队列 （CKafka）。

  > 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，其它版本开启 ACL 均会导致投递失败，可通过关闭 ACL 解决投递失败的问题。

- 确保当前操作账号拥有开通投递到 CKafka 的权限。

## 操作步骤

1. 准备和配置Ckafka Topic。在日志主题同地域下，创建一个 CKafka 实例和一个 Topic。
   您需对该Topic的配置做如下修改：
   - cleanup.policy：选择 delete。该参数需设置为 delete，否则会投递失败。
   - max.message.bytes：设置为 ≥ 8MB。当投递 CKafka 的单条 message（一组日志数据）的大小超过1MB时，无法写入 CKafka Topic，导致投递失败。




2. 前往日志服务控制台，在左侧导航栏中，单击投递任务，选择地域、日志集和日志主题。

3. 在投递到CKafka页面，单击右侧的编辑，开启投递到 CKafka 开关。




4. 选择相应的 CKafka 实例以及对应的 Topic，配置。
   您可以选择以原始内容或者 JSON 投递日志。

- 以原始内容投递

  原始内容投递配置项说明如下：

| 配置项 | 解释说明 | 规则 | 是否必填 |
| --- | --- | --- | --- |
| 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 | 支持 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 投递

  JSON 投递配置项说明如下：

| 配置项 | 解释说明 | 规则 | 是否必填 |
| --- | --- | --- | --- |
| 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 |
| JSON 格式的转义/不转义 | - 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。示例：<br>- 日志原文： {"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}}<br>- 投递到 CKafka： {"a":"aa","b":"{\"b1\":\"b1b1\", \"c1\":\"c1c1\"}"}<br>- 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。示例：<br>- 日志原文： {"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}}<br>- 投递到 CKafka： {"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} | 列表选择 | 必填 |
| __TAG__元信息 | 将__TAG__元信息平铺或者不平铺，请按照您的实际业务场景选择。示例：<br>- __TAG__元信息： {"__TAG__":{"fieldA":200, "fieldB":"text"}}<br>- 平铺： {"__TAG__.fieldA":200,"__TAG__.fieldB":"text"}<br>- 不平铺： {"__TAG__":{"fieldA":200, "fieldB":"text"}} | | |
| 数据压缩格式 | 支持 Snappy、LZ4 | 列表选择 | 必填 |
| 投递日志预览 | 预览您投递的日志数据。 | - | - |

5. 单击确定，启动投递到 CKafka 。

# 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

## 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. 在投递到CKafka页面，选择投递任务的地域、日志集、日志主题，即可查看任务的详情和监控信息。
    - 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：Lantency 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。

## 修改投递 CKafka 任务

1. 在投递到CKafka页面中，选择投递任务的地域、日志集、日志主题，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、数据压缩格式等配置项。

## 关闭投递 Kafka 任务

1. 在投递到CKafka页面中，选择投递任务的地域、日志集、日志主题，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。

> 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

## 配置 CLS 投递 Ckafka 任务的告警

1. 登录云监控控制台。
2. 在左侧导航栏中，选择告警配置 > 告警策略。
3. 单击新建，根据实际情况，配置相关信息。
    - 策略名称：自定义，最多30个字符。
    - 策略类型：选择日志服务 / Ckafka投递状态监控。
    - 告警对象：选择您投递 Ckafka 的日志主题。
    - 触发条件：按需设置触发告警的判断条件。
4. 单击完成。

## 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. 在告警策略页面，单击新建，配置相关信息。
    - 策略名称：自定义，最多30个字符。
    - 策略类型：选择消息服务Ckafka / 实例。
    - 告警对象：选择您的 Ckafka 实例。
    - 触发条件：设置磁盘使用百分比相关的告警指标。
2. 单击完成。

# Kafka 协议消费
# Kafka 协议消费

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。
本文提供了开源组件 Flink、Flume、Logstash消费日志主题的Demo。

## 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。

## 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. 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。
4. 在日志主题管理页面中，单击Kafka 协议消费页签。
5. 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，单击确定。
6. 控制台给出 Topic、Host+Port 的信息。用户可以复制信息，构造消费者。

## 消费者参数说明

Kafka 协议消费者的参数说明如下：

| 参数 | 说明 |
| --- | --- |
| 用户认证方式 | 目前仅支持 SASL_PLAINTEXT。 |
| topic | CLS kafka 协议消费控制台给出的主题名称，单击旁边按钮可以复制。例如XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX。 |
| username | 配置为${LogSetID}，即日志集 ID。 例如：0f8e4b82-8adb-47b1-XXXX-XXXXXXXXXXXX ，在日志主题列表中可以复制日志集 ID。 |
| password | 配置为 ${SecretId}#${SecretKey} 。例如：XXXXXXXXXXXXXXX#YYYYYYYY，请登录访问管理控制台，在左侧导航栏中单击云API密钥，API 密钥或者项目密钥均可使用，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 |

| 参数 | 说明 |
| --- | --- |
| | 配置为最小范围，可以满足操作即可。 |

> 注意：
>
> 下面的例子中的代码，jaas.config 的配置， ${SecretId}#${SecretKey} 后有(;分号)，不要漏填，否则会报错。

# Python SDK

```
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 = uuid.uuid4().hex,
auto_offset_reset='earliest',
#服务地址+端口，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 = ['tcloud-cls-kafka-access-consumer.{{regionName}}.{{domainMain}}: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 = "${logsetID}",
#密码是用户的SecretId#SecretKey组合的字符串，例如AKIDWrwkHYYHjvqhz1mHVS8YhXXXX#XXXXuXtymIXT0Lac注意不要丢失#。建议使用子账号密钥,主账号为子账号授权时,遵循最小权限原则,即子账号的访问策略中的action、resource都配置为最小范围,可以满足操作即可.
sasl_plain_password = "${SecretId}#${SecretKey}",
api_version = (1,1,1)
)
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')
```

# Logstash 消费 CLS 日志

```
input {
    kafka {
```

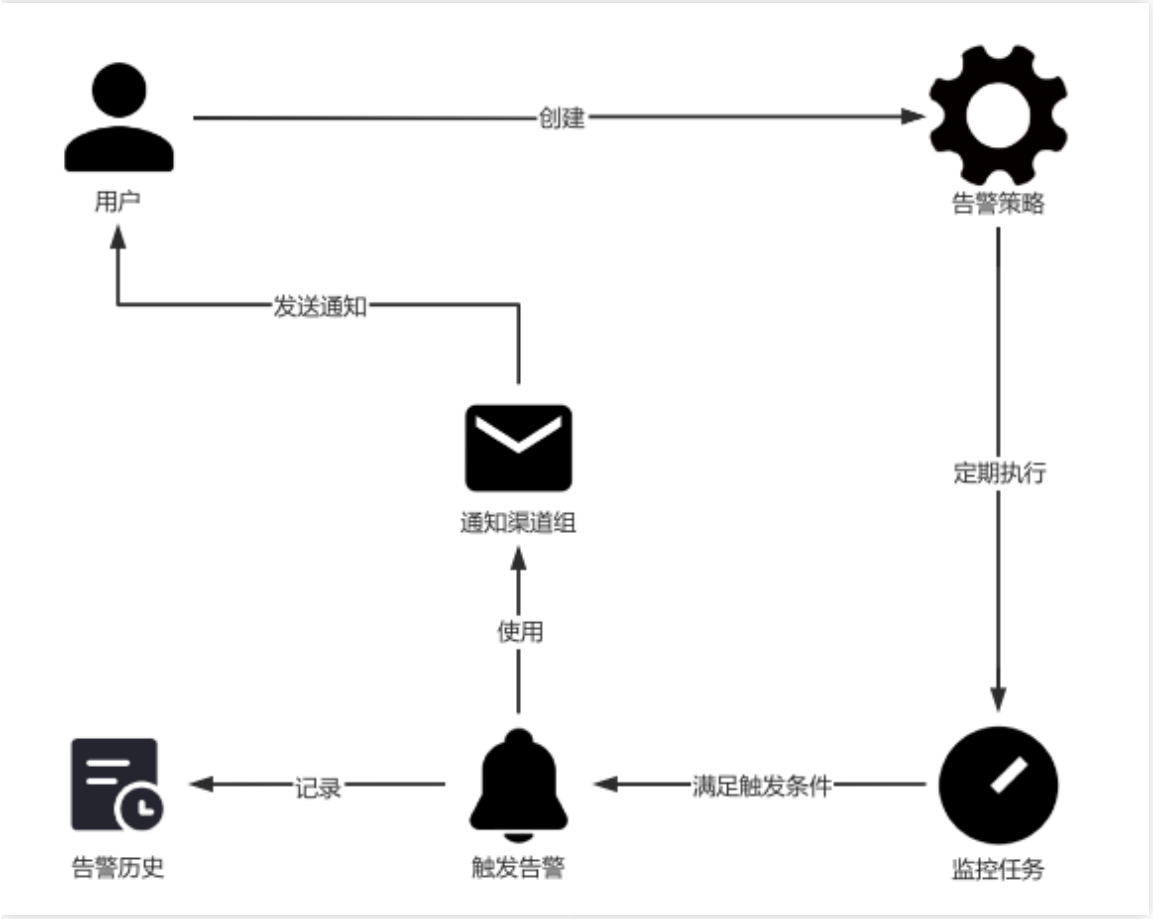
```
        #cls kafka协议消费控制台给出的的主题名称，例如XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX,可在控制台复制
        topics => "您的消费主题"
        #服务地址+端口，内网端口9095,例子是内网消费，请根据您的实际情况填写
        bootstrap_servers => "tcloud-cls-kafka-access-consumer.{{regionName}}.{{domainMain}}:9095"
        group_id => "您的消费组id"
        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AKIDWrwkHYYHjvqhz1mHVS8YhXXXX#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='${logsetID}' password='${securityID}#${securityKEY}';"
    }

}
output {
  stdout { codec => json }
}
```

# 监控告警

# 监控告警简介

日志服务支持对日志主题设置告警策略，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

## 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。
- 告警级别：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。
- 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）。

### 多维分析

触发告警时可针对原始日志进行额外的检索分析，将结果附加在告警通知中，以辅助定位告警原因。例如错误日志条数触发告警时，同时查询具体的错误日志，查询告警通知时即可快速了解具体的错误原因。

### 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：支持邮件、短信、微信、电话、企业微信、自定义接口回调等告警通知方式。

## 产品优势

- 告警策略执行语句 语法规则 与检索分析一致，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。
- 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。

## 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。

# 管理告警策略
# 配置告警策略

## 操作场景

基于日志配置告警策略，在出现符合特定条件的日志时发送告警，例如 error 日志数量过多，接口响应时间过长等。

## 前提条件

- 日志已上传到某个日志主题，且已 配置索引。
- 告警策略需使用 SQL 语句，建议对日志进行结构化处理。

## 操作步骤

登录日志服务控制台，并进入监控告警 > 告警策略页面，单击新建，配置如下主要信息。

### 1. 基本信息

- 告警名称：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主题：

- 所有执行语句使用相同的日志主题：支持多选，最多可选择20个日志主题，不支持跨地域。每个日志主题单独判断是否满足触发条件，同时有多个日志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日志主题：可为每个执行语句单独设置日志主题，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：作用于日志主题的执行语句，执行语句需包含分析语句（即 SQL 语句）。
  - 例1：统计出现 error 的日志条数  status:error | select count(*) as ErrCount
  - 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：执行语句需要查询的数据时间范围，最大时间范围为最近24小时。
- 触发条件：满足触发条件时进行告警。触发条件最多可配置10个，且触发条件之间为或的关系，每个触发条件均会单独进行告警。
  - 告警级别：支持提醒、警告和紧急三种级别，默认为警告。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。
  - 条件表达式：使用 $N.keyname 引用执行语句结果，其中 $N 表示当前告警策略中的第 N 个执行语句， keyname 表示对应的字段名称。表达式语法参见 触发条件表达式语法。
    - 例1：出现 error 的日志条数大于10即触发告警， $1.ErrCount > 10 ，其中 $1 表示第一个执行语句， ErrCount 表示统计结果中的 ErrCount 字段。
    - 例2：域名 domain:aaa.com 的平均响应时间大于5s即触发告警， $2.Latency > 5 ，其中 $2 表示第二个执行语句， Latency 表示统计结果中的 Latency 字段。
- 分组触发：触发条件表达式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。
  - 示例：
    - 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：
      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：
  - 固定频率：按固定的时间间隔执行一次监控任务时间间隔：1 - 1440分钟，粒度：分钟级
  - 固定时间：按固定的时间点执行一次监控任务时间点范围：00:00 - 23:59，粒度：分钟级

## 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。<br>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。<br>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。<br>例1：通过 * \| select avg(timeCost) as time,URL group by URL 获取每个接口请求耗时。<br>例2：通过 status:>499 获取错误日志。 |

> 说明
>
> “相关原始日志”及“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），表示需要对符合哪些条件的原始日志进行多维分析。

## 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、自定义回调接口（webhook）等通知方式。详情参见 管理通知渠道组。
- 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。通知内容最长为200个英文字母或66个中文汉字，超出长度后将会截断。自定义接口回调中使用 {{.Message}} 变量引用该字段时，无长度限制。
- 自定义接口回调配置：
  如果选中的通知渠道组里包含自定义接口回调，则会显示自定义回调接口配置输入框。在输入框中可自定义请求头（Request Header）和请求内容（Request Body），触发告警时 CLS 将使用该请求头和请求内容调用指定的接口。请求头及请求内容中可使用 告警通知变量，将触发告警时的一些数据发送至指定接口。

# 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，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...，两者并不一致。
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（日志上报一般会有10秒左右延迟，严格按实际时间执行告警时，很可能因为最新的日志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日志，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 |

# 触发条件表达式

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.keyname | 引用执行语句的结果， N 为对应的执行语句编号， keyname 为预览结果中的字段名<br>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，请使用 AS 语法 对分析结果设置别名 | $1.ErrCount |
| + | 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<br>$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 &&<br>$1.level=="Error" |
| \|\| | 逻辑运算符：或 | $1.ErrCount>100 \|\|<br>$1.level=="Error" |

- 需使用比较运算符对变量进行比较，且结果为“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 ：结果计算为100，不为“真”，不触发告警
  - 表达式 $1.a+$1.b>=100 ：结果计算为“真”，触发告警
- $N.keyname 中的 keyname 为执行语句结果中的字段名（首字符必须是字母，可以包含字母、数字、下划线）。例如：
  - 执行语句1为： level:error | select count(*) AS ErrCount
  - 执行语句结果中的 ErrCount 可作为触发条件表达式中的 keyname ，触发条件可写为 $1.ErrCount>10
- 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。
- 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：当近5分钟出现 error 级别日志时触发告警

执行语句：

```
level:error | select count(*) AS ErrCount
```

查询时间范围：近5分钟
触发条件：

```
$1.ErrCount>0
```

示例2：当近5分钟 5xx 状态码出现10次时触发告警

执行语句：

```
status:500 | select count(*) AS Count
```

查询时间范围：近5分钟
触发条件：

```
$1.Count>10
```

示例3: 当某 VIP 用户（uid:10001）的请求延时大于10s时触发告警

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：

```
$1.latency>10
```

# 管理通知渠道组

## 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

### 新建通知渠道组

1. 登录日志服务控制台。
2. 在左侧导航树中，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，填写通知相关信息后，单击确定。
   通知渠道支持如下三种：

   - 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。
     通知时段：可接收告警通知的时间段。
     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 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。
     >   - 以上限制不局限于日志服务产生的电话告警通知，云监控及消息中心产生的电话通知会共享该限制。

   - 企业微信
     回调地址：企业微信机器人的回调地址，获取方式详见企业微信帮助文档：如何设置群机器人。
     通知时段：可接收告警通知的时间段。

> 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

  回调地址：支持 HTTP 及 HTTPS 地址。

  请求方法：支持 POST 及 PUT。

  通知时段：可接收告警通知的时间段。

  > 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。
  > - 自定义回调支持需支持通过公网进行访问，日志服务无法回调内网地址。
  > - 自定义回调没有固定的请求信息，需在告警策略中按需进行配置，包括请求头（Request Header）及请求内容（Request Body），操作说明详见 配置告警策略。

## 复制通知渠道组

1. 在通知渠道组管理页面，单击待复制渠道组右侧的复制。
2. 查看并修改相应信息后，单击确定。

## 修改通知渠道组

1. 在通知渠道组管理页面，单击待修改渠道组右侧的编辑。
2. 修改相应信息，单击确定。

## 删除通知渠道组

1. 在通知渠道组管理页面，单击待删除渠道组右侧的删除。
2. 在弹出对话框中，单击确定。

# 告警通知变量

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。

## 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

相关通知变量的说明如下表所示：

| 变量 | 含义 | 变量值示例 | 说明 |
| --- | --- | --- | --- |
| {{.UIN}} | 账号ID | 100007xxx827 | - |
| {{.Nickname}} | 账号昵称 | xx企业 | - |
| {{.Region}} | 地域 | city1 | - |
| {{.Alarm}} | 告警策略名称 | Nginx 错误日志过多 | - |
| {{.AlarmID}} | 告警策略ID | notice-3abd7ad6-15b7-4168-xxxx-52e5b961a561 | - |
| {{.ExecuteQuery}} | 执行语句 | ["* \| select approx_percentile(request_time, 0.99) as p99"] | 数组结构，{{.ExecuteQuery[0]}}代表第1个执行语句，{{.ExecuteQuery[1]}}代表第2个执行语句，以此类推 |
| {{.Condition}} | 触发条件 | $1.errorLogCount > 1 | - |
| {{.HappenThreshold}} | 告警所需的触发条件持续满足次数 | 1 | - |
| {{.AlertThreshold}} | 告警间隔时间 | 15 | 单位：分钟 |
| {{.Topic}} | 日志主题名称 | nginxLog | - |
| {{.TopicId}} | 日志主题ID | a54de372-ffe0-49ae-xxxx-c340bb2b03f2 | - |

| 变量 | 含义 | 变量值示例 | 说明 |
| --- | --- | --- | --- |
| {{.Level}} | 告警级别 | Info | - 紧急：Critical<br>- 警告：Warn<br>- 提醒：Info |
| {{.Label}} | 附加分类字段 | {"service":"payService"} | |
| {{.StartTime}} | 第一次告警触发时间 | 2022-05-28 18:56:37 | 时区：Asia/Shanghai |
| {{.StartTimeUnix}} | 第一次告警触发时间戳 | 1653735397099 | 毫秒级 UNIX 时间戳 |
| {{.NotifyTime}} | 本次告警通知时间 | 2022-05-28 19:41:37 | 时区：Asia/Shanghai |
| {{.NotifyTimeUnix}} | 本次告警通知时间戳 | 1653738097099 | 毫秒级 UNIX 时间戳 |
| {{.NotifyType}} | 告警通知类型 | 1 | 1代表告警通知，2代表恢复通知 |
| {{.ConsecutiveAlertNums}} | 连续告警次数 | 2 | - |
| {{.Duration}} | 告警持续时间 | 0 | 单位：分钟 |
| {{.TriggerParams}} | 告警触发时参 | $1.errorLogCount=5; | - |

| 变量 | 含义 | 变量值示例 | 说明 |
| --- | --- | --- | --- |
| | 数 | | |
| {{.ConditionGroup}} | 告警分组触发时对应的分组信息 | {"$1.AppName":"userManageService"} | 告警策略启用分组触发功能时才具备该变量 |
| {{.RecordGroupId}} | 告警分组ID | 52746a79-7681-495f-xxxx-2715593e625a | 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；<br>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 |
| {{.Message}} | 通知内容 | - | 特指告警策略配置中填写的“通知内容” |
| {{.QueryResult}} | 执行语句执行结果 | 详见 {{.QueryResult}} 说明 | - |
| {{.QueryLog}} | 执行语句中检索条件匹配到的详细日志 | 详见 {{.QueryLog}} 说明 | - |
| {{.AnalysisResult}} | 多维分析结果 | 详见 {{.AnalysisResult}} 说明 | 仅告警触发时具备该变量，告警恢复时无该变量 |

相关说明如下：
{{.QueryResult}}

- 含义：执行语句执行结果。

- 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。

- 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"
  }]
]
```

{{.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。每个执行语句最多包含最近的10条详细日志。

- 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 (K
HTML, 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。

- 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": 0.017647058823529412,
      "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 (K
HTML, 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时，请使用 {{index .variable "childNodeName"}} 形式的语法，例如 {{index .AnalysisResult "Top URL"}} 。

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。

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

### 转义特殊字符

语法格式

```
{{escape .variable}}
```

语法说明

告警变量中很多变量包含特殊符号，如果直接将变量拼接在 JSON 字符串中进行自定义接口回调，可能会导致 JSON 格式错误，导致回调失败。此时可对原始变量值进行转义，再拼接至 JSON 字符串中。

使用示例

{{.ExecuteQuery[0]}} 的变量值为 status:>=400 | select count(*) as "错误日志数"

如果不使用转义，自定义接口回调配置中请求内容为：

```
{
  "Query":"{{.ExecuteQuery[0]}}"
}
```

返回结果如下（不是一个合法的 JSON）：

```
{
   "Query":"status:>=400 | select count(*) as "错误日志数""
}
```

此时可使用转义，将自定义接口回调配置中请求内容修改为：

```
{
  "Query":"{{escape .ExecuteQuery[0]}}"
}
```

返回结果如下（符合 JSON 语法要求）：

```
{
  "Query":"status:>=400 | select count(*) as \"错误日志数\""
}
```

## 字符串截取

### 按字符串长度截取

语法格式

```
{{substr .variable start}} 或 {{substr .variable start length}}
```

语法说明

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。

使用示例

{{.QueryLog[0][0].fileName}} 变量值为：

```
/root/testLog/nginx.log
```

通过以下表达式获取第6个字节开始、总长7个字节的字符串：

```
{{substr .QueryLog[0][0].fileName 6 7 }}
```

返回结果：

```
testLog
```

按首尾字符串截取

语法格式

```
{{extract .variable "startStringRegex" ["endStringRegex"]}}
```

语法说明

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。

使用示例

{{.QueryLog[0][0].fileName}} 变量值为：

```
/root/testLog/nginx.log
```

通过以下表达式获取 /root/ 和 /nginx 之间的字符串：

```
{{extract .QueryLog[0][0].fileName "/root/" "/nginx"}}
```

返回结果：

```
testLog
```

{{.Condition}} 变量值为：

```
$1.errorLogCount > 1
```

通过以下表达式获取 $1. 之后的字符串（因为 $ 及 . 是正则表达式中的特殊符号，所以需要转义）：

```
{{extract .Condition "\\$1\\."}}
```

返回结果：

```
errorLogCount > 1
```

## 是否包含指定字符串

语法格式

```
{{containstr .variable "searchstring"}}
```

语法说明
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。

使用示例
{{.QueryLog[0][0].fileName}} 变量值为：

```
/root/testLog/nginx.log
```

通过以下表达式获取 /root/ 和 /nginx 之间的字符串：

```
{{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
```

## UNIX 时间戳转换

语法格式

```
{{fromUnixTime .variable}} 或 {{fromUnixTime .variable "timezone"}}
```

语法说明
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，客户需要修改为当前时区。

使用示例
{{.QueryLog[0][0].time}} 变量值为：

```
1653893435008
```

通过以下表达式分别获取不同时区下的日期时间：

```
{{fromUnixTime .QueryLog[0][0].time}}
{{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
```

## 字符串拼接

语法格式

```
{{concat .variable1 .variable2 ...}}
```

语法说明
将指定的多个变量或字符串拼接在一起。

使用示例
将地域及告警策略名称拼接在一起：

```
{{concat .Region .Alarm}}
```

返回结果：

```
city1 alarmTest
```

## base64/base64url/url 编码及解码

语法格式

```
{{base64_encode .variable}}
{{base64_decode .variable}}
{{base64url_encode .variable}}
{{base64url_decode .variable}}
{{url_encode .variable}}
{{url_decode .variable}}
```

语法说明
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"="。

使用示例

```
{{base64_encode "test测试"}}
{{base64_decode "dGVzdOa1i+ivlQ=="}}
{{base64url_encode "test测试"}}
{{base64url_decode "dGVzdOa1i-ivlQ=="}}
```

```
{{url_encode "https://console.abc.com:80/ctetenls?region=ap-city"}}
{{url_decode "https%3A%2F%2Fconsole.abc.com%3A80%2Fcls%3Fregion%3Dap-city"}}
```

返回结果：

```
dGVzdOa1i+ivlQ==
test测试
dGVzdOa1i-ivlQ==
test测试
https%3A%2F%2Fconsole.abc.com%3A80%2Fcls%3Fregion%3Dap-city
https://console.abc.com:80/cls?region=ap-city
```

## md5/sha1/sha256/sha512 加密

语法格式

```
{{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}}
{{sha512 .variable | base64_encode}}
{{sha512 .variable | base64url_encode}}
```

语法说明

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。

使用示例

```
{{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}}
```

返回结果：

```
098F6BCD4621D373CADE4E832627B4F6
CY9rzUYh03PK3k6DJie09g==
CY9rzUYh03PK3k6DJie09g==
A94A8FE5CCB19BA61C4C0873D391E987982FBBD3
qUqP5cyxm6YcTAhz05Hph5gvu9M=
qUqP5cyxm6YcTAhz05Hph5gvu9M=
9F86D081884C7D659A2FEAA0C55AD015A3BF4F1B2B0B822CD15D6C15B0F00A08
n4bQgYhMfWWaL+qgxVrQFaO/TxsrC4Is0V1sFbDwCgg=
n4bQgYhMfWWaL-qgxVrQFaO_TxsrC4Is0V1sFbDwCgg=
EE26B0DD4AF7E749AA1A8EE3C10AE9923F618980772E473F8819A5D4940E0DB27AC185F8A0E1D5F84
F88BC887FD67B143732C304CC5FA9AD8E6F57F50028A8FF
7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc/iBml1JQODbJ6wYX4oOHV+E+IvIh/1nsUNzLDBMxfqa2Ob1f1ACi
o/w==
7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc_iBml1JQODbJ6wYX4oOHV-E-IvIh_1nsUNzLDBMxfqa2Ob1f1ACio
_w==
```

## hmac_md5/hmac_sha1/hmac_sha256/hmac_sha512 加密

语法格式

```
{{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}}
```

语法说明

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。

使用示例

```
{{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=
FC49EBC05209B1359773D87C216BA85BCE0163FDE459EA37AB603EC9D8445D23
/EnrwFIJsTWXc9h8IWuoW84BY/3kWeo3q2A+ydhEXSM=
_EnrwFIJsTWXc9h8IWuoW84BY_3kWeo3q2A-ydhEXSM=
D18DF3D943F74769A8B66E43D7EF03639BB6B8B8A2EBC9976170DC58EEE58BE98478F3183E4B5AA34
81DE12026AAE3843F8213B39D639EAC6EE93734EA667BC5
0Y3z2UP3R2motm5D1+8DY5u2uLii68mXYXDcWO7li+mEePMYPktao0gd4SAmquOEP4ITs51jnqxu6Tc06
mZ7xQ==
0Y3z2UP3R2motm5D1-8DY5u2uLii68mXYXDcWO7li-mEePMYPktao0gd4SAmquOEP4ITs51jnqxu6Tc06
mZ7xQ==
```

# 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) Chro
me/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/3
5.0.3319.102 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3：在多维分析中使用执行语句执行结果作为过滤条件

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].status_code}} 代表第一条执行语句执行结果中第一行第二列的值，既可以类似上图使用在SQL语句中，也可以使用在检索条件中，例如：

status_code:{{.QueryResult[0][0].status_code}} | select url,status_code limit 5

> 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。SQL中需使用单引号包裹，例如 '{{.QueryResult[0][0].url}}' ，检索条件中使用双引号包裹，例如 "{{.QueryResult[0][0].url}}" 。

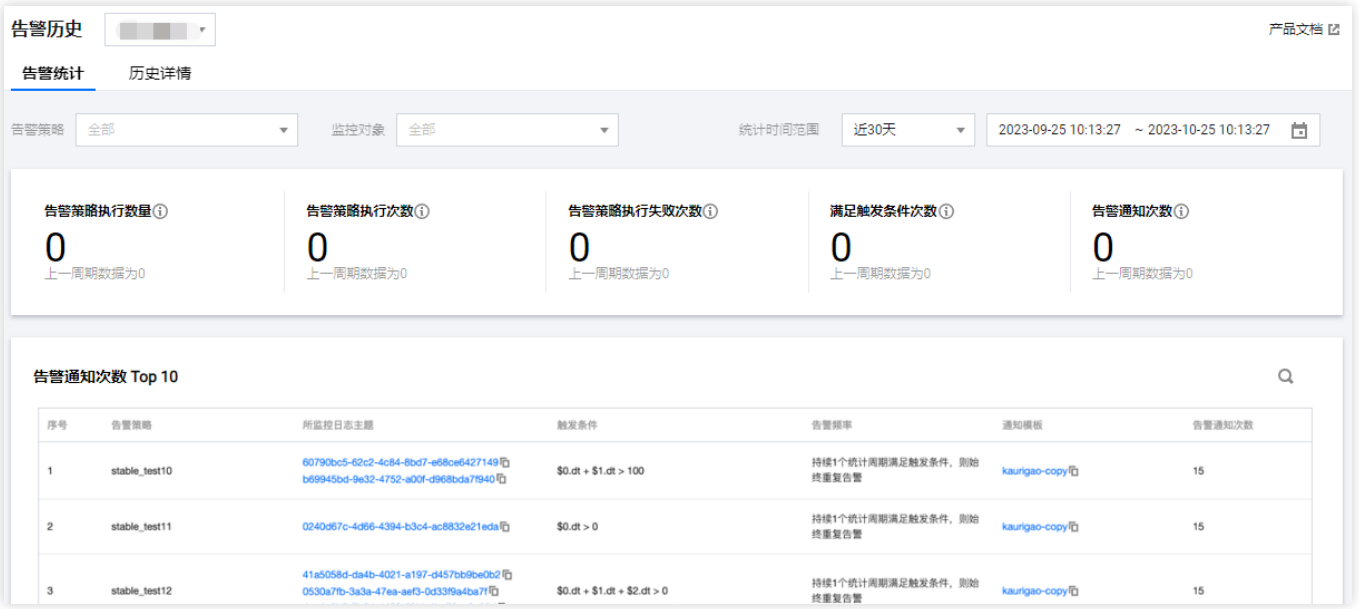
# 查看告警历史

## 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，选择监控告警 > 告警历史，进入告警统计页面。
   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策略、监控对象和时间范围。




### 告警统计

【告警统计】展示了当前地域下的告警重要信息，包括告警策略统计、监控任务执行情况等，统计指标详细如下：

| 统计指标 | 说明 |
| --- | --- |
| 告警策略执行数量 | 统计时间范围内，执行过的告警策略个数 |
| 告警策略执行次数 | 统计时间范围内，运行告警策略中的查询分析语句的次数 |
| 告警策略执行失败次数 | 统计时间范围内，告警策略执行结果失败的次数，执行失败所包含的错误范围：AlarmConfigNotFound、QuerySyntaxError、QueryError、QueryResultParseError、ConditionSyntaxError、ConditionEvaluateError、ConditionValueTypeError |

| 统计指标 | 说明 |
| --- | --- |
| 满足触发条件次数 | 统计时间范围内，告警策略的查询分析语句运行成功，且返回结果满足触发条件的次数 |
| 告警策略触发通知次数 | 统计时间范围内，告警策略执行后触发通知发送的次数 |
| 告警通知次数 Top 10 | 统计时间范围内，触发通知发送次数最多的10个告警策略 |

## 历史详情

告警策略一旦生效，会周期性执行监控任务，每次监控任务执行的情况会记录在【历史详情】中，主要信息包括每次执行的结果情况，通过查看告警策略执行的记录，方便进行告警任务的历史追溯。

> 说明：
>
> 日志服务提供近30天的历史信息查看。

告警状态：

| 告警状态 | 说明 |
| --- | --- |
| 未恢复 | 系统持续满足触发条件达到阈值，触发告警 |
| 已恢复 | 当前监控周期未满足触发条件 |
| 已失效 | 告警策略发生删除或修改 |

策略执行结果：

| 策略执行结果 | 触发条件 | 告警通知 | 通知类型 | 详细说明 |
| --- | --- | --- | --- | --- |
| QueryResultUnmatch | 不满足 | 未发送 | - | 执行语句结果不满足触发条件 |
| TriggerNotice.SendSuccess | 满足 | 发送成功 | 告警通知 | 告警触发,通知发送成功 |
| TriggerNotice.SendFail | 满足 | 发送失败 | 告警通知 | 告警触发,通知发送失败 |
| TriggerNotice.SendPartFail | 满足 | 发送失败 | 告警通知 | 告警触发,部分通知发送失败 |
| TriggerNotice.TypeUnmatched | 满足 | 未发送 | - | 告警触发,但不符合通知渠道组类型要求 |
| TriggerNotice.TimeUnmatched | 满足 | 未发送 | - | 告警触发,但不符合通知渠道组通知时段要求 |
| RecoverNotice.SendSuccess | 不满足 | 发送成功 | 告警恢复 | 告警恢复,通知发送成功 |
| RecoverNotice.SendFail | 不满足 | 发送失败 | 告警恢复 | 告警恢复,通知发送失败 |

| 策略执行结果 | 触发条件 | 告警通知 | 通知类型 | 详细说明 |
| --- | --- | --- | --- | --- |
| RecoverNotice.SendPartFail | 不满足 | 发送失败 | 告警恢复 | 告警恢复,部分通知发送失败 |
| RecoverNotice.TypeUnmatched | 不满足 | 未发送 | - | 告警恢复,但不符合通知渠道组类型要求 |
| RecoverNotice.TimeUnmatched | 不满足 | 未发送 | - | 告警恢复,但不符合通知渠道组通知时段要求 |
| EvalTimesLimited | 不满足 | 未发送 | - | 执行语句结果不满足触发条件（触发条件计算次数超出上限） |
| EvalTimesLimited.SendSuccess | 不满足 | 发送成功 | 告警恢复 | 告警恢复（触发条件计算次数超出上限）,通知发送成功 |
| EvalTimesLimited.SendFail | 不满足 | 发送失败 | 告警恢复 | 告警恢复（触发条件计算次数超出上限）,通知发送失败 |
| EvalTimesLimited.SendPartFail | 不满足 | 发送失败 | 告警恢复 | 告警恢复（触发条件计算次数超出上限）,部分通知发送失败 |
| EvalTimesLimited.TypeUnmatched | 不满足 | 未发送 | - | 告警恢复（触发条件计算次数超出上限）,但不符合通知渠道组类型要求 |
| EvalTimesLimited.TimeUnmatched | 不满足 | 未发送 | - | 告警恢复（触发条件计算次数超出上限）,但不符合通知渠道组通知时段要求 |
| HappenThresholdUnreach | 满足 | 未发送 | - | 满足触发条件,但不满足告警持续周期要求 |
| AlertIntervalThresholdUnreach | 满足 | 未发送 | - | 满足触发条件,但不满足告警频率要求 |
| AlarmConfigNotFound | 执行失败 | 未发送 | - | 告警策略配置异常 |
| QueryError | 执行失败 | 未发送 | - | 执行语句执行错误 |
| ConditionEvaluateError<br>ConditionSyntaxError<br>ConditionValueTypeError | 执行失败 | 未发送 | - | 触发条件表达式执行错误 |
| InternalError | 执行失败 | 未发送 | - | 内部错误 |
| InternalError.QueryResultParseError | 执行失败 | 未发送 | - | 内部错误.执行语句结果解析错误 |

# 常见问题
# 健康监测问题解释
# 日志上传相关

## 如何应对提示参数错误？

该错误说明通过 API 或 SDK 上传日志时，请求输入参数填写错误，请确保参数填写正确。

## 哪些原因可能会导致鉴权失败？

该错误说明上传日志时出现鉴权错误。 该类问题的出现通常可能由以下几种原因导致：

| 错误原因 | 解决方法 |
| --- | --- |
| 密钥不存在 | 请在控制台检查密钥是否已被删除或者禁用。<br>如状态正常，请检查密钥是否填写正确，注意前后不得有空格。 |
| 签名错误 | 签名计算错误，请对照API文档调用方式中的“签名方法”检查签名计算过程。 |
| 签名过期 | 请对照调用方式中的签名方法文档重新计算签名。 |
| 请求未授权 | 无上传日志权限。请前往访问管理控制台，为您的账号添加 CLS 上传日志的权限。 |
| 密钥非法 | 密钥格式不正确。 |

## 如何应对上传日志大小超限？

单条日志上传请求中的日志大小存在超限， 请参考上传日志限制调整日志上传大小。

## 为什么会出现触发流控或频控？

该错误说明存在以下规格超限的情况：

| 限制项 | 说明 |
| --- | --- |
| 写频控 | 单个日志主题分区写请求限制：500QPS。 |
| 写流控 | 单个日志主题分区写流量限制：5MB/s。 |

建议减少日志上传的频次与流量。 若不希望改变日志上传的频次或流量，建议为日志主题 开启自动分裂。

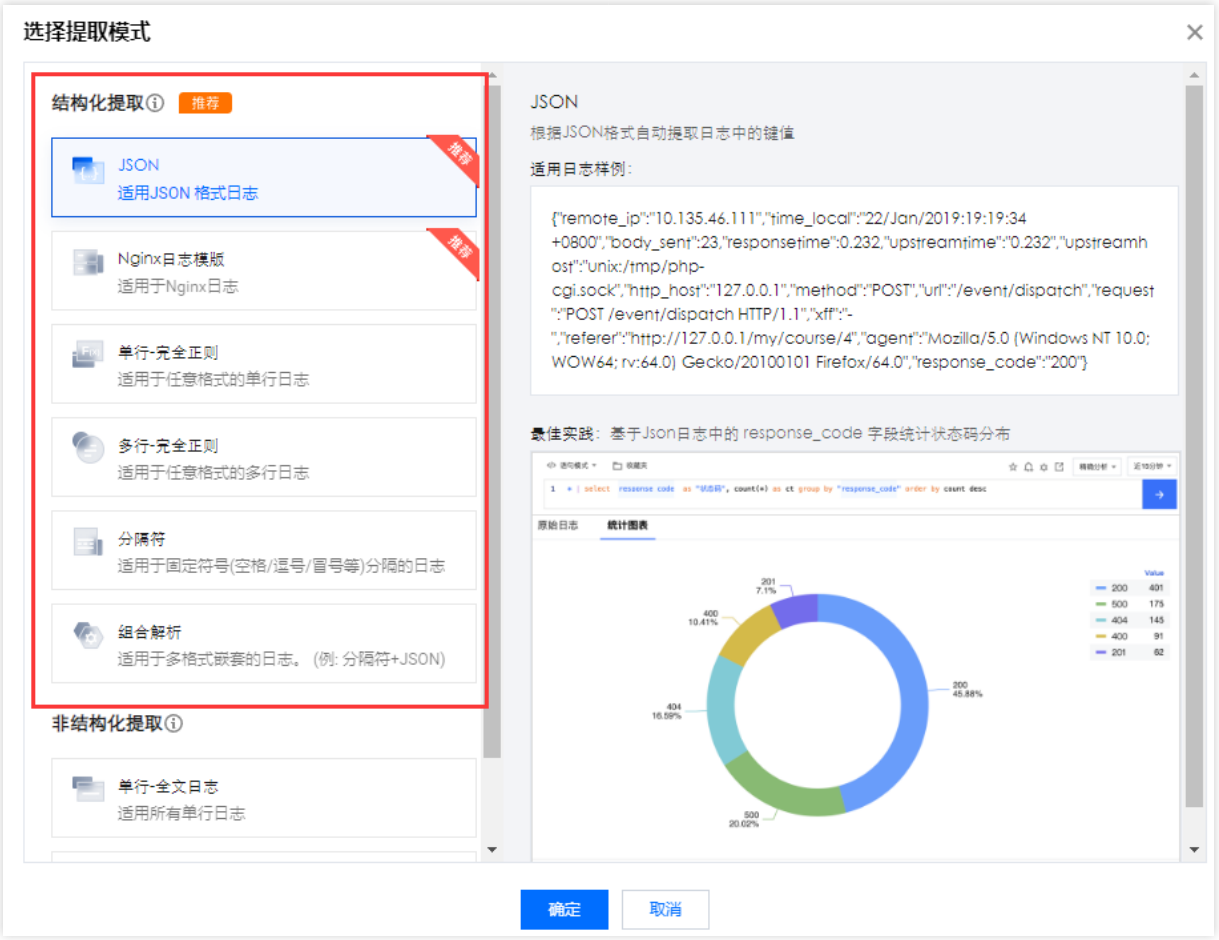
# 采集配置相关

## 非结构化提取模式和结构化提取模式的区别？

非结构化提取模式指的是采集配置中，提取模式配置为单行全文或多行全文。 这类提取模式不会对日志进行格式化解析， 而是全文采集。在这种提取模式下，日志采集上来只能进行最简单的全文检索， 日志条数统计，日志条数告警。
CLS 日志服务支持更强大的基于日志内容中的特定字段进行检索分析，可视化与监控告警。 该能力依赖对所采集日志进行格式化解析，即在采集配置中选择结构化提取模式。若您的日志输出格式统一， 建议配置结构化提取模式， 以最大提升您的日志分析体验。

## 如何配置结构化提取模式？

1. 在左侧导航栏中，单击日志主题，进入日志主题管理页面。
2. 找到目标日志主题， 单击日志主题名称进入日志主题配置页面。
3. 选择采集配置页签，在“LogListener 采集配置”找到提取模式为单行或多行全文的采集配置，单击查看进入采集配置编辑页面。
4. 找到提取模式配置项， 根据您的日志格式， 单击修改并选择相应的结构化提取模式。




5. 结构化提取模式

| 提取模式 | 说明 |
| --- | --- |
| 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， |

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

| 提取模式 | 说明 |
| --- | --- |
| | 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。 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。假设一条日志原始数据为： |
| [2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened<br>at TestPrintStackTrace.f(TestPrintStackTrace.java:3) | |

| 提取模式 | 说明 |
| --- | --- |
| at TestPrintStackTrace.g(TestPrintStackTrace.java:7)<br>at TestPrintStackTrace.main(TestPrintStackTrace.java:16)<br>配置的自定义正则表达式为： | |
| \[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*<br>行首正则表达式为： | |
| \[(\d+-\d+-\w+:\d+:\d+,\d+)\]\s\[(\w+)\]\s(.*)<br>根据提取的 key，采集到日志服务的数据为： | |
| time： 2018-10-01T10:30:01,000``<br> level： INFO``<br>msg：java.lang.Exception: exception happened<br>at TestPrintStackTrace.f(TestPrintStackTrace.java:3)<br>at TestPrintStackTrace.g(TestPrintStackTrace.java:7)<br>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 | |

| 提取模式 | 说明 |
| --- | --- |
| +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"}<br>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<br>body_sent: 23<br>http_host: 127.0.0.1<br>method: POST<br>referer: http://127.0.0.1/my/course/4<br>remote_ip: 10.135.46.111<br>request: POST /event/dispatch HTTP/1.1<br>response_code: 200<br>responsetime: 0.232<br>time_local: 22/Jan/2019:19:19:34 +0800<br>upstreamhost: unix:/tmp/php-cgi.sock<br>upstreamtime: 0.232<br>url: /event/dispatch<br>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。 |

| 提取模式 | 说明 |
| --- | --- |
| | 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/<br>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 -<br>bytes: 35<br>host: 127.0.0.1<br>length: 647<br>referer: http://127.0.0.1/<br>request: GET /online/sample HTTP/1.1<br>status: 200<br>time: [Tue Jan 22 14:49:45 CST 2019 +0800] | |

# 索引配置相关

## 如何开启索引配置？

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，未开启索引将无法对日志进行检索分析。 强烈建议开启。

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. 单击需要配置索引的日志主题名称，进入日志主题管理页面。
4. 选择索引配置页签， 单击编辑，进入编辑索引配置页面。
5. 单击索引状态开关， 开启索引。
   开启索引后， 您可进一步配置全文索引或键值。
   - 全文索引：支持对日志进行全文检索。
   - 键值索引：支持基于日志内容中的指定字段进行检索。该配置的前提是日志采集配置中的提取模式为结构化提取模式（即将日志解析为键值对）。

## 如何开启键值索引配置？

键值索引指的是在全文索引的基础上，进一步将原始日志按字段（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）。强烈建议开启，以最大化提升日志检索的效率。

> 注意：
>
> 开启键值索引的前提条件是日志采集配置中的提取模式为结构化提取模式（即将日志解析为键值对）。

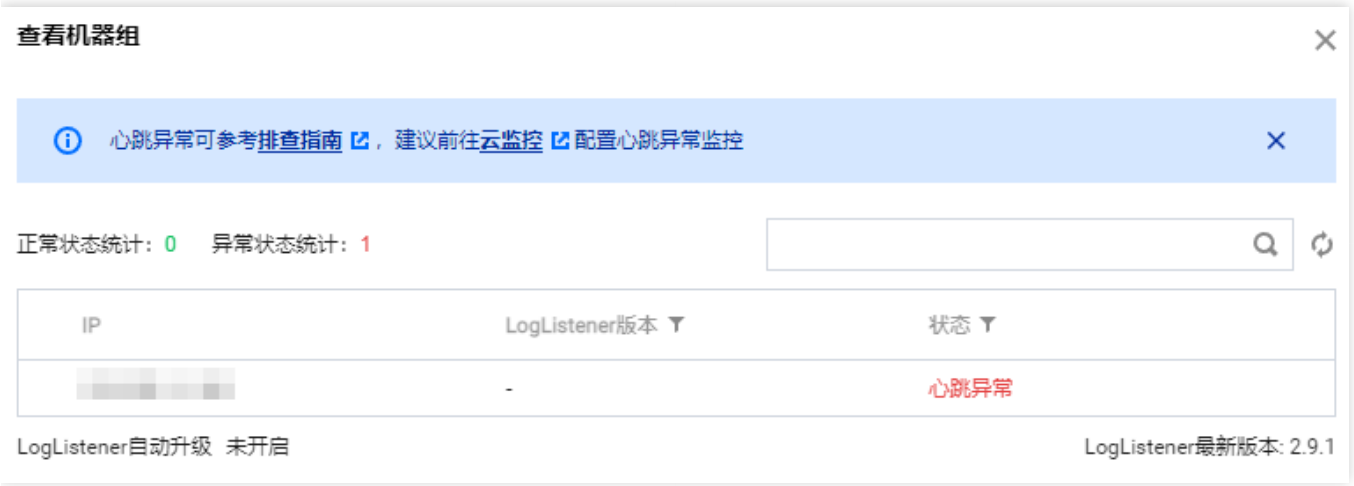
操作步骤

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. 单击需要配置索引的日志主题 ID/名称，进入日志主题管理页面。
4. 选择索引配置页签， 单击编辑，进入编辑索引配置页面。
5. 单击键值索引开关， 开启键值索引。
   开启键值索引后， 您还可以单击自动配置，系统将自动获取采集到的最近1条日志作为样例，并将其中的字段解析为键值索引。您可以在自动配置的基础上进行微调，快速获取最终的索引配置信息。

# 采集相关
# 机器组状态异常问题

## 概述

完成机器组配置后，在机器组管理页中，点击机器组名称可以查看与该机器组关联的机器心跳。状态正常说明机器心跳正常上报，若不为正常（如下图），则说明该机器上的 LogListener 存在异常。




LogListener 可能存在以下异常状态：

- 心跳异常
- 目录超限
- 文件超限
- CPU 超限
- 内存超限
- 鉴权失败

  本文将针对以上异常状态进行解释，并提供相应的解决方案。

## 心跳异常

当状态为心跳异常时，说明 LogListener 与日志服务后端连接中断，会导致 LogListener 无法正常上传日志。您可通过以下方式进一步排查连接中断的原因：

> 说明
>
> 仅适合于 LogListener 2.2.4 及以上版本。

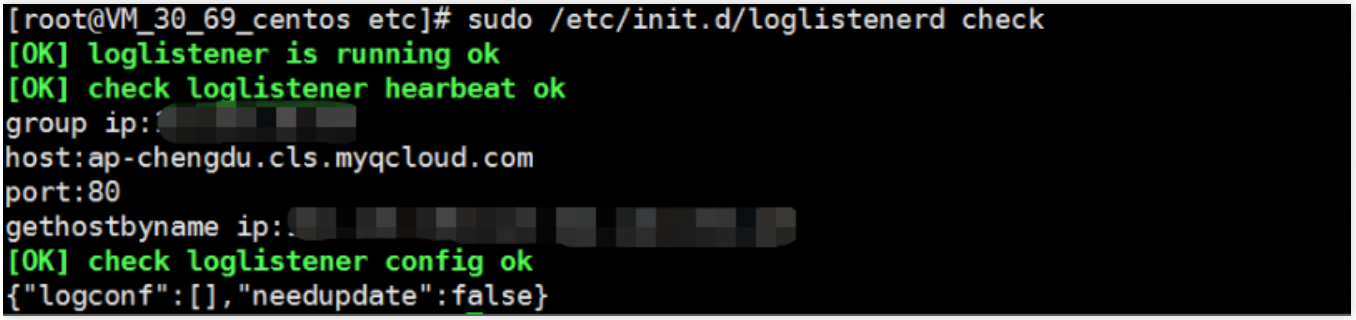
1. 使用 LogListener 快速诊断工具

LogListener 快速诊断工具可以快速诊断 LogListener 是否启动、心跳是否正常、配置拉取是否正常。

在命令行下执行如下指令：

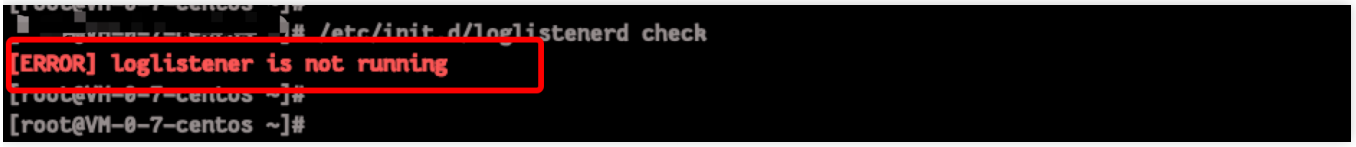
```
/etc/init.d/loglistenerd check
```

若 LogListener 运行正常，诊断工具返回的结果如图所示：



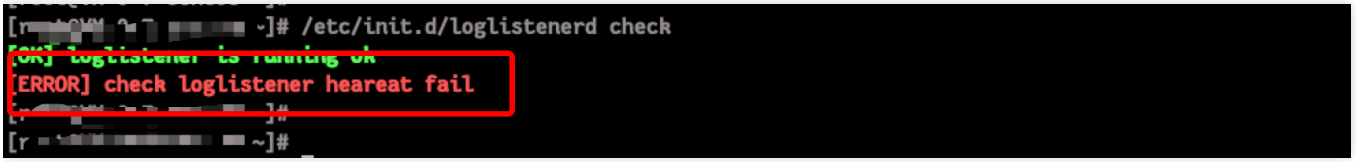

LogListener 进程异常

如果出现如下图所示 “[ERROR] loglistener is not running” 字样，表示 LogListener 没有启动。 执行 /etc/init.d/loglistenerd start 启动，更多操作指令参见 LogListener 常用操作指令。




LogListener 心跳异常

如果出现如下图所示 “[ERROR] check loglistener heareat fail” 字样，表示 LogListener 心跳异常。




引起 LogListener 心跳异常的原因有很多，最常见的情况有：

- 网络异常
  执行以下命令检查网络环境是否连通，命令中的 “cls domain name” 为 CLS 服务域名。

  ```
  telnet <cls domain name> 80
  ```

- 密钥信息错误
  检查 LogListener 密钥信息是否正确，进入到 LogListener 安装目录执行如下命令。

  > 说明：
  >
  > 若无特殊指定，LogListener 安装目录通常为 /usr/local/loglistener 。

grep secret etc/loglistener.conf




2. 检查机器组 IP 配置

检查机器组所添加的 IP 地址是否为 LogListener 安装过程中获取的 IP 地址。进入到 LogListener 安装目录执行如下命令检查 LogListener 配置的 IP 地址：

> 说明：
>
> 若无特殊指定，LogListener 安装目录通常为 /usr/local/loglistener 。

grep group_ip etc/loglistener.conf



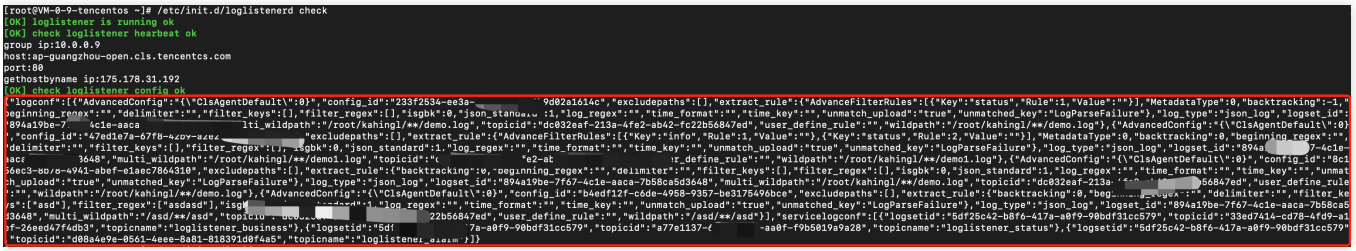

登录日志服务控制台，单击机器组管理，查看机器组配置的 IP 地址，机器组所配置的 IP 地址必须与 LogListener 获取的 IP 地址完全一致。

# 目录超限

当状态为目录超限时，说明 LogListener 监听的采集目录数量已超过规格限制5000，会导致 LogListener 部分超出部分的目录的日志无法被采集。您可通过以下方式进行修复：

1. 执行以下命令查看当前 LogListener 关联的采集配置。

   /etc/init.d/loglistenerd check



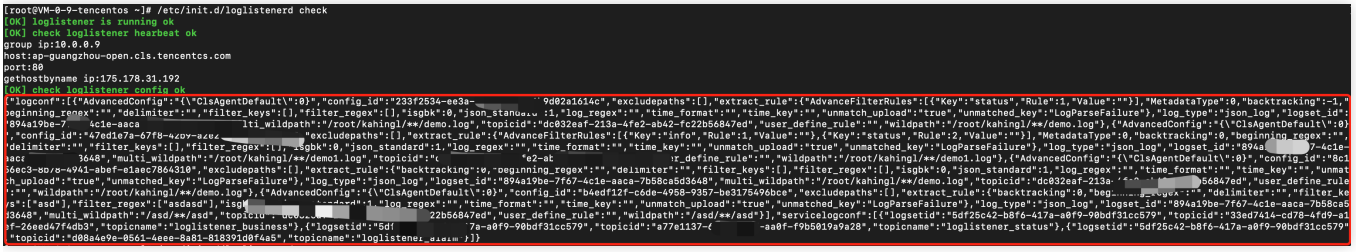

2. 以上返回结果中，logconfig 数组包含了当前 LogListener 的所有采集配置，您可针对每个采集配置，检查 wildpath 是否可能模糊命中大量无需采集的目录，例如 /root/**/error.log 这类跟目录的模糊路径。
3. 找到目标可优化的采集配置后， 从步骤1的返回数据中查看该采集配置关联的 topicid，并基于该 topicid 前往 CLS 控制台找到目标日志主题，优化或删除该采集配置。

# 文件超限

当状态为文件超限时，说明 LogListener 监听的采集文件数量已超过规格限制10000，会导致 LogListener 部分超出部分的目录的日志无法被采集。您可通过以下方式进行修复：

1. 执行以下命令查看当前 LogListener 关联的采集配置。

   /etc/init.d/loglistenerd check




2. 以上返回结果中，logconfig 数组包含了当前 LogListener 的所有采集配置，您可针对每个采集配置，检查 wildpath 是否可能模糊命中大量无需采集的文件，例如 /root/**/*.log 这类从根目录模糊匹配任意名称的日志文件。
3. 找到目标可优化的采集配置后，从步骤1的返回数据中查看该采集配置关联的 topicid，并基于该 topicid 前往 CLS 控制台找到目标日志主题，优化或删除该采集配置。

# CPU 超限

当状态为 CPU 超限时，说明 LogListener 所占用 CPU 已超过预先设置的阈值。您可通过以下方式进行修复：

1. 以 LogListener 安装目录为 /usr/local/loglistenr 为例，执行以下命令打开 LogListener 配置文件：

vim etc/loglistener.conf

2. 在 LogListener 配置文件中， 找到 cpu_usage_thres 并按需提升 CPU 利用率限制， 如下图：

```
max_send_rate = 0
# Max cpu usage limit with single core, set to 50 means 50% usage limit, 0 for unlimited.
cpu_usage_thres = 20
```

3. 完成配置后，保存并重启 LogListener。

# 内存超限

当状态为内存超限时，说明 LogListener 所占用内存已超过预先设置的阈值。您可通过以下方式进行修复：

1. 以 LogListener 安装目录为 /usr/local/loglistenr 为例，执行以下命令打开 LogListener 配置文件：

vim etc/loglistener.conf

2. 在 LogListener 配置文件中，找到 max_mem 并按需提升内存占用限制，如下图：

```
max_connection = 10
# Max memory loglistener would use.
max_mem = 2097152000
```

3. 完成配置后，保存并重启 LogListener。

# 鉴权失败

当状态为鉴权失败时，说明 LogListener 中配置的密钥错误或已失效。您可通过以下方式进行修复：

1. 以 LogListener 安装目录为 /usr/local/loglistenr 为例，执行以下命令打开 LogListener 配置文件：

vim etc/loglistener.conf

2. 在 LogListener 配置文件中，找到 secrete_id 和 secrete_key 并判断是否正确有效，若无效则修改为正确的密钥信息，如下图：




3. 完成配置后，保存并重启 LogListener。

# LogListener常见问题

## 为何单个文件不支持上报至多个 topic？

LogListener 采集程序的策略是，单个文件，只能上传至一个 topic。
例如，您有 topicA 和 topicB 两个文件，并对这两个文件进行如下设置。

- 设置 topicA 的采集路径为： /data/log/**/*.log
- 设置 topicB 的采集路径为： /data/log/test/**/*.log 、 /data/log/**/test*.log 、 /data/log/**/*.log ，或者其他与 topicA 采集路径类似的路径。
  此场景的配置，虽对应到两条采集路径交集的文件，但只会将数据上传至其中一个 topic 中。因此，我们建议针对不同的日志主题，应承载不同业务类型的日志，并在设置采集配置路径时，尽量精确配置信息。如果仍需要将某一个文件上传至不同的 topic，可使用软链接实现。对同一个采集目标创建不同的软链接，不同的 topic 分别采集不同的软链接路径/文件。

## 如何进行采集路径的设置？

目前采集路径的配置为：路径前缀+ "/**/"+通配文件名的形式。例如： /data/log + /**/ + *.log ==> /data/log/**/*.log 。
由于设置通配采集路径时，需要将采集路径（前缀部分）设置的尽量准确，从而使采集器能够更有效率地提供服务。

> 注意：
>
> 如果采集路径前缀部分设置不准确，可能会导致采集路径匹配到的路径数量巨大，从而造成采集器进入异常状态，无法工作。
>
> 例如，采集路径设置为：//**/*.log，其前缀部分为"/"，这种情况下采集程序会扫描整个根目录，导致采集程序无法工作。

## 推荐的日志轮转方案是什么？

对于日志的轮转，推荐方式为轮转后的文件名，不要被采集通配路径覆盖到。
例如，配置的采集路径是 /var/log/xxxx/**/*.log ，需要采集的日志是 test.log 。当 test.log 轮转成 test.2021-07-13.4.log 时，LogListener 能够识别 test.2021-07-13.4.log 是 test.log 的轮转文件，对它仍按照 test.log 来标记。因此，LogListener 保存的位点文件中，没有 test.2021-07-13.4.log 这个文件的采集记录。
而当 LogListener 重启后，LogListener 会按照 /var/log/xxxx/**/*.log 去扫描文件，并发现 test.2021-07-13.4.log 文件符合匹配规则，且没有采集记录，是一个新文件，然后重新采集。
所以建议，采集通配路径不要匹配到轮转文件，避免造成重启后重新采集轮转文件。
我们推荐的日志轮转方案是，如果您需要采集 test.log ，建议将轮转后的文件名命名为 test.log.2021-07-13.xxx ，可以有效不被 *.log 覆盖到。

## LogListener 升级说明？

在 LogListener 迭代过程中，采集路径的接口参数做过变更，比较老的版本（ver< 2.2.8）设置的采集路径，在新版

本中不再被支持。
因此，如需对存量老版本（ver < 2.2.8）进行升级，在采集程序升级之后，需要在控制台重新以通配路径的方式，再次设置采集路径。

## LogListener 采集配置如何使用正则采集模式？

在控制台设置采集配置时，如果选择正则相关的采集模式，控制台虽提供正则 kv 提取小工具，但此工具暂不提供对中文内容的正则自动生成功能。如需对中文文本进行正则提取，可以自行编写正则表达式，在控制台进行验证，或者使用其他第三方工具进行验证。

## 初次使用 LogListener 采集器接入时，发现无日志上传，怎么办？

可能是采集器配置不正确导致，常见情况如下：

- 配置的服务端域名不匹配，采集器拉取不到当前地域的采集配置，无采集业务进行。
- 采集器加入了 IP 机器组，但是采集器中又配置了标签 label 信息，导致在当前地域拉取不到采集配置，无采集业务进行。
- 采集器中配置的 secret ID/KEY 不正确，或者权限不足，导致无法上传日志。
- 环境问题（如 VPC 子网内，外网未开启），如果配置了跨地域上传，是不生效的，采集器实际上还是与本地域服务端进行通信。
- 通常这种情况下，可以登录采集器所在机器，进入采集器安装目录，并执行 ./bin/check 命令，检查如下内容：
    - 域名是否正确。
    - 心跳上报是否正常。
    - 采集配置是否正确拉取。

## 机器组使用混用，导致采集不采集，怎么办？

目前机器组分为两类，其使用方法相互独立：

- IP 机器组，机器 IP 需要在控制台上手动加入机器组，对应机器上 loglistener.conf 的group_label 需为空。
- 标签机器组，控制台设置机器组标签，对应机器上 loglistener.conf的group_label 需要设置为相同的标签。
如上两种用法不兼容，如果混合使用，采集机器将拉取不到正确的采集配置，造成不采集的现象。

## LogListener 的采集策略是什么？

一堆文件排队进行 LogListener 采集时，队首文件先采集，且要求其在某个时刻读取到文件尾才会让出队首位置。
即：不是每个文件在单位时间内，都能均等的享受采集资源。
当单个文件始终写入大于采集速度，且采集速度慢导致始终消费不到文件最新位置时，会出现某个文件长时间或一直霸占采集资源，从而导致其他的文件无法进入采集流程。

## Topic 采集阻塞严重怎么办？

在某段时间内，如果单个文件的产生速度大于采集速度，LogListener 会持续一直在采集这一个文件，陷入了对其他文件的采集阻塞场景。

## 过滤器的规则是什么？

过滤器的规则是匹配后采集，而不是匹配后丢弃。对于未能匹配的日志，LogListener 将不会进行采集。

## 如何对 LogListener 的进程进行绑核？

使用 taskset 工具进行绑核， taskset -cp ${cpu number} ${pid>} 。

## 如何处理 LogListener 占用内存过高，控制资源的使用？

- 建议升级到最新 LogListener 版本 ，并设置 memory_tight_mode = true 。
- 使用 CGroup 限制 CPU 和内存使用。

## LogListener 是否支持软链接方式采集？

LogListener 低于2.3.0版本不支持监听软连接方式的日志文件和 NFS、CIFS 等共享文件目录上的日志文件，以上版本均可支持。

## LogListener 可以向多个日志主题上传数据吗？

- LogListener 可以为同地域的多个日志主题采集数据，但不支持为异地多个日志主题采集。
- 同一个日志文件只支持采集到一个主题。

## LogListener 初始化的时候是否可以自动加入机器组？

标识机器组支持, 参见文档 管理机器组。

## LogListener 日志上传策略是什么？

- 缓存的日志量超过4M。
- 缓存的日志条数超过10000条。
- 读到文件末尾。

## LogListener 支持的最大性能是多少？

- 单行全文日志最大处理能力为115MB/s。
- 多行全文日志最大处理能力为40MB/s。
- JSON 格式日志最大处理能力为25MB/s。
- CSV 格式日志最大处理能力为50MB/s。
- 完全正则格式日志最大处理能力为18MB/s (和正则的复杂度有关)。

## 服务器更换 IP 地址后，LogListener 应该如何适配？

- 若服务器通过机器标识绑定机器组，用户无需变更 LogListener 配置。若服务器 IP 需要频繁变更，建议用户使用机器标识配置机器组。

- 若服务器通过 IP 地址绑定机器组，用户需要完成以下配置变更：

  i. 修改配置文件中 group_ip 选项，填入变更后的 IP 地址，例如：

```
sed -i '' "s/group_ip *=.*/group_ip = ${group_ip}/" etc/loglistener.conf
```

  ii. 重启 LogListener。

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

  iii. 如果使用的是 IP 机器组，登录日志服务控制台，在左侧导航栏中，单击机器组管理，修改该服务器绑定的机器组配置，使用新 IP 替换原机器 IP 地址并确定。

# LogListener安装异常问题

如何安装使用日志服务 LogListener，详情请参见 LogListener 安装指南 文档，并了解 LogListener 机制。

## 哪些原因可能导致无法正确安装 LogListener？

1. 内核版本不是64位。
2. 安装方式出错。
3. 最新特性功能依赖较高版本 LogListener。

## 如何处理 LogListener 安装异常？

1. 确认内核版本。
   LogListener 安装目录下的 bin 目录中的可执行文件只支持 Linux 64位内核，执行命令 uname -a ，确认内核版本是否为 x86_64。
2. 确认安装执行命令是否正确。
   具体请参见 LogListener 安装指南 文档进行操作。
3. 确认 Loglistener 版本。
   日志服务最新特性可能依赖新版 LogListener，若确认是使用新特性异常，请下载 LogListener 最新版本。
   LogListener 下载及详细安装步骤请参见 LogListener 安装指南。
4. 验证 LogListener 成功安装。
   参考 LogListener常用操作 检查 LogListener 进程、心跳和拉配置是否正常。

# 采集配置常见问题

## 完整日志内容如何通过完全正则采集配置？

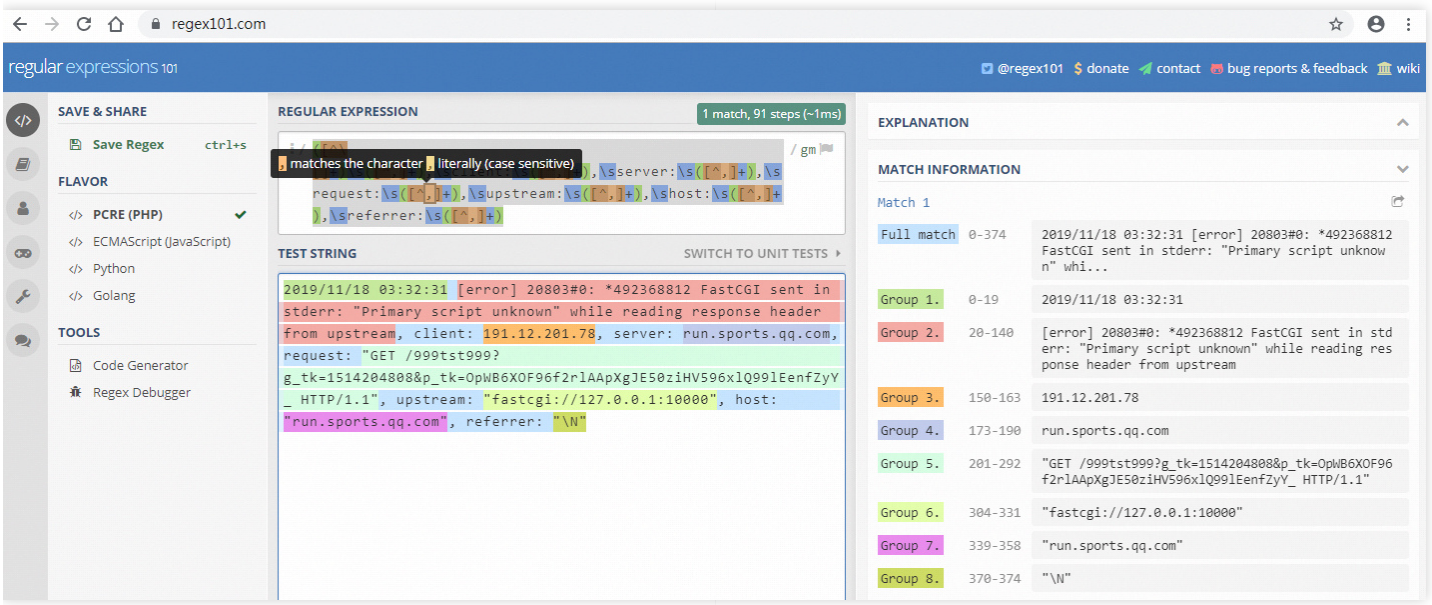
原始日志中，完整的日志格式如下：

```
2019/11/18 03:32:31 [error] 20803#0: *492368812 FastCGI sent in stderr: "Primary script unknown" while reading response header from upstream, client: 191.12.201.78, server: run.sports.abc.com, request: "GET /999tst999?g_tk=1514204808&p_tk=OpWB6XOF96f2rlAApXgJE50ziHV596xlQ99lEenfZyY_ HTTP/1.1", upstream: "fastcgi://127.0.0.1:10000", host: "run.sports.abc.com", referrer: "\N"
```

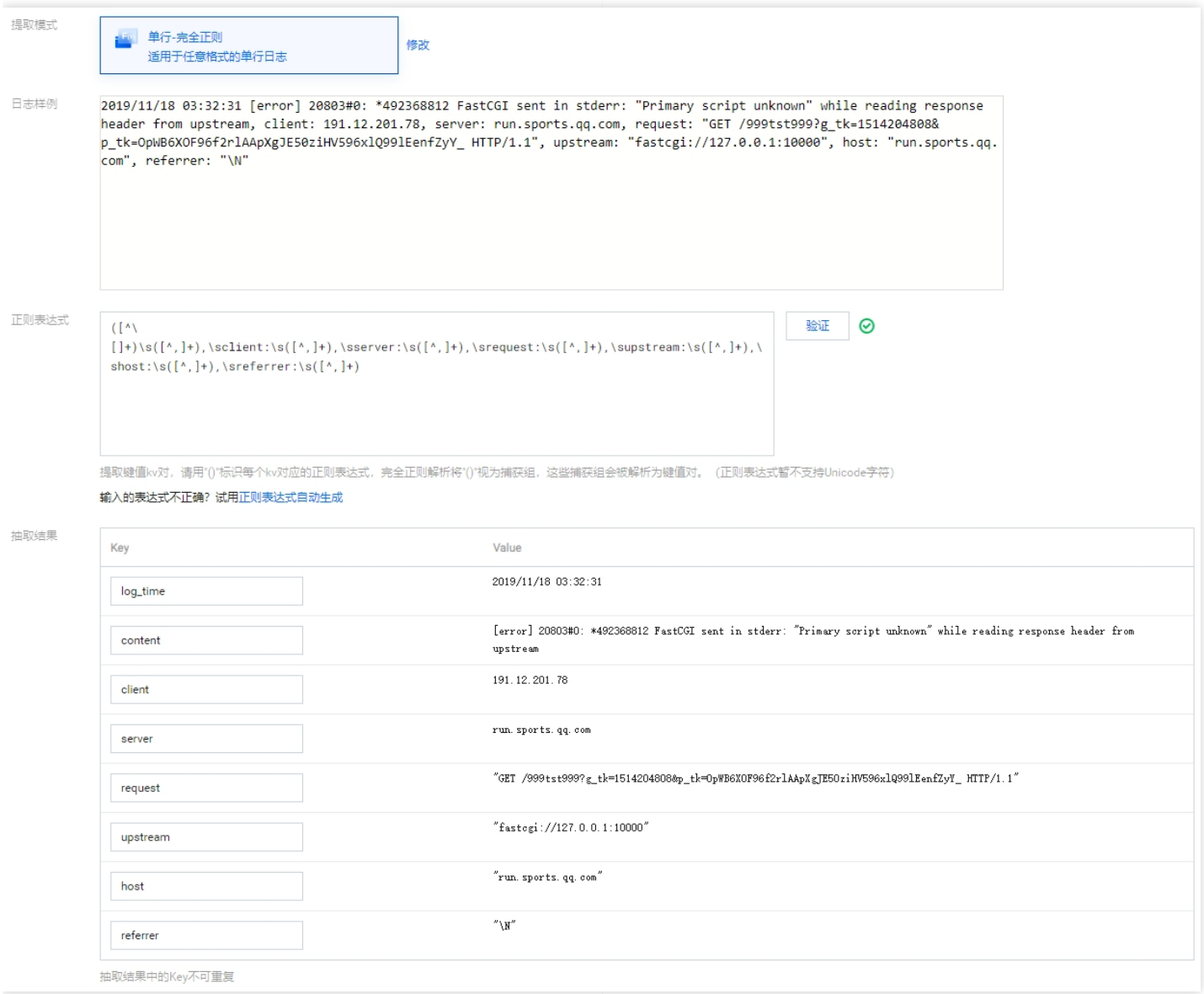
通过如下正则表达式，可以提取出对应的字段：

```
([^\[]+)\s([^,]+),\sclient:\s([^,]+),\sserver:\s([^,]+),\srequest:\s([^,]+),\supstream:\s([^,]+),\shost:\s([^,]+),\sreferrer:\s([^,]+)
```

原始日志格式化的字段如下（推荐使用 regex 构建自定义的正则表达式）：




各字段依次命名为：log_time、content、client、server、request、upstream、host、referrer。




## 日志内容字段部分缺失如何采集？

但是在某些情况下，日志中的 upstream、referrer 字段会缺失。
例如，无 upstream 有 referrer：

> 2019/11/18 04:02:38 [error] 20802#0: *492391323 access forbidden by rule, client: 45.71.63.206, server: admin.sports.abc.com, request: "GET /index HTTP/1.1", host: "admin.sports.abc.com", referrer: "http://admin.sports.abc.com/index"

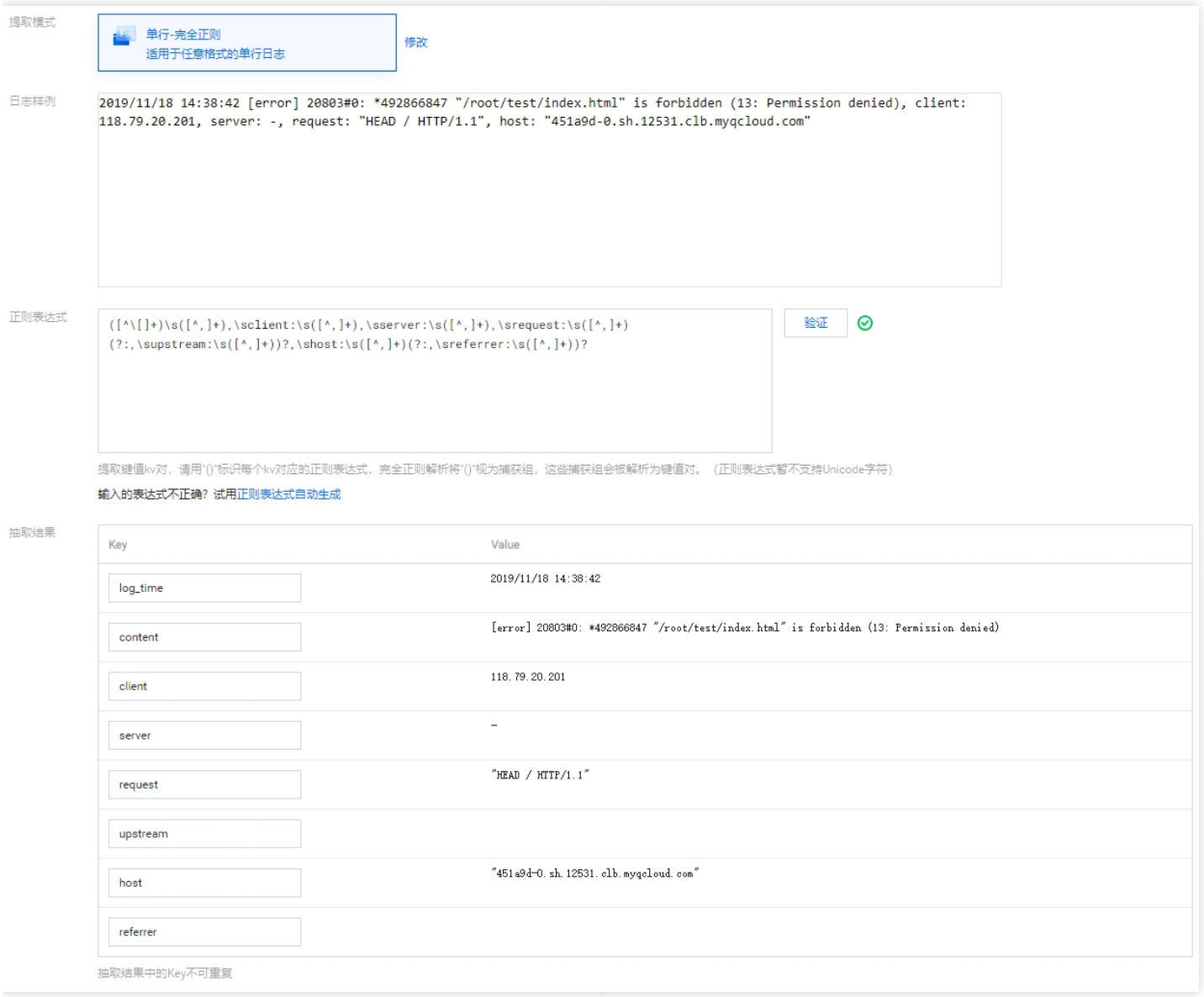
又或者，无 upstream 无 referrer：

> 2019/11/18 14:38:42 [error] 20803#0: *492866847 "/root/test/index.html" is forbidden (13: Permission denied), client: 118.79.20.201, server: -, request: "HEAD / HTTP/1.1", host: "451a9d-0.sh.12531.clb.abc.com"

此时，前文提到的正则表达式就无法适用这两种缺失的场景。
通过正则表达式中的组合的 非捕获括号语法(?:x) 以及 匹配0次或1次? 语法，既能兼容 upstream 和 referrer 缺失的场景，又能保证 捕获括号 提取的序号一致。
完整的正则表达式如下：

```
([^\[]+)\s([^,]+),\sclient:\s([^,]+),\sserver:\s([^,]+),\srequest:\s([^,]+)(?:,\supstream:\s([^,]+))?,\shost:\s([^,]+)(?:,\sreferrer:\s([^,]+))?
```

针对无 upstream 和 referrer 的日志进行测试，CLS 采集页面上的提取截图如下：




## 如何进行索引配置？

开启索引，并配置全文和键值索引。

## 索引配置

导入配置规则

索引状态 [开]

开启后可对日志进行检索分析，将产生索引流量、索引存储及相应费用。

全文索引 [开]

开启后支持使用关键词检索日志全文，例如输入 error 检索包含 error 关键词的日志。

全文分词符 @&?|#()="„;:<>[]{}/ \n\t\r\\

将日志全文按照分词符拆分成若干个分词用于检索。

大小写敏感 [关]

包含中文 [关]

日志中包含中文且需对中文进行检索时可开启该功能，将每一个汉字拆分为独立的分词用于检索。

键值索引 [开]

开启后支持使用键值检索日志，例如添加名称为level的字段，输入level:error即可检索level为error的日志。**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。**

大小写敏感 [关]

批量添加字段

按字段名称搜索

| 字段名称 | 字段类型 | 分词符 | 包含中文 | 开启统计 | |
|---|---|---|---|---|---|
| log_time | text | @&?\|#()="„;:<>[]{}/ \n\t\ | [关] | [开] | ⊗ |
| content | text | @&?\|#()="„;:<>[]{}/ \n\t\ | [关] | [开] | ⊗ |
| client | text | @&?\|#()="„;:<>[]{}/ \n\t\ | [关] | [开] | ⊗ |
| server | text | @&?\|#()="„;:<>[]{}/ \n\t\ | [关] | [开] | ⊗ |
| request | text | @&?\|#()="„;:<>[]{}/ \n\t\ | [关] | [开] | ⊗ |
| upstream | text | @&?\|#()="„;:<>[]{}/ \n\t\ | [关] | [开] | ⊗ |
| host | text | @&?\|#()="„;:<>[]{}/ \n\t\ | [关] | [开] | ⊗ |
| referrer | text | @&?\|#()="„;:<>[]{}/ \n\t\ | [关] | [开] | ⊗ |

＋ 添加字段

▸ 高级设置

确定　取消

## 如何检索日志？

构造测试日志，执行命令如下：

```
[root@VM_2_4_centos tools]# echo '2019/11/18 03:32:31 [error] 20803#0: *492368812 FastCGI sent in stderr: "Primary script unknown" while reading response header from upstream, client: 191.12.201.78, server: run.sports.abc.com, request: "GET /999tst999?g_tk=1514204808&p_tk=OpWB6XOF96f2rlAApXgJE50ziHV596xlQ99lEenfZyY_ HTTP/1.1", upstream: "fastcgi://127.0.0.1:10000", host: "run.sports.abc.com", referrer: "\N"' >> /var/log/regex/regex.log
[root@VM_2_4_centos tools]# echo '2019/11/18 04:02:38 [error] 20802#0: *492391323 access forbidden by rule, client: 45.71.63.206, server: admin.sports.abc.com, request: "GET /index HTTP/1.1", host: "admin.sports.abc.com", referrer: "http://admin.sports.abc.com/index"' >> /var/log/regex/regex.log
[root@VM_2_4_centos tools]# echo '2019/11/18 14:38:42 [error] 20803#0: *492866847 "/root/test/index.html" is forbidden (13: Permission denied), client: 118.79.20.201, server: -, request: "HEAD / HTTP/1.1", host: "451a9d-0.sh.12531.clb.abc.com"' >> /var/log/regex/regex.log
```

在日志服务控制台检索日志。

检索关键词： referrer:admin*

# 检索分析相关
# 检索不到日志问题

## 检索语句为空时，为什么检索不到任何日志？

检索语句为空时，代表无任何过滤条件、查询所有日志。如果未能查看到日志，可按以下步骤排查：

1. 选择较大的时间范围（例如最近30天），保持检索语句为空，确定是否有日志。
2. 无日志：

- 可能为当前日志主题未开启索引，请在索引配置中为该主题开启索引。
- 可能为之前日志上报时未开启索引，日志上报结束后才开启的索引，请对历史数据重建索引，或在开启索引后重新采集日志。
- 可能为日志上报失败，使用 LogListener 采集日志时，请检查 机器组状态是否出现异常，使用 SDK 等方式上报日志时请检查 SDK 自身输出日志是否存在错误。
- 可能为日志时间错误，日志上报至未来时间或已超出日志主题存储时长的时间。使用 LogListener 采集日志时，请检查日志所在服务器时间是否正确，如果使用了自定义日志时间，请检查时间格式是否配置正确。使用 SDK 等方式上报日志时，请检查日志时间参数传参是否正确。

3. 部分时间段有日志，部分时间段无日志：

- 可能为日志上报失败，使用 LogListener 采集日志时，请检查 机器组状态是否出现异常，使用 SDK 等方式上报日志时请检查 SDK 自身输出日志是否存在错误。
- 可能为该时间段确实无日志，请检查自身业务在指定时间段是否产生日志。

## 检索语句为空时能够检索到日志，为什么输入检索语句后检索不到任何日志？

有以下几种可能会导致该问题：

- 部分日志上报失败：使用 LogListener 采集日志时，请检查 机器组状态是否出现异常，使用 SDK 等方式上报日志时请检查 SDK 自身输出日志是否存在错误。
- 索引配置中分词不正确：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。
- 历史日志未采用最新的索引配置：修改索引配置后，仅针对新采集的日志生效。您可以在索引配置页面查看上一次索引修改时间，若被检索日志位于该时间前，则有可能按照当前的索引配置无法检索该历史日志，请对历史数据重建索引。
- 使用 LogListener 采集日志时，某些日志在采集环节出现解析异常，存储在了 LogParseFailure 字段下，请尝试全文检索或对 LogParseFailure 字段使用键值检索（需为 LogParseFailure 开启键值索引）。
- 某些日志在创建索引的过程中出现异常，存储在了__RAWLOG__字段下，请尝试全文检索。

## 使用日志服务控制台能够检索到日志，为什么使用 API 检索不到日志？

该问题是由于 API 传参错误导致，建议在日志服务控制台页面下使用浏览器开发者工具查看接口的请求参数，与您调用 API 的参数是否一致。最常见的问题是 SearchLog 接口的 From 和 To 参数使用的 Unix 时间戳单位错误，应当使用毫秒格式 Unix 时间戳，不支持秒格式。

# 检索分析报错

## 检索分析常见报错信息有哪些？

常见报错信息、原因及解决方案如下：

| 报错信息 | 报错原因 | 解决方案 |
| --- | --- | --- |
| SyntaxError [field: xxx, can not search on this field, since it is not indexed]<br>QueryError [illegal_argument_exception.Cannot search on field [xxx] since it is not indexed.] | 字段xxx未开启键值索引 | 为该字段开启键值索引，详情请参见 键值索引 |
| QueryError [illegal_argument_exception.Cannot search on Full-Text since it is not indexed.] | 未开启全文索引 | 开启全文检索，详情请参见 全文索引 |
| QueryError [line X:X:Column 'XXX' cannot be resolved] | SQL语句中xxx字段未开启统计，不能用于统计分析 | 为该字段开启统计，详情请参见 键值索引。另外请注意 SQL 中字符串需使用单引号包裹，双引号包括代表字段 |
| QueryError [parse_exception.XXX] | 查询语句语法错误 | 检查报错信息中指出的错误位置及错误原因，一般为检索条件语法错误（非 SQL） |
| QueryError [line X:X: XXX] | 查询语句语法错误 | 检查报错信息中指出的错误位置及错误原因，一般为 SQL 语法错误 |
| QueryError [line X:X:Function 'xxx' not registered] | SQL 语句中不支持xxx函数 | 检查函数名称是否拼写错误 |
| QueryError [line X:X:Unexpected parameters (XXX) for function XXX] | SQL 语句中函数入参数据类型不正确 | 检查函数入参，可使用 类型转换函数 修改数据类型 |
| QueryError [line X:X:'XXX' must be an aggregate expression or appear in GROUP BY clause] | SQL 语句中 GROUP BY 语法错误 | 一般为 SELECT 字段没有出现在 GROUP BY 子句中，详见 GROUP BY 语法 |

| 报错信息 | 报错原因 | 解决方案 |
| --- | --- | --- |
| QueryError [Syntax error in query statement, please check] | 查询语句语法错误 | 查询语句语法错误，但系统暂时不能明确指出具体错误位置，请检查完整的语句，或联系技术支持 |
| QueryError [illegal_argument_exception.syntax error on field [and\|or\|not], or full text search is closed] | 检索条件不支持小写逻辑操作符，小写逻辑操作符会按照普通字段进行全文检索 | 使用大写逻辑操作符 AND\|OR\|NOT，如您并不需要逻辑操作，而是全文检索包含 and\|or\|not 的日志，请开启全文索引 |
| QueryError [number_format_exception.For input string: ">"] | 数值比较语句语法错误 | 检查数值比较符号周围是否存在空格等特殊符号，正确格式参见 status:>400 |
| SyntaxError [query: *tes, prefix fuzzy query is not supported]*<br>*QueryError [parse_exception.parse_exception: Cannot parse 'xxx': '* or '?' not allowed as first character in WildcardQuery] | 不允许使用前缀模糊查询，例如 content:*example | 前缀模糊检索的使用限制及示例请参见 语法规则 |
| QueryError [circuit_breaking_exception. Analysis data is too large,please reduce the scope of data query] | 查询数据量过大 | 适当缩减查询时间范围，精确检索条件。如果仍旧报错，请联系技术支持 |
| Internal error. Please try again later RequestId: [7be994d4-xxxx-xxxxx-xxxx-9c38xxxx65de] | CLS 内部错误 | 请联系技术支持，并提供报错信息中的 RequestId |
| SearchTimeout | 查询超时 | 适当缩小数据查询范围及 SQL 复杂度，或稍后再试 |
| LimitExceeded.LogSearch | 搜索并发超过限制 | 降低查询频率（包括 API），稍后再试。如当前查询频率并不高，仍旧报错，请联系技术支持 |

# 统计分析(SQL)结果小数位数不正确

## 现象1：未保留小数位数，仅有整数部分

使用类似如下的检索分析语句时，期望能够保留小数点两位数，但结果只有整数部分：

- 检索分析语句：

  * | select count_if(status>=400)*100/count(*) as "错误日志百分比"

- 结果：

  0

即便使用 round 函数 指定需要保留的小数点位数，结果仍然只有整数部分：

- 检索分析语句：

  * | select round(count_if(status>=400)*100/count(*),2) as "错误日志百分比"

- 结果：

  0

### 原因

SQL 中，整数(bigint)之间进行除法运算时，结果仍为整数，对整数使用 round 函数也是无效的。

### 解决办法

- 方法1：
  在四则运算中，使用乘法将其中一个整数(bigint)转换为小数(double)，例如：

  * | select round(count_if(status>=400)*100.0/count(*),2) as "错误日志百分比"

- 方法2：

使用 类型转换函数 将其中一个整数(bigint)转换为小数(double)，例如：

```
* | select round(cast(count_if(status>=400)*100 as double)/count(*),2) as "错误日志百分比"
```

# 现象2：小数位数保留不正确

使用类似如下的检索分析语句时，期望能够最终保留小数点2位数，但结果保留了14位数：

- 检索分析语句：

```
* | select round(count_if(status>=400)*1.0/count(*),5)*100 as "错误日志百分比"
```

- 结果：

```
99.99900000000001
```

## 原因

count_if(status>=400)*1.0/count(*) 的结果类型为 double，使用 round 函数后结果仍为 double，再进行运算时不能再保证小数位数的正确性。

## 解决方法

将所有的运算进行完毕后，最终再统一保留小数位数，例如：

```
* | select round(count_if(status>=400)*100.0/count(*),3) as "错误日志百分比"
```

# 其他问题

## 日志服务 CLS 是什么？

日志服务（Cloud Log Service，CLS）提供一站式的日志数据解决方案。您无需关注扩缩容等资源问题，五分钟快速便捷接入，即可享受日志的采集、加工、检索分析、消费投递、生成仪表盘、告警等全方位稳定可靠服务。全面提升问题定位、指标监控的效率，大大降低日志运维门槛。
日志服务主要提供以下功能：

- 日志采集：便捷实时采集跨地域、多渠道、多平台、不同数据源的日志数据，轻松采集多种其他云产品日志。
- 日志检索分析：使用关键词检索日志，帮助用户快速定位异常日志，同时支持使用 SQL 对日志进行统计分析，获取日志条数随时间变化趋势、错误日志比例等统计指标。
- 日志数据加工：日志过滤、清洗、脱敏、富化、分发、结构化。
- 日志投递与消费：投递到存储、中间件，消费到流计算。
- 仪表盘：将检索分析结果快速生成自定义 Dashborad。
- 告警：异常日志秒级告警，支持通过电话、短信、邮件、微信、企业微信和自定义接口回调等方式通知用户。

## 日志服务如何定义一条日志？

日志（Log）是应用系统运行过程中产生的记录数据，如用户操作日志、接口访问日志、系统错误日志等。日志通常以文本的形式存储在应用系统所在的机器上，一条系统运行记录对应的日志可能为一行文本（单行日志），也可能为多行文本（多行日志）。

## 日志可以保存多长时间？

日志服务提供日志生命周期管理，在创建日志主题时可以指定日志的有效保存周期，支持保存1 - 3600天或永久保存，逾期后数据将会被清理。

## 日志集和日志主题的区别是什么？

日志主题（Topic）是日志数据在日志服务（Cloud Log Service，CLS）平台进行采集、存储、检索和分析的基本单元，采集到的海量日志以日志主题为单元进行管理，包括采集规则配置、保存时间配置、日志检索分析以及日志下载/消费/投递等。
日志集（Logset）是对日志主题的分类，一个日志集可包含多个日志主题。日志集本身不存储任何日志数据，仅方便用户管理日志主题。

## 单个日志主题最多可采集多少日志？

为应对海量日志采集需求，单个日志主题包含多个主题分区，每个主题分区写请求最大为500QPS，写流量最大为5MB/s。采集日志量较大时，建议开启主题分区自动分裂功能（默认开启），单个日志主题最大可拥有50个分区，此时单个日志主题的写请求最大为50 * 500 = 25000QPS，写流量最大为50 * 5 = 250MB/s。
此处的写请求及写流量并不能简单的等于日志条数及日志量，日志上传时会将多条日志打包为一个日志组并进行压缩，实际支持的日志条数及日志量将远大于上述限制。使用 Loglistener 时将自动进行日志打包及压缩，您无需关注

具体的打包策略。

## 什么是索引和分词？

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。创建索引实际上是将原始日志按指定的符号切分为多个片段（即分词），并对分词进行倒排索引的过程。

## 全文索引和键值索引有什么区别？

- 全文索引：全文索引将原始日志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）。例如输入 error 表示检索包含 error 关键词的日志。
- 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）。例如输入 level:error 表示检索 level 字段中包含 error 的日志。

## 检索和分析有什么区别？

- 检索：根据指定的条件查找匹配的原始日志，例如使用 status:404 检索响应状态码为404的应用请求日志。
- 分析：针对符合检索条件的日志使用 SQL 进行统计分析，例如使用 status:404 | select count(*) as logCounts 统计响应状态码为404的应用请求日志数量。

## 日志检索分析性能如何？

- 检索性能：百亿级别日志，秒级返回结果。
- 分析性能：亿级别日志，秒级返回结果；百亿级别日志，1分钟内返回结果。该性能与分析使用的 SQL 复杂度有很大的关系，SQL 非常复杂时可能低于该性能指标。

## 从日志生成到可以检索到需要多久？

使用 Loglistener 采集日志时，延迟不超过1分钟。使用 API 及 SDK 采集日志时，从发起 API 调用到可以检索到日志不超过1分钟。

## 服务器更换 IP 地址后，Loglistener 应该如何适配？

- 若服务器通过机器标识绑定机器组，用户无需变更 Loglistener 配置。若服务器 IP 需要频繁变更，建议用户使用机器标识配置机器组。
- 若服务器通过 IP 地址绑定机器组，用户需要完成以下配置变更：

  i. 修改 Loglistener 安装目录下的 /etc/loglistener.conf 文件。此处安装目录以 /user/local 为例：

  vi /usr/local/loglistener-2.3.0/etc/loglistener.conf

  ii. 键盘按 i 键，进入编辑模式。

  iii. 修改配置文件中 group_ip 部分，填入变更后的 IP 地址。

iv. 保存设置并退出编辑器，具体操作步骤：按 Esc 键，输入 :wq，按 Enter 键。

v. 执行如下命令，重启 Loglistener。

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

vi. 登录日志服务控制台，在左侧导航栏中，单击机器组管理，修改该服务器绑定的机器组配置，使用新 IP 替换原机器 IP 地址，单击确定。

## 如何排查测试告警通知渠道报错或未收到测试消息？

情况1：页面显示“发送失败”
鼠标在“发送失败”上悬停可查看错误码及详细的失败原因，常见的错误码如下：

| 错误码 | 含义 | 排查方式 |
| --- | --- | --- |
| -1004 | 该通知渠道消息发送失败 | 一般是由于接受对象中的用户或用户组未配置或验证手机、邮箱或微信导致，可在用户列表 查看并配置。 |
| -1005 | 该通知渠道部分消息发送失败，例如部分用户未成功发送，或部分渠道未成功发送 | 一般是由于接受对象中的部分用户或用户组未配置或验证手机、邮箱或微信导致，可在用户列表中查看并配置。 |
| -1006 | 企业微信或自定义接口回调报错 | 根据详细的失败原因进一步排查，常见的错误包括：<br>- invalid URI for request：不是一个合法的 URL 地址。<br>- i/o timeout：接口访问超时，请检查接口地址及能否通过公网直接访问。<br>- callback custom error with status:xxx：接口响应报错，请检查接口地址及后端服务是否正常。<br>- http status code is 400：接口 HTTP 状态码为 400<br>- ssrf attack：回调接口地址需为公网可直接访问的地址，接口地址为内网时可能出现该错误。 |

> 说明：
>
> 测试“自定义接口回调”告警渠道时，平台采用固定的请求参数调用该自定义接口，可能会存在请求参数不符合接口要求而导致通知发送失败的情况，此时测试通知渠道功能无实际意义，您可直接在告警策略中按接口要求配置合适的请求头和请求内容。

情况2：页面显示“已发送”，但实际未收到测试消息
根据接收渠道，常见原因如下：

| 接收渠道 | 原因 |
| --- | --- |

| 接收渠道 | 原因 |
| --- | --- |
| 邮件、短信、微信、电话 | 为避免重复通知干扰用户，一天内仅允许向同一用户使用同一渠道发送一次测试消息。 |
| 企业微信 | 测试消息不符合企业微信接口要求，消息被忽略，此时测试通知渠道功能无实际意义，您可直接在告警策略中按企业微信 API 要求配置合适的请求头和请求内容来发送告警，详情可参见 管理通知渠道组。 |
| 自定义接口回调（其它地址） | CLS 以 HTTP 响应状态码来判断消息发送是否成功，请检查自定义接口是否存在 HTTP 响应状态码正常，但仍存在其他业务逻辑限制的情况。 |

# 词汇表

## 分词

分词是指在一个较长的日志中，按照符号或中文将日志切分为多个片段以便于检索的过程。

## 键值索引

键值索引（Key-Value Index）是检索分析的另一种索引类型，键值索引将原始日志拆分成多个 Key-Value 对，每组 Key-Value 对需要定义唯一的 Key 名称，键值索引的查询语法是 Key：Value。

## 机器组

机器组是一组需要采集日志的机器列表，一个日志主题可以绑定多个机器组，一个机器组也可以被多个日志主题绑定。

## LogListener

日志服务 CLS 提供采集 Agent（LogListener），成功安装的 LogListener 将与日志服务保持长久的心跳连接。

## 全文索引

全文索引（Full-Text Index）是查询检索的一种索引类型，全文索引将原始日志拆分成多个分词，检索日志时基于分词进行查询，输入关键词即可查询。

## 日志集

日志集（Logset）是日志服务 CLS 提供的项目管理逻辑概念，可以将一个日志集对应一个项目。

## 日志主题

日志主题（Topic）是日志服务 CLS 提供的基本管理单元，一个日志主题对应一个应用或者服务。日志主题 Topic 是 CLS 的最小管理单元，采集、索引、投递等配置围绕 Topic 进行。一个日志集可以包含多个日志主题。

## 索引

为了快速检索出需要的日志，日志服务对上传至平台的日志进行包括分词在内的很多预处理，这个过程称之为创建“索引（index）”。索引决定了日志能够以什么样的条件来进行检索和分析，因此在上传日志数据前，需要为日志主题设置一个合理的索引规则，以方便后续检索分析。索引主要包括全文索引及键值索引两类，详细介绍请参见 配置索引。

## 元数据

元数据（Tag）是对日志本身的描述或分类，例如容器日志对应的容器集群或 Pod 信息。上传日志到日志服务时，元数据通过 LogTag 字段传递（详情可参见 上传结构化日志 中的LogTag字段 ），而原始日志内容通过 Log 字段传递。所有通过LogTag 传递的数据，配置索引时，均需配置元数据索引。

# API文档
# 日志服务（cls）
# 版本（2020-10-16）
# API 概览

## API版本

V3

## COS数据导入任务相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| SearchCosRechargeInfo | 预览cos导入信息 |

## Kafka协议消费相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| CloseKafkaConsumer | 关闭Kafka协议消费 |
| DescribeKafkaConsumer | 获取Kafka协议消费信息 |
| ModifyKafkaConsumer | 修改Kafka协议消费信息 |
| OpenKafkaConsumer | 打开Kafka协议消费 |

## 主题分区相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| DescribePartitions | 获取分区列表 |
| MergePartition | 合并分区 |

| 接口名称 | 接口功能 |
| --- | --- |
| SplitPartition | 分裂主题分区 |

# 仪表盘相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| DescribeDashboards | 获取仪表盘 |

# 告警策略相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| CreateAlarm | 创建告警策略 |
| CreateAlarmNotice | 创建告警通知模板 |
| DeleteAlarm | 删除告警策略 |
| DeleteAlarmNotice | 删除告警通知模板 |
| DescribeAlarmNotices | 获取告警通知模板列表 |
| DescribeAlarms | 获取告警策略列表 |
| DescribeAlertRecordHistory | 告警历史记录 |
| GetAlarmLog | 获取告警任务历史记录 |
| ModifyAlarm | 修改告警策略 |
| ModifyAlarmNotice | 修改告警通知模板 |

# 对象存储数据导入任务相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| CreateCosRecharge | 创建对象存储导入任务 |
| DescribeCosRecharges | 获取对象存储导入配置 |

| 接口名称 | 接口功能 |
| --- | --- |
| ModifyCosRecharge | 修改对象存储导入任务 |

# 投递Ckafka任务相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| CreateConsumer | 创建投递Ckafka任务 |

# 投递对象存储任务相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| CreateShipper | 创建投递规则 |
| DeleteShipper | 删除投递规则 |
| DescribeShipperTasks | 获取投递任务列表 |
| DescribeShippers | 获取投递规则 |
| ModifyShipper | 修改投递规则 |
| RetryShipperTask | 重试失败的投递任务 |

# 日志主题相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| CreateTopic | 创建日志主题 |
| DeleteTopic | 删除日志主题 |
| DescribeTopics | 获取日志主题列表 |
| ModifyTopic | 修改日志主题 |

# 日志相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| DescribeLogContext | 上下文检索 |
| DescribeLogHistogram | 获取直方图 |
| SearchLog | 检索分析日志 |

# 日志集相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| CreateLogset | 创建日志集 |
| DeleteLogset | 删除日志集 |
| DescribeLogsets | 获取日志集列表 |
| ModifyLogset | 修改日志集 |

# 机器组相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| AddMachineGroupInfo | 添加机器组信息 |
| CreateMachineGroup | 创建机器组 |
| DeleteMachineGroup | 删除机器组 |
| DeleteMachineGroupInfo | 删除机器组信息 |
| DescribeMachineGroups | 获取机器组列表 |
| DescribeMachines | 获取机器状态 |
| ModifyMachineGroup | 修改机器组 |

# 索引相关接口

| 接口名称 | 接口功能 |
| --- | --- |

| 接口名称 | 接口功能 |
| --- | --- |
| CreateIndex | 创建索引 |
| DeleteIndex | 删除索引配置 |
| DescribeIndex | 获取索引配置信息 |
| ModifyIndex | 修改索引 |

# 采集配置相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| ApplyConfigToMachineGroup | 应用采集配置到指定机器组 |
| CreateConfig | 创建采集规则配置 |
| CreateConfigExtra | 创建特殊采集配置任务 |
| DeleteConfig | 删除采集规则配置 |
| DeleteConfigFromMachineGroup | 删除应用到机器组的采集配置 |
| DescribeConfigExtras | 获取特殊采集配置 |
| DescribeConfigMachineGroups | 获取采集规则配置所绑定的机器组 |
| DescribeConfigs | 获取采集规则配置 |
| DescribeMachineGroupConfigs | 获取机器组绑定的采集规则配置 |
| ModifyConfig | 修改采集规则配置 |

# 调用方式
# 接口签名v1

TCloudFinanceZone API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。
签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往云API密钥页面申请，否则无法调用云API接口。

## 1. 申请安全凭证

在第一次使用云API之前，请前往云API密钥页面申请安全凭证。
安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- 用户必须严格保管安全凭证，避免泄露。

申请安全凭证的具体步骤如下：

1. 登录TCloudFinanceZone管理中心控制台。
2. 前往云API密钥的控制台页面
3. 在云API密钥页面，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

## 2. 生成签名串

有了安全凭证SecretId 和 SecretKey后，就可以生成签名串了。以下是生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

注意：这里只是示例，请根据用户实际申请的 SecretId 和 SecretKey 进行后续操作！

以云服务器查看实例列表(DescribeInstances)请求为例，当用户调用这一接口时，其请求参数可能如下:

| 参数名称 | 中文 | 参数值 |
| --- | --- | --- |

| 参数名称 | 中文 | 参数值 |
| --- | --- | --- |
| Action | 方法名 | DescribeInstances |
| SecretId | 密钥Id | AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE |
| Timestamp | 当前时间戳 | 1465185768 |
| Nonce | 随机正整数 | 11886 |
| Region | 实例所在区域 | shjr |
| InstanceIds.0 | 待查询的实例ID | ins-09dx96dg |
| Offset | 偏移量 | 0 |
| Limit | 最大允许输出 | 20 |
| Version | 接口版本号 | 2017-03-12 |

## 2.1. 对参数排序

首先对所有请求参数按参数名的字典序（ ASCII 码）升序排序。注意：1）只按参数名进行排序，参数值保持对应即可，不参与比大小；2）按 ASCII 码比大小，如 InstanceIds.2 要排在 InstanceIds.12 后面，不是按字母表，也不是按数值。用户可以借助编程语言中的相关排序函数来实现这一功能，如 php 中的 ksort 函数。上述示例参数的排序结果如下:

```
{
    'Action' : 'DescribeInstances',
    'InstanceIds.0' : 'ins-09dx96dg',
    'Limit' : 20,
    'Nonce' : 11886,
    'Offset' : 0,
    'Region' : 'shjr',
    'SecretId' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE',
    'Timestamp' : 1465185768,
    'Version': '2017-03-12',
}
```

使用其它程序设计语言开发时，可对上面示例中的参数进行排序，得到的结果一致即可。

## 2.2. 拼接请求字符串

此步骤生成请求字符串。
将把上一步排序好的请求参数格式化成“参数名称”=“参数值”的形式，如对 Action 参数，其参数名称为 "Action" ，参数值为 "DescribeInstances" ，因此格式化后就为 Action=DescribeInstances 。
注意：“参数值”为原始值而非url编码后的值。

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

## 2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机:查看实例列表(DescribeInstances)的请求域名为：cvm.finance.cloud.tencent.com。实际的请求域名根据接口所属模块的不同而不同，详见各接口说明。
3. 请求路径: 当前版本云API的请求路径固定为 / 。
4. 请求字符串: 即上一步生成的请求字符串。

签名原文串的拼接规则为: 请求方法 + 请求主机 +请求路径 + ？ + 请求字符串

示例的拼接结果为：

```
GETcvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

## 2.4. 生成签名串

此步骤生成签名串。
首先使用 HMAC-SHA1 算法对上一步中获得的签名原文字符串进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例:

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3EXAMPLE';
$srcStr = 'GETcvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为:

```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

使用其它程序设计语言开发时，可用上面示例中的原文进行签名验证，得到的签名串与例子中的一致即可。

# 3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 EliP9YW3pW28FpsEdkXt/+WcGeI= ，最终得到的签名串请求参数（Signature）为：EliP9YW3pW28FpsEdkXt%2f%2bWcGeI%3d，它将用于生成最终的请求 URL。

注意：如果用户的请求方法是 GET，或者请求方法为 POST 同时 Content-Type 为 application/x-www-form-urlencoded，则发送请求时所有请求参数的值均需要做 URL 编码，参数键和=符号不需要编码。非 ASCII 字符在 URL 编码前需要先以 UTF-8 进行编码。

注意：有些编程语言的 http 库会自动为所有参数进行 urlencode，在这种情况下，就不需要对签名串进行 URL 编码了，否则两次 URL 编码会导致签名失败。

注意：其他参数值也需要进行编码，编码采用 RFC 3986。使用 %XY 对特殊字符例如汉字进行百分比编码，其中“X”和“Y”为十六进制字符（0-9 和大写字母 A-F），使用小写将引发错误。

# 4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理

| 错误代码 | 错误描述 |
| --- | --- |
| AuthFailure.SignatureExpire | 签名过期 |
| AuthFailure.SecretIdNotFound | 密钥不存在 |
| AuthFailure.SignatureFailure | 签名错误 |
| AuthFailure.TokenFailure | token 错误 |
| AuthFailure.InvalidSecretId | 密钥非法（不是云 API 密钥类型） |

# 5. 签名演示

在实际调用 API 3.0 时，推荐使用配套的TCloudFinanceZone SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java

- PHP
- Go
- Node

为了更清楚的解释签名过程，下面以实际编程语言为例，将上述的签名过程具体实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK 。

最终输出的 url 可能为： https://cvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=shjr&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Signature=EliP9YW3pW28FpsEdkXt%2F%2BWcGeI%3D&Timestamp=1465185768&Version=2017-03-12

注意：由于示例中的密钥是虚构的，时间戳也不是系统当前时间，因此如果将此 url 在浏览器中打开或者用 curl 等命令调用时会返回鉴权错误：签名过期。为了得到一个可以正常返回的 url ，需要修改示例中的 SecretId 和 SecretKey 为真实的密钥，并使用系统当前时间戳作为 Timestamp 。

注意：在下面的示例中，不同编程语言，甚至同一语言每次执行得到的 url 可能都有所不同，表现为参数的顺序不同，但这并不影响正确性。只要所有参数都在，且签名计算正确即可。

注意：以下代码仅适用于 API 3.0，不能直接用于其他的签名流程，即使是旧版的 API ，由于存在细节差异也会导致签名计算错误，请以对应的实际文档为准。

### Java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class CloudAPIDemo {
    private final static String CHARSET = "UTF-8";

    public static String sign(String s, String key, String method) throws Exception {
        Mac mac = Mac.getInstance(method);
        SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAlgorithm());
        mac.init(secretKeySpec);
        byte[] hash = mac.doFinal(s.getBytes(CHARSET));
        return DatatypeConverter.printBase64Binary(hash);
    }

    public static String getStringToSign(TreeMap<String, Object> params) {
        StringBuilder s2s = new StringBuilder("GETcvm.finance.cloud.tencent.com/?");
```

```
        // 签名时要求对参数进行字典排序，此处用TreeMap保证顺序
        for (String k : params.keySet()) {
            s2s.append(k).append("=").append(params.get(k).toString()).append("&");
        }
        return s2s.toString().substring(0, s2s.length() - 1);
    }

    public static String getUrl(TreeMap<String, Object> params) throws UnsupportedEncodingException
{
        StringBuilder url = new StringBuilder("https://cvm.finance.cloud.tencent.com/?");
        // 实际请求的url中对参数顺序没有要求
        for (String k : params.keySet()) {
            // 需要对请求串进行urlencode，由于key都是英文字母，故此处仅对其value进行urlencode
            url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).app
end("&");
        }
        return url.toString().substring(0, url.length() - 1);
    }

    public static void main(String[] args) throws Exception {
        TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
        // 实际调用时应当使用随机数，例如：params.put("Nonce", new Random().nextInt(java.lang.Intege
r.MAX_VALUE));
        params.put("Nonce", 11886); // 公共参数
        // 实际调用时应当使用系统当前时间，例如：  params.put("Timestamp", System.currentTimeMillis() /
1000);
        params.put("Timestamp", 1465185768); // 公共参数
        params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"); // 公共参数
        params.put("Action", "DescribeInstances"); // 公共参数
        params.put("Version", "2017-03-12"); // 公共参数
        params.put("Region", "shjr"); // 公共参数
        params.put("Limit", 20); // 业务参数
        params.put("Offset", 0); // 业务参数
        params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
        params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE
", "HmacSHA1")); // 公共参数
        System.out.println(getUrl(params));
    }
}
```

## Python

注意：如果是在 Python 2 环境中运行，需要先安装 requests 依赖包：  pip install requests 。

```
# -*- coding: utf8 -*-
import base64
```

```
import hashlib
import hmac
import time

import requests

secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

def get_string_to_sign(method, endpoint, params):
    s = method + endpoint + "/?"
    query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
    return s + query_str

def sign_str(key, s, method):
    hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
    return base64.b64encode(hmac_str)

if __name__ == '__main__':
    endpoint = "cvm.finance.cloud.tencent.com"
    data = {
        'Action' : 'DescribeInstances',
        'InstanceIds.0' : 'ins-09dx96dg',
        'Limit' : 20,
        'Nonce' : 11886,
        'Offset' : 0,
        'Region' : 'shjr',
        'SecretId' : secret_id,
        'Timestamp' : 1465185768, # int(time.time())
        'Version': '2017-03-12'
    }
    s = get_string_to_sign("GET", endpoint, data)
    data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
    print(data["Signature"])
    # 此处会实际调用，成功后可能产生计费
    # resp = requests.get("https://" + endpoint, params=data)
    # print(resp.url)
```

# 接口签名v3

TCloudFinanceZone API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。
签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往云API密钥页面申请，否则无法调用云API接口。

## 1. 申请安全凭证

在第一次使用云API之前，请前往云API密钥页面申请安全凭证。
安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- 用户必须严格保管安全凭证，避免泄露。

申请安全凭证的具体步骤如下：

1. 登录TCloudFinanceZone管理中心控制台。
2. 前往云API密钥的控制台页面
3. 在云API密钥页面，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

## 2. TC3-HMAC-SHA256 签名方法

注意：对于GET方法，只支持 Content-Type: application/x-www-form-urlencoded 协议格式。对于POST方法，目前支持 Content-Type: application/json 以及 Content-Type: multipart/form-data 两种协议格式，json 格式默认所有业务接口均支持，multipart 格式只有特定业务接口支持，此时该接口不能使用 json 格式调用，参考具体业务接口文档说明。

下面以云服务器查询广州区实例列表作为例子，分步骤介绍签名的计算过程。我们仅用到了查询实例列表的两个参数：Limit 和 Offset，使用 GET 方法调用。

假设用户的 SecretId 和 SecretKey 分别是：AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE 和 Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

### 2.1. 拼接规范请求串

按如下格式拼接规范请求串（CanonicalRequest）：

```
CanonicalRequest =
    HTTPRequestMethod + '\n' +
    CanonicalURI + '\n' +
    CanonicalQueryString + '\n' +
    CanonicalHeaders + '\n' +
    SignedHeaders + '\n' +
    HashedRequestPayload
```

- HTTPRequestMethod：HTTP 请求方法（GET、POST ），本示例中为 GET；
- CanonicalURI：URI 参数，API 3.0 固定为正斜杠（/）；
- CanonicalQueryString：发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串，对于 GET 请求，则为 URL 中问号（?）后面的字符串内容，本示例取值为：Limit=10&Offset=0。注意：CanonicalQueryString 需要经过 URL 编码。
- CanonicalHeaders：参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入自定义的头部参与签名以提高自身请求的唯一性和安全性。拼接规则：1）头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接；2）多个头部，按照头部 key（小写）的字典排序进行拼接。此例中为： content-type:application/x-www-form-urlencoded\nhost:cvm.finance.cloud.tencent.com\n
- SignedHeaders：参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。拼接规则：1）头部 key 统一转成小写；2）多个头部 key（小写）按照字典排序进行拼接，并且以分号（;）分隔。此例中为： content-type;host
- HashedRequestPayload：请求正文的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，对 HTTP 请求整个正文 payload 做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。注意：对于 GET 请求，RequestPayload 固定为空字符串，对于 POST 请求，RequestPayload 即为 HTTP 请求正文 payload。

根据以上规则，示例中得到的规范请求串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
GET
/
Limit=10&Offset=0
content-type:application/x-www-form-urlencoded
host:cvm.finance.cloud.tencent.com

content-type;host
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

## 2.2. 拼接待签名字符串

按如下格式拼接待签名字符串：

```
StringToSign =
    Algorithm + \n +
```

```
RequestTimestamp + \n +
CredentialScope + \n +
HashedCanonicalRequest
```

- Algorithm：签名算法，目前固定为 TC3-HMAC-SHA256；
- RequestTimestamp：请求时间戳，即请求头部的 X-TC-Timestamp 取值，如上示例请求为 1539084154；
- CredentialScope：凭证范围，格式为 Date/service/tc3_request，包含日期、所请求的服务和终止字符串（tc3_request）。Date 为 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service 为产品名，必须与调用的产品域名一致，例如 cvm。如上示例请求，取值为 2018-10-09/cvm/tc3_request；
- HashedCanonicalRequest：前述步骤拼接所得规范请求串的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。

注意：

> 1. Date 必须从时间戳 X-TC-Timestamp 计算得到，且时区为 UTC+0。如果加入系统本地时区信息，例如东八区，将导致白天和晚上调用成功，但是凌晨时调用必定失败。假设时间戳为 1551113065，在东八区的时间是 2019-02-26 00:44:25，但是计算得到的 Date 取 UTC+0 的日期应为 2019-02-25，而不是 2019-02-26。
> 2. Timestamp 必须是当前系统时间，且需确保系统时间和标准时间是同步的，如果相差超过五分钟则必定失败。如果长时间不和标准时间同步，可能导致运行一段时间后，请求必定失败（返回签名过期错误）。

根据以上规则，示例中得到的待签名字符串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
TC3-HMAC-SHA256
1539084154
2018-10-09/cvm/tc3_request
91c9c192c14460df6c1ffc69e34e6c5e90708de2a6d282cccf957dbf1aa7f3a7
```

## 2.3. 计算签名

1）计算派生签名密钥，伪代码如下

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

- SecretKey：原始的 SecretKey；
- Date：即 Credential 中的 Date 字段信息，如上示例，为2018-10-09；
- Service：即 Credential 中的 Service 字段信息，如上示例，为 cvm；

2）计算签名，伪代码如下

```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

- SecretSigning：即以上计算得到的派生签名密钥；
- StringToSign：即步骤2计算得到的待签名字符串；

## 2.4. 拼接 Authorization

按如下格式拼接 Authorization：

```
Authorization =
    Algorithm + ' ' +
    'Credential=' + SecretId + '/' + CredentialScope + ', ' +
    'SignedHeaders=' + SignedHeaders + ', '
    'Signature=' + Signature
```

- Algorithm：签名方法，固定为 TC3-HMAC-SHA256；
- SecretId：密钥对中的 SecretId；
- CredentialScope：见上文，凭证范围；
- SignedHeaders：见上文，参与签名的头部信息；
- Signature：签名值

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
```

最终完整的调用信息如下：

```
https://cvm.finance.cloud.tencent.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.finance.cloud.tencent.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: shjr
```

# 3. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理

| 错误代码 | 错误描述 |
| --- | --- |
| AuthFailure.SignatureExpire | 签名过期 |
| AuthFailure.SecretIdNotFound | 密钥不存在 |
| AuthFailure.SignatureFailure | 签名错误 |
| AuthFailure.TokenFailure | token 错误 |
| AuthFailure.InvalidSecretId | 密钥非法（不是云 API 密钥类型） |

# 4. 签名演示

Java

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.bind.DatatypeConverter;

import org.apache.commons.codec.digest.DigestUtils;

public class CloudAPITC3Demo {
    private final static String CHARSET = "UTF-8";
    private final static String ENDPOINT = "cvm.finance.cloud.tencent.com";
    private final static String PATH = "/";
    private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE";
    private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE";
    private final static String CT_X_WWW_FORM_URLENCODED = "application/x-www-form-urlencode
d";
    private final static String CT_JSON = "application/json";
```

```
    private final static String CT_FORM_DATA = "multipart/form-data";

    public static byte[] sign256(byte[] key, String msg) throws Exception {
        Mac mac = Mac.getInstance("HmacSHA256");
        SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
        mac.init(secretKeySpec);
        return mac.doFinal(msg.getBytes(CHARSET));
    }

    public static void main(String[] args) throws Exception {
        String service = "cvm";
        String host = "cvm.finance.cloud.tencent.com";
        String region = "shjr";
        String action = "DescribeInstances";
        String version = "2017-03-12";
        String algorithm = "TC3-HMAC-SHA256";
        String timestamp = "1539084154";
        //String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        // 注意时区，否则容易出错
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

        // ************* 步骤 1：拼接规范请求串 *************
        String httpRequestMethod = "GET";
        String canonicalUri = "/";
        String canonicalQueryString = "Limit=10&Offset=0";
        String canonicalHeaders = "content-type:application/x-www-form-urlencoded\n" + "host:" + host
+ "\n";
        String signedHeaders = "content-type;host";
        String hashedRequestPayload = DigestUtils.sha256Hex("");
        String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryStri
ng + "\n"
                + canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
        System.out.println(canonicalRequest);

        // ************* 步骤 2：拼接待签名字符串 *************
        String credentialScope = date + "/" + service + "/" + "tc3_request";
        String hashedCanonicalRequest = DigestUtils.sha256Hex(canonicalRequest.getBytes(CHARSET));
        String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCan
onicalRequest;
        System.out.println(stringToSign);

        // ************* 步骤 3：计算签名 *************
        byte[] secretDate = sign256(("TC3" + SECRET_KEY).getBytes(CHARSET), date);
        byte[] secretService = sign256(secretDate, service);
        byte[] secretSigning = sign256(secretService, "tc3_request");
```

```
        String signature = DatatypeConverter.printHexBinary(sign256(secretSigning, stringToSign)).toLow
erCase();
        System.out.println(signature);

        // ************* 步骤 4：拼接 Authorization *************
        String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
                + "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
        System.out.println(authorization);

        TreeMap<String, String> headers = new TreeMap<String, String>();
        headers.put("Authorization", authorization);
        headers.put("Host", host);
        headers.put("Content-Type", CT_X_WWW_FORM_URLENCODED);
        headers.put("X-TC-Action", action);
        headers.put("X-TC-Timestamp", timestamp);
        headers.put("X-TC-Version", version);
        headers.put("X-TC-Region", region);
    }
}
```

## Python

```
# -*- coding: utf-8 -*-
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# 密钥参数
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

service = "cvm"
host = "cvm.finance.cloud.tencent.com"
endpoint = "https://" + host
region = "shjr"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
timestamp = 1539084154
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 10, "Offset": 0}

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = "GET"
canonical_uri = "/"
canonical_querystring = "Limit=10&Offset=0"
ct = "x-www-form-urlencoded"
```

```
payload = ""
if http_request_method == "POST":
    canonical_querystring = ""
    ct = "json"
    payload = json.dumps(params)
canonical_headers = "content-type:application/%s\nhost:%s\n" % (ct, host)
signed_headers = "content-type;host"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
                     canonical_uri + "\n" +
                     canonical_querystring + "\n" +
                     canonical_headers + "\n" +
                     signed_headers + "\n" +
                     hashed_request_payload)
print(canonical_request)

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
                  str(timestamp) + "\n" +
                  credential_scope + "\n" +
                  hashed_canonical_request)
print(string_to_sign)

# ************* 步骤 3：计算签名 *************
# 计算签名摘要函数
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
print(signature)

# ************* 步骤 4：拼接 Authorization *************
authorization = (algorithm + " " +
                 "Credential=" + secret_id + "/" + credential_scope + ", " +
                 "SignedHeaders=" + signed_headers + ", " +
                 "Signature=" + signature)
print(authorization)

# 公共参数添加到请求头部
headers = {
    "Authorization": authorization,
    "Host": host,
    "Content-Type": "application/%s" % ct,
```

```
    "X-TC-Action": action,
    "X-TC-Timestamp": str(timestamp),
    "X-TC-Version": version,
    "X-TC-Region": region,
}
```

# 请求结构

## 1. 服务地址

地域（Region）是指物理的数据中心的地理区域。TCloudFinanceZone交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 查询地域列表 查看完成的地域列表。

## 2. 通信协议

TCloudFinanceZone API 的所有接口均通过 HTTPS 进行通信，提供高安全性的通信通道。

## 3. 请求方法

支持的 HTTP 请求方法:

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

- application/json（推荐），必须使用 TC3-HMAC-SHA256 签名方法。
- application/x-www-form-urlencoded，必须使用 HmacSHA1 或 HmacSHA256 签名方法。
- multipart/form-data（仅部分接口支持），必须使用 TC3-HMAC-SHA256 签名方法。

GET 请求的请求包大小不得超过 32 KB。POST 请求使用签名方法为 HmacSHA1、HmacSHA256 时不得超过 1 MB。POST 请求使用签名方法为 TC3-HMAC-SHA256 时支持 10 MB。

## 4. 字符编码

均使用UTF-8编码。

# 返回结果

## 正确返回结果

以云服务器的接口查看实例状态列表 (DescribeInstancesStatus) 2017-03-12 版本为例，若调用成功，其可能的返回如下为：

```
{
    "Response": {
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```

- Response 及其内部的 RequestId 是固定的字段，无论请求成功与否，只要 API 处理了，则必定会返回。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系我们，并提供该 ID 来解决问题。
- 除了固定的字段外，其余均为具体接口定义的字段，不同的接口所返回的字段参见接口文档中的定义。此例中的 TotalCount 和 InstanceStatusSet 均为 DescribeInstancesStatus 接口定义的字段，由于调用请求的用户暂时还没有云服务器实例，因此 TotalCount 在此情况下的返回值为 0， InstanceStatusSet 列表为空。

## 错误返回结果

若调用失败，其返回值示例如下为：

```
{
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please check your signature is cor
rect."
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

- Error 的出现代表着该请求调用失败。Error 字段连同其内部的 Code 和 Message 字段在调用失败时是必定返回的。
- Code 表示具体出错的错误码，当请求出错时可以先根据该错误码在公共错误码和当前接口对应的错误码列表里面查找对应原因和解决方案。

- Message 显示出了这个错误发生的具体原因，随着业务发展或体验优化，此文本可能会经常保持变更或更新，用户不应依赖这个返回值。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系我们，并提供该 ID 来解决问题。

# 公共错误码

返回结果中如果存在 Error 字段，则表示调用 API 接口失败。 Error 中的 Code 字段表示错误码，所有业务都可能出现的错误码为公共错误码，下表列出了公共错误码。

| 错误码 | 错误描述 |
| --- | --- |
| AuthFailure.InvalidSecretId | 密钥非法（不是云 API 密钥类型）。 |
| AuthFailure.MFAFailure | MFA 错误。 |
| AuthFailure.SecretIdNotFound | 密钥不存在。 |
| AuthFailure.SignatureExpire | 签名过期。 |
| AuthFailure.SignatureFailure | 签名错误。 |
| AuthFailure.TokenFailure | token 错误。 |
| AuthFailure.UnauthorizedOperation | 请求未 CAM 授权。 |
| DryRunOperation | DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。 |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidAction | 接口不存在。 |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| LimitExceeded | 超过配额限制。 |
| MissingParameter | 缺少参数错误。 |
| NoSuchVersion | 接口版本不存在。 |
| RequestLimitExceeded | 请求的次数超过了频率限制。 |
| ResourceInUse | 资源被占用。 |
| ResourceInsufficient | 资源不足。 |

| 错误码 | 错误描述 |
| --- | --- |
| ResourceNotFound | 资源不存在。 |
| ResourceUnavailable | 资源不可用。 |
| UnauthorizedOperation | 未授权操作。 |
| UnknownParameter | 未知参数错误。 |
| UnsupportedOperation | 操作不支持。 |
| UnsupportedProtocol | http(s)请求协议错误，只支持 GET 和 POST 请求。 |
| UnsupportedRegion | 接口不支持所传地域。 |

# 公共参数

公共参数是用于标识用户和接口鉴权目的的参数，如非必要，在每个接口单独的接口文档中不再对这些参数进行说明，但每次请求均需要携带这些参数，才能正常发起请求。

## 签名方法 v3

使用 TC3-HMAC-SHA256 签名方法时，公共参数需要统一放到 HTTP Header 请求头部中，如下：

| 参数名称 | 类型 | 必选 | 描述 |
| --- | --- | --- | --- |
| X-TC-Action | String | 是 | 操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。 |
| X-TC-Region | String | 是 | 地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。 |
| X-TC-Timestamp | Integer | 是 | 当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。 |
| X-TC-Version | String | 是 | 操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。 |
| Authorization | String | 是 | HTTP 标准身份认证头部字段，例如：<br>TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024<br>其中，<br>- TC3-HMAC-SHA256：签名方法，目前固定取该值；<br>- Credential：签名凭证，AKIDEXAMPLE 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service为产品名，必须与调用的产品域名一致，例如cvm；<br>- SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部；<br>- Signature：签名摘要。 |
| X-TC-Token | String | 否 | 临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token。 |

## 签名方法 v1

使用 HmacSHA1 和 HmacSHA256 签名方法时，公共参数需要统一放到请求串中，如下

| 参数名称 | 类型 | 必选 | 描述 |
| --- | --- | --- | --- |
| Action | String | 是 | 操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。 |
| Region | String | 是 | 地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。 |

| 参数名称 | 类型 | 必选 | 描述 |
| --- | --- | --- | --- |
| Timestamp | Integer | 是 | 当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。 |
| Nonce | Integer | 是 | 随机正整数，与 Timestamp 联合起来，用于防止重放攻击。 |
| SecretId | String | 是 | 在云API密钥上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey ，而 SecretKey 会用来生成请求签名 Signature。 |
| Signature | String | 是 | 请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见接口鉴权文档。 |
| Version | String | 是 | 操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。 |
| SignatureMethod | String | 否 | 签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。 |
| Token | String | 否 | 临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token 。 |

# 地域列表

地域（Region）是指物理的数据中心的地理区域。TCloudFinanceZone交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 查询地域列表 查看完成的地域列表。

# COS数据导入任务相关接口
# 预览cos导入信息

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于预览cos导入信息

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-18 01:07:04。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：SearchCosRechargeInfo |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题 ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| LogsetId | 是 | 否 | String | 日志集ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Name | 是 | 否 | String | 投递任务名称<br>示例值： cos导入任务 |
| Bucket | 是 | 否 | String | COS存储桶，详见产品支持的存储桶命名规范。<br>示例值： examplebucket-1-1250000000 |
| BucketRegion | 是 | 否 | String | COS存储桶所在地域，详见产品支持的地域列表。<br>示例值： region |
| Prefix | 否 | 否 | String | COS文件所在文件夹的前缀。默认为空，投递存储桶下所有的文件。<br>示例值： var/log |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Compress | 否 | 否 | String | 压缩模式: "", "gzip", "lzop", "snappy"; 默认""<br>示例值： gzip |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Data | Array of String | 匹配到的存储桶下的某个文件的前几行数据<br>示例值： [{"metric":{"code":"InternalError","region":"region","service_name":"service1","url":"api1"},"values":[[1735040190,"100"],[1735040205,"91"],[1735040310,"98"],[1735040325,"85"],[1735040340,"76"],[1735040730,"100"]]}] |
| Sum | Uint64 | 匹配到的存储桶下的文件个数<br>示例值： 1 |
| Path | String | 当前预览文件路径<br>示例值： https://bucket1.region.demo.com/csv/g.gz |
| Msg | String | 预览获取数据失败原因<br>示例值： 授权失败，请确认授权 |
| Status | Int64 | 状态<br>示例值： 0 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |

| 错误码 | 描述 |
| --- | --- |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# Kafka协议消费相关接口
# 关闭Kafka协议消费

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

关闭Kafka协议消费

默认接口请求频率限制：20次/秒。

接口更新时间：2024-12-28 02:26:44。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CloseKafkaConsumer |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| FromTopicId | 是 | 否 | String | CLS对应的topic标识<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| AuthFailure | CAM签名/鉴权错误。 |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |

# 获取Kafka协议消费信息

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

获取Kafka协议消费信息

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 00:28:56。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeKafkaConsumer |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| FromTopicId | 是 | 否 | String | CLS对应topic标识<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Status | Bool | Kafka协议消费打开状态，true表示消费打开，false表示消费关闭<br>示例值： false |
| TopicID | String | KafkaConsumer 消费时使用的Topic参数<br>示例值： out-a5ce0c9c-0690-44a5-bc79-5f2190626bd0 |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Compression | Int64 | 压缩方式[0:NONE；2:SNAPPY；3:LZ4]<br>示例值： 2 |
| ConsumerContent | KafkaConsumerContent | kafka协议消费数据格式<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| AuthFailure | CAM签名/鉴权错误。 |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |

# 修改Kafka协议消费信息

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

修改Kafka协议消费信息

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 05:31:37。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyKafkaConsumer |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| FromTopicId | 是 | 否 | String | 日志主题ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Compression | 否 | 否 | Int64 | 压缩方式[0:NONE；2:SNAPPY；3:LZ4]<br>示例值： 2 |
| ConsumerContent | 否 | 否 | KafkaConsumerContent | kafka协议消费数据格式<br>示例值：查看 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| AuthFailure | CAM签名/鉴权错误。 |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |

# 打开Kafka协议消费

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

打开Kafka协议消费功能

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 19:55:54。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：OpenKafkaConsumer |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| FromTopicId | 是 | 否 | String | CLS控制台创建的TopicId<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Compression | 否 | 否 | Int64 | 压缩方式[0:NONE；2:SNAPPY；3:LZ4]<br>示例值： 0 |
| ConsumerContent | 否 | 否 | KafkaConsumerContent | kafka协议消费数据格式<br>示例值：查看 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| TopicID | String | KafkaConsumer 消费时使用的Topic参数<br>示例值： out-57f5808c-4a55-11eb-b378-0242ac130002 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| AuthFailure | CAM签名/鉴权错误。 |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |

# 主题分区相关接口
# 获取分区列表

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于获取主题分区列表。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-17 19:21:35。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribePartitions |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Partitions | Array of PartitionInfo | 分区列表<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| ResourceNotFound | 资源不存在。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 合并分区

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于合并一个读写态的主题分区，合并时指定一个主题分区 ID，日志服务会自动合并范围右相邻的分区。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 04:47:32。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：MergePartition |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 427bba81-e149-4a8b-bd0d-b2a412520d7b |
| PartitionId | 是 | 否 | Int64 | 合并的PartitionId（找到下一个分区InclusiveBeginKey与入参PartitionId对应的ExclusiveEndKey相等，且找到的分区必须是读写分区（Staus:readwrite），入参PartitionId与找到的PartitionId设置为只读分区（Status:readonly）,再新建一个新的读写分区）。获取分区列表 入参PartitionId只能是读写分区（Status的值有readonly，readwrite），且能找到入参PartitionId的下一个可读写分区（找到下一个分区InclusiveBeginKey与入参PartitionId对应的ExclusiveEndKey相等）； 入参PartitionId不能是最后一个分区（PartitionId的ExclusiveEndKey不能是ffffffffffffffffffffffffffffffff）； topic的分区数量是有限制的（默认50个），合并之后不能超过最大分区，否则不能合并。<br>示例值： 2 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Partitions | Array of PartitionInfo | 合并结果集<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| UnsupportedOperation | 操作不支持。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| ResourceNotFound.PartitionNotExist | 分区不存在。 |
| LimitExceeded.Partition | 分区超过限制。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 分裂主题分区

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于分裂主题分区

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-16 00:55:20。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：SplitPartition |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 427bba81-e149-4a8b-bd0d-b2a412520d7b |
| PartitionId | 是 | 否 | Int64 | 待分裂分区ID<br>示例值： 1 |
| SplitKey | 否 | 否 | String | 分区切分的哈希key的位置，只在Number=2时有意义<br>示例值： key |
| Number | 否 | 否 | Int64 | 分区分裂个数(可选)，默认等于2<br>示例值： 2 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Partitions | Array of PartitionInfo | 分裂结果集<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| UnsupportedOperation | 操作不支持。 |
| ResourceNotFound | 资源不存在。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| ResourceNotFound.PartitionNotExist | 分区不存在。 |
| LimitExceeded.Partition | 分区超过限制。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |
| FailedOperation.TopicClosed | 日志主题已关闭。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 仪表盘相关接口
# 获取仪表盘

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于获取仪表盘

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:59:37。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeDashboards |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Offset | 否 | 否 | Int64 | 分页的偏移量，默认值为0。<br>示例值： 1 |
| Limit | 否 | 否 | Int64 | 分页单页限制数目，默认值为20，最大值100。<br>示例值： 5 |
| Filters | 否 | 否 | Array of Filter | dashboardId 按照【仪表盘id】进行过滤，类型：String， 必选：<br>否。 dashboardName 按照【仪表盘名字】进行模糊搜索过滤，类型：String，必选：<br>否。 dashboardRegion 按照【仪表盘地域】进行过滤，为了兼容老的仪表 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | 盘，通过云API创建的仪表盘没有地域属性，类型：String，必选：否。 tagKey 按照【标签键】进行过滤，类型：String，必选：否。 tag:tagKey 按照【标签键值对】进行过滤。tagKey使用具体的标签键进行替换，类型：String，必选：否，使用请参考示例2。 每次请求的Filters的上限为10，Filter.Values的上限为100。<br>示例值：查看 |
| TopicIdRegionFilter | 否 | 否 | Array of TopicIdAndRegion | 按照topicId和regionId过滤。<br>示例值：查看 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| TotalCount | Int64 | 仪表盘的数量<br>示例值： 10 |
| DashboardInfos | Array of DashboardInfo | 仪表盘详细明细<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| LimitExceeded | 超过配额限制。 |

| 错误码 | 描述 |
| --- | --- |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| LimitExceeded.Tag | tag超过限制。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 告警策略相关接口
# 创建告警策略

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于创建告警策略。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-08-14 03:00:48。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateAlarm |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Name | 是 | 否 | String | 告警策略名称<br>示例值： alarm_name |
| AlarmTargets | 是 | 否 | Array of AlarmTarget | 监控对象列表。<br>示例值：查看 |
| MonitorTime | 是 | 否 | MonitorTime | 监控任务运行时间点。<br>示例值：查看 |
| Condition | 是 | 否 | String | 触发条件 注意：Condition和AlarmLevel是一组配置，MultiConditions是另一组配置，2组配置互斥。 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | 示例值： $1.count > 10 |
| TriggerCount | 是 | 否 | Int64 | 持续周期。持续满足触发条件TriggerCount个周期后，再进行告警；最小值为1，最大值为2000。<br>示例值： 1 |
| AlarmPeriod | 是 | 否 | Int64 | 告警重复的周期。单位是分钟。取值范围是0~1440。<br>示例值： 0 |
| AlarmNoticeIds | 是 | 否 | Array of String | 关联的告警通知模板列表。<br>示例值： ["notice-b02af159-f67b-44c3-877c-a55343f874c3"] |
| Status | 否 | 否 | Bool | 是否开启告警策略。默认值为true，true表示开启告警策略，false表示关闭告警策略。<br>示例值： true |
| Enable | 否 | 否 | Bool | 该参数已废弃，请使用Status参数控制是否开启告警策略。<br>示例值： true |
| MessageTemplate | 否 | 否 | String | 用户自定义告警内容<br>示例值： content |
| CallBack | 否 | 否 | CallBackInfo | 用户自定义回调<br>示例值：查看 |
| Analysis | 否 | 否 | Array of AnalysisDimensional | 多维分析<br>示例值：查看 |
| GroupTriggerStatus | 否 | 否 | Bool | 分组触发状态。 默认值false<br>示例值： true |
| GroupTriggerCondition | 否 | 否 | Array of String | 分组触发条件。<br>示例值： ["$1.conut"] |
| Tags | 否 | 否 | Array of Tag | 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告警策略。 最大支持10个标签键值对，并且不能有重复的键值对。<br>示例值：查看 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| MonitorObjectType | 否 | 否 | Uint64 | 监控对象类型。0:执行语句共用监控对象; 1:每个执行语句单独选择监控对象。 不填则默认为0。 当值为1时，AlarmTargets元素个数不能超过10个，AlarmTargets中的Number必须是从1开始的连续正整数，不能重复。<br>示例值： 0 |
| Classifications | 否 | 否 | Array of AlarmClassification | 告警附加分类信息列表。 Classifications元素个数不能超过20个。 Classifications元素的Key不能为空，不能重复，长度不能超过50个字符，符合正则 ^a-z$。 Classifications元素的Value长度不能超过200个字符。<br>示例值：查看 |
| AlarmLevel | 否 | 否 | Uint64 | 告警级别 0:警告(Warn); 1:提醒(Info); 2:紧急 (Critical)。 注意: 不填则默认为0。 Condition和AlarmLevel是一组配置，MultiConditions是另一组配置，2组配置互斥。<br>示例值： 0 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| AlarmId | String | 告警策略ID。<br>示例值： alarm-479dec34-4a59-11eb-b378-0242ac130002 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| InvalidParameter.AlarmConflict | 告警策略已经存在。 |
| ResourceNotFound.AlarmNoticeNotExist | 告警策略通知模板不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| OperationDenied.AlarmNotSupportForSearchLow | 低频不支持告警。 |

# 创建告警通知模板

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

该接口用户创建告警通知模板。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-08-14 09:48:12。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateAlarmNotice |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Name | 是 | 否 | String | 通知渠道组名称。<br>示例值： notice_name |
| Type | 否 | 否 | String | 【简易模式】（简易模式/告警模式二选一，分别配置相应参数）需要发送通知的告警类型。可选值： Trigger - 告警触发 Recovery - 告警恢复 All - 告警触发和告警恢复<br>示例值： All |
| NoticeReceivers | 否 | 否 | Array of NoticeReceiver | 【简易模式】（简易模式/告警模式二选一，分别配置相应参数）通知接收对象。<br>示例值：查看 |
| WebCallbacks | 否 | 否 | Array of WebCallback | 【简易模式】（简易模式/告警模式二选一，分别配置相应参数）接口回调信息（包括企 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | 业微信、钉钉、飞书）。<br>示例值：查看 |
| Tags | 否 | 否 | Array of Tag | 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通知渠道组。最大支持50个标签键值对，并且不能有重复的键值对。<br>示例值：查看 |
| JumpDomain | 否 | 否 | String | 查询数据链接。http:// 或者 https:// 开头，不能/结尾<br>示例值： https://console.demo.com |
| NoticeRules | 否 | 否 | Array of NoticeRule | 【高级模式】（简易模式/告警模式二选一，分别配置相应参数）通知规则。<br>示例值：查看 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| AlarmNoticeId | String | 告警模板ID<br>示例值： notice-f0454407-1bb2-48c7-b224-653d4b236d14 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| InvalidParameter.AlarmNoticeConflict | 告警策略通知模板已经存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |

| 错误码 | 描述 |
| --- | --- |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 删除告警策略

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于删除告警策略。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-09 20:42:54。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteAlarm |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| AlarmId | 是 | 否 | String | 告警策略ID。<br>示例值： 5944e926-ae21-4fa1-8af6-e1539b577482 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.AlarmNotExist | 告警策略不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 删除告警通知模板

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

该接口用于删除告警通知模板

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-09 20:48:06。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteAlarmNotice |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| AlarmNoticeId | 是 | 否 | String | 告警通知模板<br>示例值： notice-d7286784-46d9-4be3-b7ba-2c464e6a5ad1 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| FailedOperation.BindedAlarm | 告警策略通知模板已经绑定到了某个告警策略上。 |
| ResourceNotFound.AlarmNoticeNotExist | 告警策略通知模板不存在。 |
| OperationDenied.NoticeHasAlarm | 通知模版已绑定告警，无法删除。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取告警通知模板列表

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

该接口用于获取告警通知模板列表

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-22 19:33:59。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeAlarmNotices |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Filters | 否 | 否 | Array of Filter | name 按照【通知渠道组名称】进行过滤。 类型：String 必选：否 alarmNoticeId 按照【通知渠道组ID】进行过滤。 类型：String 必选：否 uid 按照【接收用户ID】进行过滤。 类型：String 必选：否 groupId 按照【接收用户组ID】进行过滤。 类型：String 必选：否 deliverFlag 按照【投递状态】进行过滤。 类型：String 必选：否 可选值： "1":未启用, "2": 已启用, "3":投递异常 每次请求的Filters的上限为10，Filter.Values的上限为5。<br>示例值：查看 |
| Offset | 否 | 否 | Int64 | 分页的偏移量，默认值为0。<br>示例值： 1 |
| Limit | 否 | 否 | Int64 | 分页单页限制数目，默认值为20，最大值100。<br>示例值： 1 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| AlarmNotices | Array of AlarmNotice | 告警通知模板列表。<br>示例值：查看 |
| TotalCount | Int64 | 符合条件的告警通知模板总数。<br>示例值： 0 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取告警策略列表

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于获取告警策略。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-31 01:59:40。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeAlarms |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Filters | 否 | 否 | Array of Filter | name 按照【告警策略名称】进行过滤。 类型：String 必选：否 alarmId 按照【告警策略ID】进行过滤。 类型：String 必选：否 topicId 按照【监控对象的日志主题ID】进行过滤。 类型：String 必选：否 enable 按照【启用状态】进行过滤。 类型：String 备注：enable参数值范围: 1, t, T, TRUE, true, True, 0, f, F, FALSE, false, False。 其它值将返回参数错误信息. 必选：否 每次请求的Filters的上限为10，Filter.Values的上限为5。<br>示例值：查看 |
| Offset | 否 | 否 | Int64 | 分页的偏移量，默认值为0。<br>示例值： 1 |
| Limit | 否 | 否 | Int64 | 分页单页限制数目，默认值为20，最大值100。<br>示例值： 1 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Alarms | Array of AlarmInfo | 告警策略列表。<br>示例值：查看 |
| TotalCount | Int64 | 符合查询条件的告警策略数目。<br>示例值： 0 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 告警历史记录

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

告警历史记录

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-14 19:59:26。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeAlertRecordHistory |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Filters | 否 | 否 | Array of Filter | alertId：按照告警策略ID进行过滤。类型：String 必选：否 topicId：按照监控对象ID进行过滤。类型：String 必选：否 status：按照告警状态进行过滤。类型：String 必选：否，0代表未恢复，1代表已恢复，2代表已失效 alarmLevel：按照告警等级进行过滤。类型：String 必选：否，0代表警告，1代表提醒，2代表紧急 每次请求的Filters的上限为10，Filter.Values的上限为100。<br>示例值：查看 |
| From | 是 | 否 | Uint64 | 查询时间范围启始时间，毫秒级unix时间戳<br>示例值： 1619063300000 |
| To | 是 | 否 | Uint64 | 查询时间范围结束时间，毫秒级unix时间戳<br>示例值： 1619581700000 |
| Offset | 是 | 否 | Int64 | 分页的偏移量，默认值为0。<br>示例值： 0 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Limit | 是 | 否 | Int64 | 分页单页限制数目，最大值100。<br>示例值： 10 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| TotalCount | Int64 | 告警历史总数<br>示例值： 1 |
| Records | Array of AlertHistoryRecord | 告警历史详情<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取告警任务历史记录

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于获取告警任务历史

默认接口请求频率限制：100次/秒。

接口更新时间：2025-01-17 19:54:57。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：GetAlarmLog |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| From | 是 | 否 | Int64 | 要查询的日志的起始时间，Unix时间戳，单位ms<br>示例值： 1702569600000 |
| To | 是 | 否 | Int64 | 要查询的日志的结束时间，Unix时间戳，单位ms<br>示例值： 1702655999999 |
| Query | 是 | 否 | String | 查询语句，语句长度最大为1024<br>示例值： alert_id:notice-dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Limit | 否 | 否 | Int64 | 单次查询返回的执行详情条数，最大值为1000<br>示例值： 100 |
| Context | 否 | 否 | String | 透传上次接口返回的Context值，可获取后续更多日志，总计最多可获取1万条原始日志，过期时间1小时。 注意： 透传该参数时，请勿修改除该参数外的其它参数 仅当检索分析语句(Query)不包含SQL时有效，SQL获取后续结果参考SQL LIMIT语法<br>示例值： context |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Sort | 否 | 否 | String | 原始日志是否按时间排序返回；可选值：asc(升序)、desc(降序)，默认为 desc 注意： 仅当检索分析语句(Query)不包含SQL时有效 SQL结果排序方式参考SQL ORDER BY语法<br>示例值： asc |
| UseNewAnalysis | 否 | 否 | Bool | true：代表使用新的检索结果返回方式，输出参数AnalysisRecords和Columns有效； false：代表使用老的检索结果返回方式，输出AnalysisResults和ColNames有效； 两种返回方式在编码格式上有少量区别，建议使用true。<br>示例值： false |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Context | String | 加载后续内容的Context<br>示例值： eda234f5-e060-483a-b1a1-1f76208dcb2b:eda234f5-e060-483a-b1a1-1f76208dcb2b@30@1732020030000000@1732020930000000:10 |
| ListOver | Bool | 指定时间范围内的告警执行详情是否完整返回<br>示例值： false |
| Analysis | Bool | 返回的结果是否为SQL分析结果<br>示例值： 1 |
| ColNames | Array of String | 分析结果的列名，如果Query语句有SQL查询，则返回查询字段的列名； 否则为空。<br>示例值： ["time","pv","uv"] |
| Results | Array of LogInfo | 执行详情查询结果。 当Query字段无SQL语句时，返回查询结果。 当Query字段有SQL语句时，可能返回null。<br>示例值：查看 |
| AnalysisResults | Array of LogItems | 执行详情统计分析结果。当Query字段有SQL语句时，返回SQL统计结果，否则可能返回null。<br>示例值：查看 |
| AnalysisRecords | Array of String | 执行详情统计分析结果；UseNewAnalysis为true有效。<br>示例值： ["status:200"] |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Columns | Array of Column | 分析结果的列名， UseNewAnalysis为true有效<br>示例值：查看 |
| Query | String | 返回语法优化后的语句（入参如有QueryOptimize 且为 1 时返回，其他情况返回空字符串）<br>示例值： alert_id:notice-57f5808c-4a55-11eb-b378-0242ac130002 |
| SamplingRate | Float | 当前系统使用的采样率，入参如有SamplingRate时生效（主要是当客户输入0时，返回真实后台的AutoSamplingRate值）<br>示例值： 0.1 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| FailedOperation.SearchTimeout | 查询超时。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |
| FailedOperation.QueryError | 查询语句运行失败。 |
| FailedOperation.InvalidContext | 检索游标已失效或不存在。 |
| FailedOperation.GetlogReachLimit | 检索日志触发最大条数限制。 |
| FailedOperation.SyntaxError | 查询语句解析错误。 |
| LimitExceeded.LogSearch | 并发查询超过限制，单topic并发最大值15。 |

| 错误码 | 描述 |
| --- | --- |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 修改告警策略

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于修改告警策略。需要至少修改一项有效内容。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-21 23:36:17。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyAlarm |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| AlarmId | 是 | 否 | String | 告警策略ID。<br>示例值： alarm-bea71170-4a59-11eb-b378-0242ac130002 |
| Name | 否 | 否 | String | 告警策略名称<br>示例值： alarm_name |
| MonitorTime | 否 | 否 | MonitorTime | 监控任务运行时间点。<br>示例值：查看 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Condition | 否 | 否 | String | 触发条件。 注意：Condition和AlarmLevel是一组配置，MultiConditions是另一组配置，2组配置互斥。<br>示例值： $1.source='10.0.0.1' |
| TriggerCount | 否 | 否 | Int64 | 持续周期。持续满足触发条件TriggerCount个周期后，再进行告警；最小值为1，最大值为2000。<br>示例值： 1 |
| AlarmPeriod | 否 | 否 | Int64 | 告警重复的周期。单位是分钟。取值范围是0~1440。<br>示例值： 0 |
| AlarmNoticeIds | 否 | 否 | Array of String | 关联的告警通知模板列表。<br>示例值：<br>["notice-479dec34-4a59-11eb-b378-0242ac130002"] |
| AlarmTargets | 否 | 否 | Array of AlarmTarget | 监控对象列表。<br>示例值：查看 |
| Status | 否 | 否 | Bool | 是否开启告警策略。true表示开启告警策略，false表示关闭告警策略。<br>示例值： true |
| Enable | 否 | 否 | Bool | 该参数已废弃，请使用Status参数控制是否开启告警策略。<br>示例值： true |
| MessageTemplate | 否 | 否 | String | 用户自定义告警内容<br>示例值： data |
| CallBack | 否 | 否 | CallBackInfo | 用户自定义回调<br>示例值：查看 |
| Analysis | 否 | 否 | Array of AnalysisDimensional | 多维分析<br>示例值：查看 |
| GroupTriggerStatus | 否 | 否 | Bool | 分组触发状态。true：开启，false：关闭（默认）<br>示例值： true |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| GroupTriggerCondition | 否 | 否 | Array of String | 分组触发条件。<br>示例值： ["$1.conut"] |
| Tags | 否 | 否 | Array of Tag | 标签描述列表，通过指定该参数可以同时绑定标签到相应的告警策略。最大支持10个标签键值对，并且不能有重复的键值对。<br>示例值：查看 |
| MonitorObjectType | 否 | 否 | Uint64 | 监控对象类型。0:执行语句共用监控对象; 1:每个执行语句单独选择监控对象。 当值为1时，AlarmTargets元素个数不能超过10个，AlarmTargets中的Number必须是从1开始的连续正整数，不能重复。<br>示例值： 0 |
| Classifications | 否 | 是 | Array of AlarmClassification | 告警附加分类信息列表。 Classifications元素个数不能超过20个。 Classifications元素的Key不能为空，不能重复，长度不能超过50个字符，符合正则 ^a-z$。 Classifications元素的Value长度不能超过200个字符。<br>示例值：查看 |
| AlarmLevel | 否 | 否 | Uint64 | 告警级别。 0:警告(Warn);1:提醒(Info);2:紧急 (Critical) 注意: Condition和AlarmLevel是一组配置，MultiConditions是另一组配置，2组配置互斥。<br>示例值： 0 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| InvalidParameter.AlarmConflict | 告警策略已经存在。 |
| ResourceNotFound.AlarmNotExist | 告警策略不存在。 |
| ResourceNotFound.AlarmNoticeNotExist | 告警策略通知模板不存在。 |
| FailedOperation.InvalidAlarm | 该告警策略状态异常，请检查下日志主题ID是否都存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| OperationDenied.AlarmNotSupportForSearchLow | 低频不支持告警。 |

# 修改告警通知模板

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

该接口用于修改告警通知模板。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-17 19:34:11。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyAlarmNotice |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Name | 否 | 否 | String | 告警模板名称。<br>示例值： Demo访问日志通知渠道组 |
| Type | 否 | 否 | String | 通知类型。可选值： Trigger - 告警触发 Recovery - 告警恢复 All - 告警触发和告警恢复<br>示例值： All |
| NoticeReceivers | 否 | 否 | Array of NoticeReceiver | 告警模板接收者信息。<br>示例值：查看 |
| WebCallbacks | 否 | 否 | Array of WebCallback | 接口回调信息（包括企业微信等）。<br>示例值：查看 |
| AlarmNoticeId | 是 | 否 | String | 告警通知模板ID。<br>示例值： notice-5b2ea996-4dae-47cc- |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | bbad-5b12888b4c89 |
| Tags | 否 | 否 | Array of Tag | 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通知渠道组。最大支持10个标签键值对，并且不能有重复的键值对。<br>示例值：查看 |
| JumpDomain | 否 | 否 | String | 调用链接域名。http:// 或者 https:// 开头，不能/结尾<br>示例值： https://console.cloud.demo.com |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| InvalidParameter.AlarmNoticeConflict | 告警策略通知模板已经存在。 |
| ResourceNotFound.AlarmNoticeNotExist | 告警策略通知模板不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |

| 错误码 | 描述 |
| --- | --- |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 对象存储数据导入任务相关接口
# 创建对象存储导入任务

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于创建对象存储导入任务

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:22:16。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateCosRecharge |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题 ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| LogsetId | 是 | 否 | String | 日志集ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Name | 是 | 否 | String | 投递任务名称<br>示例值： cos导入 |
| Bucket | 是 | 否 | String | COS存储桶，详见产品支持的存储桶命名规范。<br>示例值： examplebucket-1250000000 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| BucketRegion | 是 | 否 | String | COS存储桶所在地域，详见产品支持的地域列表。<br>示例值： region |
| Prefix | 否 | 否 | String | COS文件所在文件夹的前缀。默认为空，投递存储桶下所有的文件。<br>示例值： var/log |
| LogType | 是 | 否 | String | 采集的日志类型，json_log代表json格式日志，delimiter_log代表分隔符格式日志，minimalist_log代表单行全文； 默认为minimalist_log<br>示例值： minimalist_log |
| Compress | 否 | 否 | String | 压缩格式，支持无压缩，gzip，lzop，snappy。默认值为无压缩。<br>示例值： gzip |
| ExtractRuleInfo | 否 | 否 | ExtractRuleInfo | 提取规则，如果设置了ExtractRule，则必须设置LogType<br>示例值：查看 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Id | String | COS导入任务id<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidParameter | 参数错误。 |

| 错误码 | 描述 |
| --- | --- |
| MissingParameter | 缺少参数错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取对象存储导入配置

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于获取对象存储导入配置

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:58:00。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeCosRecharges |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题 ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Status | 否 | 否 | Uint64 | 状<br>态 status 0: created, 1: running, 2: pause, 3: finished, 4: failed。<br>示例值： 1 |
| Enable | 否 | 否 | Uint64 | 是否启用: 0： 未启用 ， 1：启用<br>示例值： 1 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Data | Array of CosRechargeInfo | 见: CosRechargeInfo 结构描述<br>示例值：查看 |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidParameter | 参数错误。 |
| InvalidParameter.TopicConflict | 指定日志集下已经有同名的日志主题。 |
| LimitExceeded | 超过配额限制。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| ResourceNotFound | 资源不存在。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |

# 修改对象存储导入任务

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于修改对象存储导入任务

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-17 19:37:55。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyCosRecharge |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Id | 是 | 否 | String | COS导入配置Id<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TopicId | 是 | 否 | String | 日志主题Id<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Name | 否 | 否 | String | cos导入任务名称<br>示例值： 导入任务1 |
| Enable | 否 | 否 | Uint64 | 是否启用: 0： 未启用 ， 1：启用<br>示例值： 1 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |

# 投递Ckafka任务相关接口
# 创建投递Ckafka任务

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

创建投递Ckafka任务

默认接口请求频率限制：20次/秒。

接口更新时间：2025-08-14 11:57:32。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateConsumer |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 投递任务绑定的日志主题 ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| NeedContent | 否 | 否 | Bool | 是否投递日志的元数据信息，默认为 true。 当NeedContent为true时：字段Content有效。 当NeedContent为false时：字段Content无效。<br>示例值： false |
| Content | 否 | 否 | ConsumerContent | 如果需要投递元数据信息，元数据信息的描述<br>示例值：查看 |
| Ckafka | 是 | 否 | Ckafka | CKafka的描述<br>示例值：查看 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Compression | 否 | 否 | Int64 | 投递时压缩方式，取值0，2，3。[0：NONE；2：SNAPPY；3：LZ4]<br>示例值： 0 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidParameter | 参数错误。 |
| LimitExceeded | 超过配额限制。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 投递对象存储任务相关接口
# 创建投递规则

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

创建新的投递规则，客户如果使用此接口，需要自行处理CLS对指定bucket的写权限。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:31:37。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateShipper |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 创建的投递规则所属的日志主题ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Bucket | 是 | 否 | String | 创建的投递规则投递的bucket<br>示例值： examplebucket-1250000000 |
| Prefix | 是 | 否 | String | 创建的投递规则投递目录的前缀<br>示例值： var/log |
| ShipperName | 是 | 否 | String | 投递规则的名字<br>示例值： shipper_name |
| Interval | 否 | 否 | Uint64 | 投递的时间间隔，单位 秒，默认300，范围 300-900<br>示例值： 300 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| MaxSize | 否 | 否 | Uint64 | 投递的文件的最大值，单位 MB，默认100，范围 5-256<br>示例值： 100 |
| FilterRules | 否 | 否 | Array of FilterRuleInfo | 投递日志的过滤规则，匹配的日志进行投递，各rule之间是and关系，最多5个，数组为空则表示不过滤而全部投递<br>示例值：查看 |
| Partition | 否 | 否 | String | 投递日志的分区规则，支持strftime的时间格式表示<br>示例值： %Y%m%d |
| Compress | 否 | 否 | CompressInfo | 投递日志的压缩配置<br>示例值：查看 |
| Content | 否 | 否 | ContentInfo | 投递日志的内容格式配置<br>示例值：查看 |
| FilenameMode | 否 | 否 | Uint64 | 投递文件命名配置，0：随机数命名，1：投递时间命名，默认0（随机数命名）<br>示例值： 0 |
| CustomUin | 否 | 否 | Uint64 | 跨账户uin，用于支持跨账户bucket投递<br>示例值： 1670641097 |
| StartTime | 否 | 否 | Int64 | 投递数据范围的开始时间点，不能超出日志主题的生命周期起点。如果用户不填写，默认为用户新建投递任务的时间。<br>示例值： 1713410880 |
| EndTime | 否 | 否 | Int64 | 投递数据范围的结束时间点，不能填写未来时间。如果用户不填写，默认为持续投递，即无限。<br>示例值： 1713497280 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| ShipperId | String | 投递规则ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| InvalidParameter.ShipperConflict | 投递规则命名冲突。 |
| LimitExceeded.Shipper | 投递规则超出限制。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 删除投递规则

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

删除投递规则

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:37:52。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteShipper |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| ShipperId | 是 | 否 | String | 投递规则ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| ResourceNotFound.ShipperNotExist | 投递规则不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取投递任务列表

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

获取投递任务列表

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 01:24:03。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeShipperTasks |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| ShipperId | 是 | 否 | String | 投递规则ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| StartTime | 是 | 否 | Int64 | 查询的开始时间戳，支持最近3天的查询， 毫秒。 StartTime必须小于EndTime<br>示例值： 978653222 |
| EndTime | 是 | 否 | Int64 | 查询的结束时间戳， 毫秒。 StartTime必须小于EndTime<br>示例值： 978653222 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Tasks | Array of ShipperTaskInfo | 投递任务列表<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| ResourceNotFound.ShipperNotExist | 投递规则不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取投递规则

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

获取投递规则信息列表

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 01:23:22。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeShippers |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Filters | 否 | 否 | Array of Filter | shipperName：按照【投递规则名称】进行过滤。 类型：String。 必选：否 shipperId：按照【投递规则ID】进行过滤。 类型：String。 必选：否 topicId：按照【日志主题】进行过滤。 类型：String。 必选：否 taskStatus 按照【任务运行状态】进行过滤。 支持0：停止，1：运行中，2：异常 类型：String 必选：否 每次请求的Filters的上限为10，Filter.Values的上限为10。<br>示例值：查看 |
| Offset | 否 | 否 | Uint64 | 分页的偏移量，默认值为0<br>示例值： 0 |
| Limit | 否 | 否 | Uint64 | 分页单页的限制数目，默认值为20，最大值100<br>示例值： 10 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Shippers | Array of ShipperInfo | 投递规则列表<br>示例值：查看 |
| TotalCount | Uint64 | 本次查询获取到的总数<br>示例值： 1 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| LimitExceeded | 超过配额限制。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 修改投递规则

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

修改现有的投递规则，客户如果使用此接口，需要自行处理CLS对指定bucket的写权限。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 01:44:57。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyShipper |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| ShipperId | 是 | 否 | String | 投递规则ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Bucket | 否 | 否 | String | COS存储桶，详见产品支持的存储桶命名规范。<br>示例值： examplebucket-1250000000 |
| Prefix | 否 | 否 | String | 投递规则投递的新的目录前缀。 仅支持0-9A-Za-z-_/ 最大支持256个字符<br>示例值： var/log |
| Status | 否 | 否 | Bool | 投递规则的开关状态。true：开启投递任务；false：关闭投递任务。<br>示例值： true |
| ShipperName | 否 | 否 | String | 投递规则的名字<br>示例值： shipper_name |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Interval | 否 | 否 | Uint64 | 投递的时间间隔，单位 秒，默认300，范围 300-900<br>示例值： 300 |
| MaxSize | 否 | 否 | Uint64 | 投递的文件的最大值，单位 MB，默认100，范围 5-256<br>示例值： 100 |
| FilterRules | 否 | 否 | Array of FilterRuleInfo | 投递日志的过滤规则，匹配的日志进行投递，各rule之间是and关系，最多5个，数组为空则表示不过滤而全部投递<br>示例值：查看 |
| Partition | 否 | 否 | String | 投递日志的分区规则，支持strftime的时间格式表示<br>示例值： %Y%m%d |
| Compress | 否 | 否 | CompressInfo | 投递日志的压缩配置<br>示例值：查看 |
| Content | 否 | 否 | ContentInfo | 投递日志的内容格式配置<br>示例值：查看 |
| FilenameMode | 否 | 否 | Uint64 | 投递文件命名配置，0：随机数命名，1：投递时间命名。<br>示例值： 0 |
| CustomUin | 否 | 否 | Uint64 | 跨账号uin，用于支持跨账号bucket投递<br>示例值： 1670641097 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| InternalError | 内部错误。 |
| ResourceNotFound | 资源不存在。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| ResourceNotFound.ShipperNotExist | 投递规则不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 重试失败的投递任务

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

重试失败的投递任务

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 01:49:15。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：RetryShipperTask |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| ShipperId | 是 | 否 | String | 投递规则ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TaskId | 是 | 否 | String | 投递任务ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| ResourceNotFound.ShipperNotExist | 投递规则不存在。 |
| FailedOperation.ShipperTaskNotToRetry | 投递任务不允许重试。 |
| ResourceNotFound.ShipperTaskNotExist | 投递任务不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 日志主题相关接口
# 创建日志主题

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于创建日志主题。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:32:22。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateTopic |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| LogsetId | 是 | 否 | String | 日志集ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TopicName | 是 | 否 | String | 日志主题名称<br>示例值： topic_name |
| PartitionCount | 否 | 否 | Int64 | 日志主题分区个数。默认创建1个，最大支持创建10个分区。<br>示例值： 1 |
| Tags | 否 | 否 | Array of Tag | 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10个标签键值对，同一个资源只能绑定到同一个标签键下。<br>示例值：查看 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| AutoSplit | 否 | 否 | Bool | 是否开启自动分裂，默认值为true。true表示开启自动分裂，false表示关闭自动分裂。<br>示例值： true |
| MaxSplitPartitions | 否 | 否 | Int64 | 开启自动分裂后，每个主题能够允许的最大分区数，默认值为50<br>示例值： 50 |
| StorageType | 否 | 否 | String | 日志主题的存储类型，可选值 hot（实时存储）；默认为hot。<br>示例值： hot |
| Period | 否 | 否 | Int64 | 生命周期，单位天；可取值范围1~90。默认30天<br>示例值： 30 |
| SubAssumerName | 否 | 否 | String | 二级产品标识 测试<br>示例值： TKE_AUDIO |
| Describes | 否 | 否 | String | 日志主题描述<br>示例值： A部门日志主题 |
| HotPeriod | 否 | 否 | Uint64 | 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<br>示例值： 7 |
| Encryption | 否 | 否 | Uint64 | 加密相关参数。 支持加密地域并且开白用户可以传此参数，其他场景不能传递该参数。 0或者不传： 不加密 1：kms-cls 云产品秘钥加密<br>示例值： 0 |
| BizType | 否 | 否 | Uint64 | 0: 日志主题topic 1: 时序主题topic<br>示例值： 0 |
| TopicId | 否 | 否 | String | 日志主题ID，格式为：用户自定义部分-用户appid，用户自定义部分仅支持小写字母、数字和-，且不能以-开头和结尾，长度为3至40字符，结尾使用-拼接用户appid<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| IsWebTracking | 否 | 否 | Bool | webtracking开关； false: 关闭 true： 开启<br>示例值： false |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| TopicId | String | 日志主题ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| LimitExceeded | 超过配额限制。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.LogsetNotExist | 指定的日志集不存在。 |
| LimitExceeded.Logset | 日志集数量超出限制。 |
| InvalidParameter.TopicConflict | 指定日志集下已经有同名的日志主题。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| FailedOperation.InvalidPeriod | 离线存储保存时间不得少于7天。 |
| LimitExceeded.Topic | 日志主题数目超过限制。 |

# 删除日志主题

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于删除日志主题。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-10 02:13:42。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteTopic |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： a5ce0c9c-0690-44a5-bc79-5f2190626bd0 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.LogsetNotExist | 指定的日志集不存在。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| OperationDenied.TopicHasDeliverFunction | topic绑定了函数投递。 |
| OperationDenied.TopicHasDataFormTask | topic绑定了数据加工。 |

# 获取日志主题列表

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于获取日志主题列表，支持分页

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 02:44:13。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeTopics |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Filters | 否 | 否 | Array of Filter | topicName 按照【日志主题名称】进行过滤，默认为模糊匹配，可使用 PreciseSearch 参数设置为精确匹配。类型：String。必选：否 logsetName 按照【日志集名称】进行过滤，默认为模糊匹配，可使用 PreciseSearch 参数设置为精确匹配。类型：String。必选：否 topicId 按照【日志主题ID】进行过滤。类型：String。必选：否 logsetId 按照【日志集ID】进行过滤，可通过调用 DescribeLogsets 查询已创建的日志集列表或登录控制台进行查看；也可以调用 CreateLogset 创建新的日志集。类型：String。必选：否 tagKey 按照【标签键】进行过滤。类型：String。必选：否 tag:tagKey 按照【标签键值对】进行过滤。tagKey 使用具体的标签键进行替换，例如 tag:exampleKey。类型：String。必选：否 storageType 按照【日志主题的存储类型】进行过滤。可选值 hot（标准存储），cold（低频存储）类型：String。必选：否 注意：每次请求的 Filters 的上限为10，Filter.Values 的上限为100。 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | 示例值：查看 |
| Offset | 否 | 否 | Int64 | 分页的偏移量，默认值为0。<br>示例值： 10 |
| Limit | 否 | 否 | Int64 | 分页单页限制数目，默认值为20，最大值100。<br>示例值： 30 |
| PreciseSearch | 否 | 否 | Uint64 | 控制Filters相关字段是否为精确匹配。 0: 默认值，topicName 和 logsetName 模糊匹配 1: topicName 精确匹配 2: logsetName精确匹配 3: topicName 和 logsetName 都精确匹配<br>示例值： 1 |
| BizType | 否 | 否 | Uint64 | 主题类型 0:日志主题，默认值 1:指标主题<br>示例值： 0 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Topics | Array of TopicInfo | 日志主题列表<br>示例值：查看 |
| TotalCount | Int64 | 总数目<br>示例值： 1 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |

| 错误码 | 描述 |
| --- | --- |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AnalysisSwitchClose | 字段没有开启分析功能。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 修改日志主题

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于修改日志主题。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 01:45:25。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyTopic |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TopicName | 否 | 否 | String | 日志主题名称<br>示例值： topic_name |
| Tags | 否 | 否 | Array of Tag | 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10个标签键值对，并且不能有重复的键值对。<br>示例值：查看 |
| Status | 否 | 否 | Bool | 主题是否开启采集，true：开启采集；false：关闭采集。 控制台目前不支持修改此参数。<br>示例值： true |
| AutoSplit | 否 | 否 | Bool | 是否开启自动分裂。true表示开启自动分裂，false表示关闭自动分裂。 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | 示例值： true |
| MaxSplitPartitions | 否 | 否 | Int64 | 若开启最大分裂，该主题能够能够允许的最大分区数<br>示例值： 50 |
| Period | 否 | 否 | Int64 | 生命周期，单位天；可取值范围1~90<br>示例值： 30 |
| UserSample | 否 | 否 | String | 用户自定义抽样配置<br>示例值： [{"key":"key1","rule":"!=200","rate":10}] |
| UserSampleStatus | 否 | 否 | Bool | 用户抽样配置开关：默认关闭。true表示开启用户抽样配置，false表示关闭用户抽样配置。<br>示例值： true |
| StorageType | 否 | 否 | String | 存储类型:hot 标准存储<br>示例值： hot |
| Describes | 否 | 否 | String | 日志主题描述<br>示例值： 部门业务日志主题 |
| HotPeriod | 否 | 否 | Uint64 | 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<br>示例值： 1 |
| IsWebTracking | 否 | 否 | Bool | 免鉴权开关。 false：关闭； true：开启。 开启后将支持指定操作匿名访问该日志主题。详情请参见日志主题。<br>示例值： false |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InternalError | 内部错误。 |
| ResourceNotFound | 资源不存在。 |
| LimitExceeded | 超过配额限制。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |
| FailedOperation.TopicClosed | 日志主题已关闭。 |
| InvalidParameter.TopicConflict | 指定日志集下已经有同名的日志主题。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| FailedOperation.InvalidPeriod | 离线存储保存时间不得少于7天。 |

# 日志相关接口
# 上下文检索

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于搜索日志上下文附近的内容

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 01:14:03。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeLogContext |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 要查询的日志主题ID<br>示例值： 682d0718-07bb-4ec0-9fda-f1e9a2767e0b |
| BTime | 是 | 否 | String | 日志时间, 即SearchLog接口返回信息中Results结构体中的Time，需按照 UTC+8 时区将该毫秒级Unix时间戳转换为 YYYY-mm-dd HH:MM:SS.FFF 格式的字符串。<br>示例值： 2021-04-25 14:25:00.000 |
| PkgId | 是 | 否 | String | 日志包序号，即SearchLog接口返回信息中Results结构体中的PkgId。<br>示例值： 528C1318606EFEB8-1A7 |
| PkgLogId | 是 | 否 | Int64 | 日志包内一条日志的序号，即SearchLog接口返回信息中Results结构中的PkgLogId。<br>示例值： 65536 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| PrevLogs | 否 | 否 | Int64 | 前${PrevLogs}条日志，默认值10。<br>示例值： 10 |
| NextLogs | 否 | 否 | Int64 | 后${NextLogs}条日志，默认值10。<br>示例值： 10 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| LogContextInfos | Array of LogContextInfo | 日志上下文信息集合<br>示例值：查看 |
| PrevOver | Bool | 上文日志是否已经返回完成（当PrevOver为false，表示有上文日志还未全部返回）。<br>示例值： true |
| NextOver | Bool | 下文日志是否已经返回完成（当NextOver为false，表示有下文日志还未全部返回）。<br>示例值： true |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |

| 错误码 | 描述 |
| --- | --- |
| FailedOperation.SearchTimeout | 查询超时。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |
| FailedOperation.QueryError | 查询语句运行失败。 |
| FailedOperation.InvalidContext | 检索游标已失效或不存在。 |
| FailedOperation.SyntaxError | 查询语句解析错误。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取直方图

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于构建直方图

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 01:29:35。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeLogHistogram |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 否 | 否 | String | 要查询的日志主题ID<br>示例值： 682d0718-07bb-4ec0-9fda-f1e9a2767e0b |
| From | 是 | 否 | Int64 | 要查询的日志的起始时间，Unix时间戳，单位ms<br>示例值： 1608794855000 |
| To | 是 | 否 | Int64 | 要查询的日志的结束时间，Unix时间戳，单位ms<br>示例值： 1608794855000 |
| Query | 是 | 否 | String | 查询语句<br>示例值： http_status:200 |
| Interval | 否 | 否 | Int64 | 时间间隔: 单位ms 限制性条件：(To-From) / interval <= 200<br>示例值： 10 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| QueryOptimize | 否 | 否 | Uint64 | 0（默认值）：不执行语法优化；1：执行语法优化。<br>示例值： 0 |
| SyntaxRule | 否 | 否 | Uint64 | 检索语法规则，默认值为0。 0：Lucene语法，1：CQL语法。 详细说明参见检索条件语法规则<br>示例值： 1 |
| TopicIds | 否 | 否 | Array of String | 要检索分析的日志主题ID列表。TopicId和TopicIds二个参数只能选择其中一个参数进行查询。<br>示例值： ["682d0718-07bb-4ec0-9fda-f1e9a2767e0b"] |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Interval | Int64 | 统计周期： 单位ms<br>示例值： 10 |
| TotalCount | Int64 | 命中关键字的日志总条数<br>示例值： 12 |
| HistogramInfos | Array of HistogramInfo | 周期内统计结果详情<br>示例值：查看 |
| Query | String | 返回语法优化后的语句（QueryOptimize 为 1 时返回，其他情况返回空字符串）<br>示例值： message.retcode:0 AND message.reqsource:mc |
| Border | Int64 | 日志沉降边界。Unix时间戳（毫秒），在border之前的数据为冷存储，border之后的为标准存储。<br>示例值： 0 |
| Topics | SearchLogTopics | 多日志主题检索，各个日志主题信息<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| FailedOperation.SearchTimeout | 查询超时。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |
| FailedOperation.QueryError | 查询语句运行失败。 |
| FailedOperation.InvalidContext | 检索游标已失效或不存在。 |
| FailedOperation.SyntaxError | 查询语句解析错误。 |
| LimitExceeded.LogSearch | 并发查询超过限制，单topic并发最大值15。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 检索分析日志

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于检索分析日志, 该接口除受默认接口请求频率限制外，针对单个日志主题，查询并发数不能超过15。

默认接口请求频率限制：10000次/秒。

接口更新时间：2025-01-17 19:57:14。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：SearchLog |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 否 | 否 | String | 要检索分析的日志主题ID，仅能指定一个日志主题。 如需同时检索多个日志主题，请使用Topics参数。 TopicId 和 Topics 不能同时使用，在一次请求中有且只能选择一个。<br>示例值： 682d0718-07bb-4ec0-9fda-f1e9a2767e0b |
| From | 是 | 否 | Int64 | 要检索分析的日志的起始时间，Unix时间戳（毫秒）<br>示例值： 1608794854000 |
| To | 是 | 否 | Int64 | 要检索分析的日志的结束时间，Unix时间戳（毫秒）<br>示例值： 1608794855000 |
| Query | 是 | 否 | String | 检索分析语句，最大长度为12KB 语句由 [检索条件] |
| Limit | 否 | 否 | Int64 | 表示单次查询返回的原始日志条数，默认为100，最大值为1000。 注意： 仅当检索分析语句(Query)不包含SQL时有效 SQL结果条数指定方式参考SQL LIMIT语法 可通过两种方式获取后续更多日志： Context:透传上次接口返回的Context值，获取后续更多日志，总计最多可获取1万条原始日志 Offset:偏移量，表示从第几行开始返回原始日志，无日志条数限制<br>示例值： 100 |
| Context | 否 | 否 | String | 透传上次接口返回的Context值，可获取后续更多日志，总计最多可获取1万条原始日志，过期时间1小时。 注意： 透传该参数时，请勿修改除该参数外的其它参数 仅适用于单日志主题检索，检索多个日志主题时，请使用Topics中的Context 仅当检索分析语句(Query)不包含SQL时有效，SQL获取后续结果参考SQL LIMIT语法<br>示例值：<br>Y29udGV4dC04MjMzNWRkMi01YmMxLTQ4NGYtYjQ4MS04MDg0NzAwYjQ1NDUxNjcy |
| Sort | 否 | 否 | String | 原始日志是否按时间排序返回；可选值：asc(升序)、desc(降序)，默认为 desc 注意： 仅当检索分析语句(Query)不包含SQL时有效 SQL结果排序方式参考SQL ORDER BY语法<br>示例值： asc |
| HighLight | 否 | 否 | Bool | 是否返回符合检索条件的关键词，一般用于高亮显示匹配的关键词，仅支持键值检索<br>示例值： false |
| UseNewAnalysis | 否 | 否 | Bool | 为true代表使用新的检索结果返回方式，输出参数AnalysisRecords和Columns有效 为false时代表使用老的检索结果返回方式, 输出AnalysisResults和ColNames有效 两种返回方式在编码格式上有少量区别，建议使用true<br>示例值： false |
| QueryOptimize | 否 | 否 | Uint64 | 0：不执行语法优化；1：执行语法优化<br>示例值： 0 |
| SamplingRate | 否 | 否 | Float | 执行统计分析（Query中包含SQL）时，是否对原始日志先进行采样，再进行统计分析。 0：自动采样; 0～1：按指定采样率采样，例如0.02; 1：不采样，即精确分析 默认值为1<br>示例值： 0.1 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| SyntaxRule | 否 | 否 | Uint64 | 检索语法规则，默认值为0，推荐使用1。 0：Lucene语法 1：CQL语法（日志服务专用检索语法，控制台默认也使用该语法规则）。 详细说明参见检索条件语法规则<br>示例值： 1 |
| Topics | 否 | 否 | Array of MultiTopicSearchInformation | 要检索分析的日志主题列表，最大支持50个日志主题。 检索单个日志主题时请使用TopicId。 TopicId 和 Topics 不能同时使用，在一次请求中有且只能选择一个。<br>示例值：查看 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Context | String | 透传本次接口返回的Context值，可获取后续更多日志，过期时间1小时。 注意： 仅适用于单日志主题检索，检索多个日志主题时，请使用Topics中的Context<br>示例值： Y29udGV4dC04MjMzNWRkMi01YmMxLTQ4NGYtYjQ4MS04MDg0NzAwYjQ1NDUxNjcy |
| ListOver | Bool | 符合检索条件的日志是否已全部返回，如未全部返回可使用Context参数获取后续更多日志 注意：仅当检索分析语句(Query)不包含SQL时有效<br>示例值： false |
| Analysis | Bool | 返回的是否为统计分析（即SQL）结果<br>示例值： false |
| ColNames | Array of String | 日志统计分析结果的列名 当UseNewAnalysis为false时生效<br>示例值： ["col1"] |
| Results | Array of LogInfo | 匹配检索条件的原始日志<br>示例值：查看 |
| AnalysisResults | Array of LogItems | 日志统计分析结果 当UseNewAnalysis为false时生效<br>示例值：查看 |
| AnalysisRecords | Array of String | 日志统计分析结果 当UseNewAnalysis为true时生效<br>示例值： ["record1"] |
| Columns | Array of Column | 日志统计分析结果的列属性 当UseNewAnalysis为true时生效<br>示例值：查看 |
| Query | String | 返回语法优化后的语句（QueryOptimize 为 1 时返回，其他情况返回空字符串）<br>示例值： message.retcode:0 AND message.reqsource:mc |
| SamplingRate | Float | 本次统计分析使用的采样率<br>示例值： 0.1 |
| Border | Int64 | 日志沉降边界。Unix时间戳（毫秒），在border之前的数据为冷存储，border之后的为标准存储。<br>示例值： 0 |
| Topics | SearchLogTopics | 使用多日志主题检索时，各个日志主题的基本信息，例如报错信息。<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |

| 错误码 | 描述 |
| --- | --- |
| FailedOperation.SearchTimeout | 查询超时。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |
| FailedOperation.QueryError | 查询语句运行失败。 |
| FailedOperation.InvalidContext | 检索游标已失效或不存在。 |
| FailedOperation.SyntaxError | 查询语句解析错误。 |
| LimitExceeded.LogSearch | 并发查询超过限制，单topic并发最大值15。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 日志集相关接口
# 创建日志集

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于创建日志集，返回新创建的日志集的 ID。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-17 02:54:59。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateLogset |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| LogsetName | 是 | 否 | String | 日志集名字，不能重名<br>示例值： logset_name |
| Tags | 否 | 否 | Array of Tag | 标签描述列表。最大支持10个标签键值对，并且不能有重复的键值对<br>示例值：查看 |
| Period | 否 | 否 | Int64 | 生命周期，单位天；可取值范围1～366<br>示例值： 30 |
| LogsetId | 否 | 否 | String | 日志集ID，格式为：用户自定义部分-用户appid，用户自定义部分仅支持小写字母、数字和-，且不能以-开头和结尾，长度为3至40字符，尾部需要使用-拼接用户appid<br>示例值： user-define-34534566 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| LogsetId | String | 日志集ID<br>示例值： dc01b74a-49e4-11eb-b378-0242ac130002 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| LimitExceeded | 超过配额限制。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| LimitExceeded.Logset | 日志集数量超出限制。 |
| FailedOperation.LogsetConflict | 相同的日志集已存在。 |
| InvalidParameter.LogsetConflict | 相同的日志集已存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 删除日志集

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于删除日志集。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-02-14 20:14:19。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteLogset |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| LogsetId | 是 | 否 | String | 日志集ID<br>示例值： eb6eafd9-1656-4d60-b3c2-0a5bf91c9b0d |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.LogsetNotExist | 指定的日志集不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| FailedOperation.LogsetNotEmpty | 日志集下存在日志主题。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取日志集列表

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于获取日志集信息列表。

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 01:30:51。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeLogsets |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Filters | 否 | 否 | Array of Filter | logsetName 按照【日志集名称】进行过滤。 类型：String 必选：否 logsetId 按照【日志集ID】进行过滤。 类型：String 必选：否 tagKey 按照【标签键】进行过滤。 类型：String 必选：否 tag:tagKey 按照【标签键值对】进行过滤。 tagKey使用具体的标签键进行替换。 类型：String 必选：否 每次请求的Filters的上限为10，Filter.Values的上限为5。<br>示例值：查看 |
| Offset | 否 | 否 | Int64 | 分页的偏移量，默认值为0<br>示例值： 0 |
| Limit | 否 | 否 | Int64 | 分页单页的限制数目，默认值为20，最大值100<br>示例值： 1 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| TotalCount | Int64 | 分页的总数目<br>示例值： 1 |
| Logsets | Array of LogsetInfo | 日志集列表<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| ResourceNotFound | 资源不存在。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 修改日志集

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于修改日志集信息

默认接口请求频率限制：20次/秒。

接口更新时间：2025-02-14 20:16:17。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyLogset |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| LogsetId | 是 | 否 | String | 日志集ID<br>示例值： dc01b74a-49e4-11eb-b378-0242ac130002 |
| LogsetName | 否 | 否 | String | 日志集名称<br>示例值： changename |
| Tags | 否 | 否 | Array of Tag | 日志集的绑定的标签键值对。最大支持10个标签键值对，同一个资源只能同时绑定一个标签键。<br>示例值：查看 |
| Period | 否 | 否 | Int64 | 生命周期，单位为天；可取值范围1~366<br>示例值： 30 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| LimitExceeded | 超过配额限制。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.LogsetNotExist | 指定的日志集不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| InvalidParameter.LogsetConflict | 相同的日志集已存在。 |
| FailedOperation.PeriodModifyForbidden | 修改的生命周期被禁止。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 机器组相关接口
# 添加机器组信息

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

用于添加机器组信息

默认接口请求频率限制：20次/秒。

接口更新时间：2024-12-31 20:10:59。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：AddMachineGroupInfo |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| MachineGroupType | 是 | 否 | MachineGroupTypeInfo | 机器组类型 目前type支持 ip 和 label<br>示例值：查看 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见公共错误码。

# 创建机器组

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

创建机器组

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-17 02:55:38。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateMachineGroup |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| GroupName | 是 | 否 | String | 机器组名字，不能重复<br>示例值： group_name |
| MachineGroupType | 是 | 否 | MachineGroupTypeInfo | 创建机器组类型。取值如下： Type：ip，Values中为ip字符串列表创建机器组 Type：label，Values中为标签字符串列表创建机器组<br>示例值：查看 |
| Tags | 否 | 否 | Array of Tag | 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机器组。最大支持10个标签键值对，同一个资源只能绑定到同一个标签键下。<br>示例值：查看 |
| AutoUpdate | 否 | 否 | Bool | 是否开启机器组自动更新。true表示开启机器组自动更新，false表示关闭机器组自动更新。是否开启机器组自动更新。默认false<br>示例值： true |
| UpdateStartTime | 否 | 否 | String | 升级开始时间，建议业务低峰期升级LogListener<br>示例值： "12:00:00" |
| UpdateEndTime | 否 | 否 | String | 升级结束时间，建议业务低峰期升级LogListener<br>示例值： "13:00:00" |
| ServiceLogging | 否 | 否 | Bool | 是否开启服务日志，用于记录因Loglistener 服务自身产生的log，开启后，会创建内部日志集cls_service_logging和日志主题 loglistener_status,loglistener_alarm,loglistener_business，不产生计费。默认false<br>示例值： true |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Flag | 否 | 否 | String | TKE标志位，默认值为空字符串。空字符串表示日志不是来自于TKE，label_k8s表示日志来自于TKE。<br>示例值： label_k8s |
| DelayCleanupTime | 否 | 否 | Int64 | 机器组中机器离线清理时间。单位：天<br>示例值： 30 |
| MetaTags | 否 | 否 | Array of MetaTagInfo | 机器组元数据信息列表<br>示例值：查看 |
| OSType | 否 | 否 | Uint64 | 系统类型，取值如下： 0：Linux （默认值） 1：Windows<br>示例值： 0 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| GroupId | String | 机器组ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| LimitExceeded | 超过配额限制。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| LimitExceeded.MachineGroup | 机器组超过限制。 |
| InvalidParameter.MachineGroupConflict | 同名机器组已经存在。 |
| LimitExceeded.MachineGroupIp | 机器组IP超过限制。 |
| LimitExceeded.MachineGroupLabels | 机器组Label超过限制。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |

| 错误码 | 描述 |
| --- | --- |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 删除机器组

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

删除机器组

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-10 02:10:28。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteMachineGroup |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| ResourceNotFound | 资源不存在。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.MachineGroupNotExist | 机器组不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 删除机器组信息

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

用于删除机器组信息

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-10 02:11:27。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteMachineGroupInfo |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| MachineGroupType | 是 | 否 | MachineGroupTypeInfo | 机器组类型 目前type支持 ip 和 label<br>示例值：查看 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

该接口暂无业务逻辑相关的错误码，其他错误码详见公共错误码。

# 获取机器组列表

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

获取机器组信息列表

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 01:39:48。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeMachineGroups |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Filters | 否 | 否 | Array of Filter | machineGroupName 按照【机器组名称】进行过滤。 类型：String 必选：否 machineGroupId 按照【机器组ID】进行过滤。 类型：String 必选：否 osType 按照【操作系统类型】进行过滤。 类型：Int 必选：否 tagKey 按照【标签键】进行过滤。 类型：String 必选：否 tag:tagKey 按照【标签键值对】进行过滤。tagKey使用具体的标签键进行替换。 类型：String 必选：否 每次请求的Filters的上限为10，Filter.Values的上限为5。<br>示例值：查看 |
| Offset | 否 | 否 | Int64 | 分页的偏移量，默认值为0<br>示例值： 0 |
| Limit | 否 | 否 | Int64 | 分页单页的限制数目，默认值为20，最大值100<br>示例值： 10 |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| MachineGroups | Array of MachineGroupInfo | 机器组信息列表<br>示例值：查看 |
| TotalCount | Int64 | 分页的总数目<br>示例值： 10 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取机器状态

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

获取制定机器组下的机器状态

默认接口请求频率限制：20次/秒。

接口更新时间：2025-02-26 05:22:00。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeMachines |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| GroupId | 是 | 否 | String | 查询的机器组ID<br>示例值： ae0d14a7-94e1-461e-8f5d-7d66a0d56d35 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Machines | Array of MachineInfo | 机器状态信息组<br>示例值：查看 |
| AutoUpdate | Int64 | 机器组是否开启自动升级功能。 0：未开启自动升级；1：开启了自动升级。<br>示例值： 0 |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| UpdateStartTime | String | 机器组自动升级功能预设开始时间<br>示例值： 00:00:00 |
| UpdateEndTime | String | 机器组自动升级功能预设结束时间<br>示例值： 01:00:00 |
| LatestAgentVersion | String | 当前用户可用最新的Loglistener版本<br>示例值： 2.2.0 |
| ServiceLogging | Bool | 是否开启服务日志。true表示开启服务日志，false表示不开启服务日志。<br>示例值： true |
| Flag | String | TKE标志位，默认值为空字符串。空字符串表示日志不是来自于TKE，label_k8s表示日志来自于TKE。<br>示例值： label_k8s |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.MachineGroupNotExist | 机器组不存在。 |
| ResourceNotFound.AgentVersionNotExist | agent version不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |

| 错误码 | 描述 |
| --- | --- |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 修改机器组

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

修改机器组

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-17 19:41:26。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyMachineGroup |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| GroupName | 否 | 否 | String | 机器组名称<br>示例值： group_name |
| MachineGroupType | 否 | 否 | MachineGroupTypeInfo | 机器组类型。Type：ip，Values中为ip字符串列表机器组；Type：label，Values中为标签字符串列表机器组。<br>示例值：查看 |
| Tags | 否 | 否 | Array of Tag | 标签列表<br>示例值：查看 |
| AutoUpdate | 否 | 否 | Bool | 是否开启机器组自动更新。true表示开启自动升级，false表示关闭自动升级。<br>示例值： true |
| UpdateStartTime | 否 | 否 | String | 升级开始时间，建议业务低峰期升级LogListener<br>示例值： "12:00:00" |
| UpdateEndTime | 否 | 否 | String | 升级结束时间，建议业务低峰期升级LogListener<br>示例值： "13:00:00" |
| ServiceLogging | 否 | 否 | Bool | 是否开启服务日志，用于记录因Loglistener 服务自身产生的log，开启后，会创建内部日志集cls_service_logging和日志主题loglistener_status,loglistener_alarm,loglistener_business，不产生计费<br>示例值： true |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Flag | 否 | 否 | String | TKE标志位，默认值为空字符串。空字符串表示日志不是来自于TKE，label_k8s表示日志来自于TKE。<br>示例值： label_k8s |
| DelayCleanupTime | 否 | 否 | Int64 | 机器组中机器定期离线清理时间<br>示例值： 30 |
| MetaTags | 否 | 否 | Array of MetaTagInfo | 机器组元数据信息列表<br>示例值：查看 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.MachineGroupNotExist | 机器组不存在。 |
| InvalidParameter.MachineGroupConflict | 同名机器组已经存在。 |
| LimitExceeded.MachineGroupIp | 机器组IP超过限制。 |
| LimitExceeded.MachineGroupLabels | 机器组Label超过限制。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 索引相关接口
# 创建索引

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于创建索引

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-01 04:48:55。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateIndex |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 3edc8551-111b-473b-9fc4-ed66e20dfa8a |
| Status | 否 | 否 | Bool | 是否生效，默认为true。true表示索引生效，false表示索引不生效。<br>示例值： true |
| Rule | 是 | 否 | RuleInfo | 索引规则<br>示例值：查看 |
| IncludeInternalFields | 否 | 否 | Bool | 内置保留字段（FILENAME，HOSTNAME__及__SOURCE）是否包含至全文索引，默认为false，推荐设置为true  false:不包含 true:包含<br>示例值： true |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| MetadataFlag | 否 | 否 | Uint64 | 元数据字段（前缀为__TAG__的字段）是否包含至全文索引，默认为0，推荐设置为1 0:仅包含开启键值索引的元数据字段 1:包含所有元数据字段 2:不包含任何元数据字段<br>示例值： 0 |
| CoverageField | 否 | 否 | String | 自定义日志解析异常存储字段。<br>示例值： RAWLOG |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| LimitExceeded | 超过配额限制。 |
| FailedOperation | 操作失败。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |
| InvalidParameter.IndexConflict | 指定日志主题已经存在索引规则。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter.InValidIndexRuleForSearchLow | 低频不支持配置kv和tag索引。 |

# 删除索引配置

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于日志主题的索引配置

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-10 02:03:08。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteIndex |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 826f8b26-b054-4a0d-8c8e-f3d609f5e0ea |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| ResourceNotFound.IndexNotExist | 索引不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取索引配置信息

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于获取索引配置信息

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 00:21:58。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeIndex |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 826f8b26-b054-4a0d-8c8e-f3d609f5e0ea |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| TopicId | String | 日志主题ID<br>示例值： 826f8b26-b054-4a0d-8c8e-f3d609f5e0ea |
| Status | Bool | 是否生效。true表示索引生效，false表示索引失效。<br>示例值： true |
| Rule | RuleInfo | 索引配置信息<br>示例值：查看 |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| ModifyTime | String | 索引修改时间，初始值为索引创建时间。<br>示例值： 2021-01-13 21:00:08 |
| IncludeInternalFields | Bool | 内置保留字段（FILENAME，HOSTNAME__及__SOURCE）是否包含至全文索引 false:不包含 true:包含<br>示例值： true |
| MetadataFlag | Uint64 | 元数据字段（前缀为__TAG__的字段）是否包含至全文索引 0:仅包含开启键值索引的元数据字段 1:包含所有元数据字段 2:不包含任何元数据字段<br>示例值： 1 |
| CoverageField | String | 自定义日志解析异常存储字段。<br>示例值： ROWLOG |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |
| ResourceNotFound.IndexNotExist | 索引不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |

| 错误码 | 描述 |
| --- | --- |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 修改索引

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于修改索引配置

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-15 05:30:34。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyIndex |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 826f8b26-b054-4a0d-8c8e-f3d609f5e0ea |
| Status | 否 | 否 | Bool | 索引是否生效，默认不生效。true表示索引生效，false表示索引失效。<br>示例值： true |
| Rule | 否 | 否 | RuleInfo | 索引规则<br>示例值：查看 |
| IncludeInternalFields | 否 | 否 | Bool | 内置保留字段（FILENAME，HOSTNAME__及__SOURCE）是否包含至全文索引，默认为false，推荐设置为true  false:不包含 true:包含<br>示例值： false |
| MetadataFlag | 否 | 否 | Uint64 | 元数据字段（前缀为__TAG__的字段）是否包含至全文索引，默认为0，推荐设置为1  0:仅包含 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | 开启键值索引的元数据字段 1:包含所有元数据字段 2:不包含任何元数据字段<br>示例值： 0 |
| CoverageField | 否 | 否 | String | 自定义日志解析异常存储字段。<br>示例值： ROWLOG |

# 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |
| ResourceNotFound.IndexNotExist | 索引不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |
| InvalidParameter.InValidIndexRuleForSearchLow | 低频不支持配置kv和tag索引。 |

# 采集配置相关接口
# 应用采集配置到指定机器组

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

应用采集配置到指定机器组

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:12:27。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ApplyConfigToMachineGroup |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| ConfigId | 是 | 否 | String | 采集配置ID<br>示例值： 48c3ae1f-1e6e-47e7-b08f-5c2507bb6695 |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： 48c3ae1f-1e6e-47e7-b08f-5c2507bb6695 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.MachineGroupNotExist | 机器组不存在。 |
| ResourceNotFound.ConfigNotExist | 指定的采集规则配置不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 创建采集规则配置

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

创建采集规则配置

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:17:35。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateConfig |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Name | 是 | 否 | String | 采集配置名称<br>示例值： config_name |
| LogFormat | 否 | 否 | String | 日志格式化方式<br>示例值： default |
| Path | 否 | 否 | String | 日志采集路径，包含文件名，支持多个路径，多个路径之间英文逗号分隔，文件采集情况下必填<br>示例值： /data/access.log |
| LogType | 否 | 否 | String | 采集的日志类型，默认为minimalist_log。支持以下类型： json_log代表：JSON-文件日志（详见使用 JSON 提取模式采集日志）； delimiter_log代表：分隔符-文件日志（详见使用分隔符提取模式采集日志）； minimalist_log代表：单行全文-文件日志（详见使用单行全文提取模式采集日志）； fullregex_log代表：单行完全正则-文件日志（详见使用单行-完全正则提取模式采集日志）； multiline_log代表：多行全文-文件日志（详见使用多行全文提取模式采集日志）； multiline_fullregex_log代表：多行完全正则-文件日志（详见使用多行-完全正则提取模式采集日志）； user_define_log代表：组合解析（适用于多格式嵌套的日志，详见使用组合解析提取模式采集日志）； service_syslog代表：syslog 采集（详见采集 Syslog）； windows_event_log代表：Windows事件日志（详见采集 Windows 事件日志）。<br>示例值： delimiter_log |
| ExtractRule | 否 | 否 | ExtractRuleInfo | 提取规则，如果设置了ExtractRule，则必须设置LogType<br>示例值：查看 |
| ExcludePaths | 否 | 否 | Array of ExcludePathInfo | 采集黑名单路径列表<br>示例值：查看 |
| Output | 是 | 否 | String | 采集配置所属日志主题ID即TopicId<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| UserDefineRule | 否 | 否 | String | 用户自定义采集规则，Json格式序列化的字符串。当LogType为user_define_log时，必填。<br>示例值： rule |
| ConfigExtraId | 否 | 否 | String | config_extra表主键ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| ConfigFlag | 否 | 否 | String | 采集配置标签<br>示例值： label_k8s |
| AdvancedConfig | 否 | 否 | String | 高级采集配置。 Json字符串， Key/Value定义为如下： ClsAgentFileTimeout(超时属性), 取值范围: 大于等于0的整数， 0为不超时 ClsAgentMaxDepth(最大目录深度)，取值范围: 大于等于0的整数 ClsAgentParseFailMerge(合并解析失败日志)，取值范围: true或false 样 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | 例： {"ClsAgentFileTimeout":0,"ClsAgentMaxDepth":10,"ClsAgentParseFailMerge":true} 控制台默认占位值：{"ClsAgentDefault":0}<br>示例值： config |
| Source | 否 | 否 | Uint64 | 采集配置来源，0：默认来源，1: TKE<br>示例值： 0 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| ConfigId | String | 采集配置ID<br>示例值： 48c3ae1f-1e6e-47e7-b08f-5c2507bb6695 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| LimitExceeded.Config | 采集规则配置超过最大值限制。 |
| InvalidParameter.ConfigConflict | 相同的采集配置规则已经存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 创建特殊采集配置任务

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于创建特殊采集配置任务

默认接口请求频率限制：20次/秒。

接口更新时间：2025-08-14 10:43:17。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateConfigExtra |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Name | 是 | 否 | String | 采集配置规程名称，最长63个字符，只能包含小写字符、数字及分隔符（“-”），且必须以小写字符开头，数字或小写字符结尾<br>示例值： configname |
| TopicId | 是 | 否 | String | 日志主题id<br>示例值：3269961d-52e0-407b-9dcd-34bd830c6f19 |
| Type | 是 | 否 | String | 日志源类型。支持 container_stdout：容器标准输出；container_file：容器文件路径；host_file：节点文件路径。<br>示例值： container_stdout |
| HostFile | 否 | 否 | HostFileInfo | 节点文件路径类型配置。<br>示例值：查看 |
| ContainerFile | 否 | 否 | ContainerFileInfo | 容器文件路径类型配置。<br>示例值：查看 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ContainerStdout | 否 | 否 | ContainerStdoutInfo | 容器标准输出类型配置。<br>示例值：查看 |
| LogType | 是 | 否 | String | 采集的日志类型，默认为minimalist_log。支持以下类型： json_log代表：JSON-文件日志（详见使用 JSON 提取模式采集日志）； delimiter_log代表：分隔符-文件日志（详见使用分隔符提取模式采集日志）； minimalist_log代表：单行全文-文件日志（详见使用单行全文提取模式采集日志）； fullregex_log代表：单行完全正则-文件日志（详见使用单行-完全正则提取模式采集日志）； multiline_log代表：多行全文-文件日志（详见使用多行全文提取模式采集日志）； multiline_fullregex_log代表：多行完全正则-文件日志（详见使用多行-完全正则提取模式采集日志）； user_define_log代表：组合解析（适用于多格式嵌套的日志，详见使用组合解析提取模式采集日志）。<br>示例值： minimalist_log |
| LogFormat | 否 | 否 | String | 日志格式化方式，用于容器采集场景。 - 已废弃 stdout-docker-json：用于docker容器采集场景 stdout-containerd：用于containerd容器采集场景<br>示例值： stdout-containerd |
| ExtractRule | 否 | 否 | ExtractRuleInfo | 提取规则，如果设置了ExtractRule，则必须设置LogType<br>示例值：查看 |
| ExcludePaths | 否 | 否 | Array of ExcludePathInfo | 采集黑名单路径列表<br>示例值：查看 |
| UserDefineRule | 否 | 否 | String | 组合解析采集规则，用于复杂场景下的日志采集。 取值参考：使用组合解析提取模式采集日志<br>示例值： rule |
| GroupId | 否 | 否 | String | 绑定的机器组id<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| ConfigFlag | 是 | 否 | String | 采集配置标记。 目前只支持label_k8s，用于标记自建k8s集群使用的采集配置<br>示例值： label_k8s |
| LogsetId | 是 | 否 | String | 日志集id<br>示例值： 57f5808c-4a55-11eb- |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | b378-0242ac130002 |
| LogsetName | 是 | 否 | String | 日志集name<br>示例值： name1 |
| TopicName | 是 | 否 | String | 日志主题名称<br>示例值： name2 |
| GroupIds | 否 | 否 | Array of String | 绑定的机器组id列表<br>示例值： ["59b6014b-6507-44c2-9d4f-a5bee6df9ab5","0af7e6bb-fc91-4ee8-ad24-1129e9c91c6c"] |
| CollectInfos | 否 | 否 | Array of CollectInfo | 采集相关配置信息。详情见CollectInfo复杂类型配置。<br>示例值：查看 |
| AdvancedConfig | 否 | 否 | String | 高级采集配置。 Json字符串， Key/Value定义为如下： ClsAgentFileTimeout(超时属性), 取值范围: 大于等于0的整数， 0为不超时 ClsAgentMaxDepth(最大目录深度)，取值范围: 大于等于0的整数 ClsAgentParseFailMerge(合并解析失败日志)，取值范围: true或false ClsAgentDefault(自定义默认值，无特殊含义，用于清空其他选项)，建议取值0<br>示例值： config |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| ConfigExtraId | String | 采集配置扩展信息ID<br>示例值： 6ef60bec-0242-43af-bb20-270359fb54a8 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |

| 错误码 | 描述 |
| --- | --- |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 删除采集规则配置

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

删除采集规则配置

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-10 01:12:32。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteConfig |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| ConfigId | 是 | 否 | String | 采集规则配置ID<br>示例值： 48c3ae1f-1e6e-47e7-b08f-5c2507bb6695 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| ResourceNotFound | 资源不存在。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.ConfigNotExist | 指定的采集规则配置不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 删除应用到机器组的采集配置

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

删除应用到机器组的采集配置

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:34:43。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteConfigFromMachineGroup |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： 48c3ae1f-1e6e-47e7-b08f-5c2507bb6695 |
| ConfigId | 是 | 否 | String | 采集配置ID<br>示例值： 48c3ae1f-1e6e-47e7-b08f-5c2507bb6695 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.MachineGroupNotExist | 机器组不存在。 |
| ResourceNotFound.ConfigNotExist | 指定的采集规则配置不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取特殊采集配置

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

本接口用于获取特殊采集配置，特殊采集配置应用于自建K8S环境的采集Agent

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:44:47。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeConfigExtras |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Filters | 否 | 否 | Array of Filter | 过滤器，支持如下选项： name 按照【特殊采集配置名称】进行模糊匹配过滤。 类型：String configExtraId 按照【特殊采集配置ID】进行过滤。 类型：String topicId 按照【日志主题】进行过滤。 类型：String machineGroupId 按照【机器组ID】进行过滤。 类型：String 每次请求的Filters的上限为10，Filter.Values的上限为5。<br>示例值：查看 |
| Offset | 否 | 否 | Uint64 | 分页的偏移量，默认值为0<br>示例值： 0 |
| Limit | 否 | 否 | Uint64 | 分页单页的限制数目，默认值为20，最大值100<br>示例值： 10 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Configs | Array of ConfigExtraInfo | 采集配置列表<br>示例值：查看 |
| TotalCount | Uint64 | 过滤到的总数目<br>示例值： 1 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取采集规则配置所绑定的机器组

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

获取采集规则配置所绑定的机器组

默认接口请求频率限制：20次/秒。

接口更新时间：2025-01-14 20:19:57。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeConfigMachineGroups |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| ConfigId | 是 | 否 | String | 采集配置ID<br>示例值： 48c3ae1f-1e6e-47e7-b08f-5c2507bb6695 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| MachineGroups | Array of MachineGroupInfo | 采集规则配置绑定的机器组列表<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| ResourceNotFound | 资源不存在。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.ConfigNotExist | 指定的采集规则配置不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取采集规则配置

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

获取采集规则配置

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 00:46:59。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeConfigs |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| Filters | 否 | 否 | Array of Filter | configName 按照【采集配置名称】进行模糊匹配过滤。 类型：String 必选：否 configId 按照【采集配置ID】进行过滤。 类型：String 必选：否 topicId 按照【日志主题】进行过滤。 类型：String 必选：否 每次请求的Filters的上限为10，Filter.Values的上限为5。<br>示例值：查看 |
| Offset | 否 | 否 | Int64 | 分页的偏移量，默认值为0<br>示例值： 0 |
| Limit | 否 | 否 | Int64 | 分页单页的限制数目，默认值为20，最大值100<br>示例值： 10 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Configs | Array of ConfigInfo | 采集配置列表<br>示例值：查看 |
| TotalCount | Int64 | 过滤到的总数目<br>示例值： 1 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 获取机器组绑定的采集规则配置

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

获取机器组绑定的采集规则配置

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 01:15:04。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeMachineGroupConfigs |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： 48c3ae1f-1e6e-47e7-b08f-5c2507bb6695 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Configs | Array of ConfigInfo | 采集规则配置列表<br>示例值：查看 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.MachineGroupNotExist | 机器组不存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 修改采集规则配置

## 1. 接口描述

接口请求域名： cls.api3.finance.cloud.tencent.com。

修改采集规则配置

默认接口请求频率限制：20次/秒。

接口更新时间：2025-07-30 01:30:32。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ModifyConfig |
| Version | 是 | 否 | String | 公共参数，本接口取值：2020-10-16 |
| Region | 是 | 否 | String | 公共参数，地域信息可通过DescribeRegions接口查看产品支持的地域列表 |
| ConfigId | 是 | 否 | String | 采集规则配置ID，通过获取采集规则配置返回信息获取。<br>示例值： a2c6342c-270a-4468-86cf-92625e466401 |
| Name | 否 | 否 | String | 采集规则配置名称<br>示例值： config_name |
| LogFormat | 否 | 否 | String | 日志格式化方式<br>示例值： default |
| Path | 否 | 否 | String | 日志采集路径，包含文件名<br>示例值： /data/nginx/log/access.log |
| LogType | 否 | 否 | String | 采集的日志类型。支持以下类型： json_log代表：JSON-文件日志（详见使用 JSON 提取模式采集日志）； delimiter_log代表：分隔符-文件日志（详见使用分隔符提取模式采集日志）； minimalist_log代表：单行全文-文件日志（详见使用单行全文提取模式采集日志）； fullregex_log代表：单行完全正则-文件日志（详见使用单行-完全正则提取模式采集日志）； multiline_log代表：多行全文-文件日志（详见使用多行全文提取模式采集日志）； multiline_fullregex_log代表：多行完全正则-文件日志（详见使用多行-完全正则提取模式采集日志）； user_define_log代表：组合解析（适用于多格式嵌套的日志，详见使用组合解析提取模式采集日志）； service_syslog代表：syslog 采集（详见采集 Syslog）； windows_event_log代表：Windows事件日志（详见采集 Windows 事件日志）。<br>示例值： delimiter_log |
| ExtractRule | 否 | 否 | ExtractRuleInfo | 提取规则，如果设置了ExtractRule，则必须设置LogType<br>示例值：查看 |
| ExcludePaths | 否 | 否 | Array of ExcludePathInfo | 采集黑名单路径列表<br>示例值：查看 |
| Output | 否 | 否 | String | 采集配置关联的日志主题（TopicId）<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| UserDefineRule | 否 | 否 | String | 用户自定义解析字符串，Json格式序列化的字符串<br>示例值： rule |
| ConfigExtraId | 否 | 否 | String | config_extra表主键ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ConfigFlag | 否 | 否 | String | 采集配置标<br>示例值： label_k8s |
| AdvancedConfig | 否 | 否 | String | 高级采集配置。 Json字符串， Key/Value定义为如下： ClsAgentFileTimeout(超时属性), 取值范围: 大于等于0的整数， 0为不超时 ClsAgentMaxDepth(最大目录深度)，取值范围: 大于等于0的整数 ClsAgentParseFailMerge(合并解析失败日志)，取值范围: true或false 样例： {"ClsAgentFileTimeout":0,"ClsAgentMaxDepth":10,"ClsAgentParseFailMerge":true}<br>示例值： config |
| Source | 否 | 否 | Uint64 | 采集配置来源，0：默认来源，1: TKE<br>示例值： 0 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| MissingParameter | 缺少参数错误。 |
| InternalError | 内部错误。 |
| ResourceNotFound | 资源不存在。 |
| FailedOperation | 操作失败。 |
| AuthFailure | CAM签名/鉴权错误。 |
| OperationDenied | 操作被拒绝。 |
| ResourceNotFound.ConfigNotExist | 指定的采集规则配置不存在。 |
| InvalidParameter.ConfigConflict | 相同的采集配置规则已经存在。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| OperationDenied.AccountNotExists | 账户不存在。 |

# 数据结构

## AlarmInfo

告警策略描述

被如下接口引用：DescribeAlarms

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Name | 是 | 否 | String | 告警策略名称。<br>示例值： p0-业务告警 |
| AlarmTargets | 是 | 否 | Array of AlarmTargetInfo | 监控对象列表。<br>示例值：查看 |
| MonitorTime | 是 | 否 | MonitorTime | 监控任务运行时间点。<br>示例值：查看 |
| Condition | 是 | 否 | String | 触发条件。<br>示例值： $1.cnt>0 |
| TriggerCount | 是 | 否 | Int64 | 持续周期。持续满足触发条件TriggerCount个周期后，再进行告警；最小值为1，最大值为10。<br>示例值： 1 |
| AlarmPeriod | 是 | 否 | Int64 | 告警重复的周期。单位是min。取值范围是0~1440。<br>示例值： 20 |
| AlarmNoticeIds | 是 | 否 | Array of String | 关联的告警通知模板列表。<br>示例值： ["notice-a808dcfa-6e31-4ef7-bbc3-cbf5396a3ede"] |
| Status | 是 | 否 | Bool | 开启状态。true表示开启，false表示关闭。<br>示例值： true |
| AlarmId | 是 | 否 | String | 告警策略ID。<br>示例值： alarm-b8a247be-d866-4e00-a875-897435f37d18 |
| CreateTime | 是 | 否 | String | 创建时间。<br>示例值： 2024-11-06 19:53:37 |
| UpdateTime | 是 | 否 | String | 最近更新时间。<br>示例值： 2024-11-06 19-53-37 |
| Enable | 是 | 否 | Bool | 开启状态。true表示开启，false表示关闭。<br>示例值： true |
| MessageTemplate | 是 | 是 | String | 自定义通知模板<br>示例值： message tpl TopicName raw result: RawResults |
| CallBack | 是 | 是 | CallBackInfo | 自定义回调模板<br>示例值：查看 |
| Analysis | 是 | 是 | Array of AnalysisDimensional | 多维分析设置<br>示例值：查看 |
| GroupTriggerStatus | 是 | 是 | Bool | 分组触发状态。1：开启，0：关闭（默认）<br>示例值： 1 |
| GroupTriggerCondition | 是 | 是 | Array of String | 分组触发条件。<br>示例值： ["$1.conut"] |
| Tags | 是 | 是 | Array of Tag | 告警策略绑定的标签信息。<br>示例值：查看 |
| MonitorObjectType | 是 | 是 | Uint64 | 监控对象类型。0:执行语句共用监控对象;1:每个执行语句单独选择监控对象。<br>示例值： 0 |
| AlarmLevel | 是 | 是 | Uint64 | 告警级别。0:警告(Warn);1:提醒(Info);2:紧急 (Critical)。<br>示例值： 0 |
| Classifications | 是 | 是 | Array of AlarmClassification | 告警附加分类信息列表。<br>示例值：查看 |
| MultiConditions | 否 | 否 | Array of MultiCondition | 多触发条件。与 Condition互斥。<br>示例值：查看 |
| ConditionInteractiveConfig | 否 | 否 | String | 触发条件交互模式配置信息。<br>示例值： {"Value": {"type":"CONDITION","level":0,"executeQueryIndex":0,"field":"QUERYCOUNT","comparator":">","value":0},"Children":[]} |

## TaskLogStatistic

数据加工任务的统计数据信息

被如下接口引用：DescribeDataTransformProcessInfo

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| StartTime | 是 | 否 | String | 开始时间<br>示例值： 2024-11-19 15:26:20 |
| EndTime | 是 | 否 | String | 结束时间<br>示例值： 2024-11-19 15:26:20 |
| ReadLines | 是 | 是 | Int64 | 读取的源日志主题的行数<br>示例值： 1 |
| WriteLines | 是 | 是 | Int64 | 加工后的行数<br>示例值： 1 |
| FailedLines | 是 | 是 | Int64 | 加工失败的行数<br>示例值： 1 |
| DstTopicLogStatistics | 是 | 是 | Array of TopicIdLogStatistic | 输出到目标日志主题的总体统计数据<br>示例值：查看 |
| FilterLines | 是 | 是 | Uint64 | 加工过滤的行数<br>示例值： 1 |

## FolderInfo

仪表盘文件夹信息

被如下接口引用：DescribeFolders

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Id | 是 | 否 | String | 文件夹Id。<br>示例值： dc01b74a-49e4-11eb-b378-0242ac13000 |
| Name | 是 | 否 | String | 文件夹名称。<br>示例值： 系统监控文件夹 |

## KeyLogInfo

json格式的k-v日志信息

被如下接口引用：DescribeLatestJsonLog、DescribeLatestUserLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 日志key<br>示例值： logKey |
| Value | 是 | 否 | String | 日志内容<br>示例值： logValue |

## KafkaConsumerContent

kafka协议消费内容

被如下接口引用：DescribeKafkaConsumer、DescribeKafkaConsumerPreview、ModifyKafkaConsumer、OpenKafkaConsumer

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Format | 是 | 否 | Int64 | 消费格式 0:全文；1:json<br>示例值： 1 |
| EnableTag | 是 | 否 | Bool | 是否投递 TAG 信息 Format为0时，此字段不需要赋值<br>示例值： false |
| TagTransaction | 否 | 是 | Int64 | tag数据处理方式：1:不平铺（默认值）；2:平铺。 不平铺示例： TAG信息：{"TAG":{"fieldA":200,"fieldB":"text"}} 不平铺：{"TAG":{"fieldA":200,"fieldB":"text"}} 平铺示例： TAG信息：{"TAG":{"fieldA":200,"fieldB":"text"}} 平铺：{"TAG.fieldA":200,"TAG.fieldB":"text"}<br>示例值： 1 |
| MetaFields | 是 | 否 | Array of String | 元数据信息列表, 可选值为：SOURCE、FILENAME 、TIMESTAMP、HOSTNAME、PKGID Format为0时，此字段不需要赋值<br>示例值： ["__HOSTNAME__"] |
| JsonType | 否 | 否 | Int64 | 消费数据Json格式： 1：不转义（默认格式） 2：转义 投递Json格式。 JsonType为1：和原始日志一致，不转义。示例： 日志原文：{"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} 投递到Ckafka：{"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} JsonType为2：转义。示例： 日志原文：{"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} 投递到Ckafka：{"a":"aa","b":"{"b1":"b1b1", "c1":"c1c1"}"}<br>示例值： 1 |

## RuleTagInfo

标签索引配置信息

被如下接口引用：CreateIndex、DescribeIndex、DescribeIndexs、ModifyIndex

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| CaseSensitive | 是 | 否 | Bool | 是否大小写敏感。true表示开启大小写敏感，false表示关闭大小写敏感。<br>示例值： true |
| KeyValues | 是 | 否 | Array of KeyValueInfo | 元字段索引配置中的字段信息<br>示例值：查看 |

## ExcludePathInfo

黑名单path信息

被如下接口引用：CreateConfig、CreateConfigExtra、DescribeAgentConfigs、DescribeConfigExtras、DescribeConfigs、DescribeMachineGroupConfigs、ModifyConfig、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Type | 是 | 否 | String | 类型，选填File或Path<br>示例值： File |
| Value | 是 | 否 | String | Type对应的具体内容<br>示例值： /data/file.log |

## LogsetInfo

日志集相关信息

被如下接口引用：DescribeLogsets、DescribeTopics

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| LogsetId | 是 | 否 | String | 日志集ID<br>示例值： 4463e7b0-3ec8-41a1-ae48-5d24b22167c2 |
| LogsetName | 是 | 否 | String | 日志集名称<br>示例值： 业务A日志集 |
| CreateTime | 是 | 否 | String | 创建时间<br>示例值： 2021-04-01 08:00:00 |
| AssumerUin | 是 | 是 | Uint64 | 若AssumerUin非空，则表示创建该日志集的服务方Uin<br>示例值： 1000945 |
| AssumerName | 是 | 是 | String | 若AssumerUin非空，则表示创建该日志集的服务方名称<br>示例值： TCB |
| LogsetModifyAcl | 是 | 是 | Int64 | 若AssumerUin非空，则表示非改服务方的调用者对于日志集的修改权限<br>示例值： 1 |
| Tags | 是 | 是 | Array of Tag | 日志集绑定的标签<br>示例值：查看 |
| TopicCount | 是 | 否 | Int64 | 日志集下日志主题的数目<br>示例值： 10 |
| RoleName | 是 | 否 | String | 若AssumerUin非空，则表示创建该日志集的服务方角色<br>示例值： TCB_QcsRole |
| Period | 是 | 是 | Int64 | 生命周期，单位为天<br>示例值： 30 |
| LogsetDelACL | 是 | 是 | Int64 | 若AssumerUin非空，则表示非改服务方的调用者对于日志集的删除权限<br>示例值： 1 |
| LogsetShowAcl | 是 | 是 | Uint64 | 若AssumerUin非空，则表示非改服务方的调用者对于日志集的查询权限<br>示例值： 1 |
| MetricTopicCount | 是 | 是 | Int64 | 日志集下指标主题的数目<br>示例值： 10 |

## AgentUpdateStatus

Agent自动升级状态信息

被如下接口引用：UpdateAgentStatus

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| UpdateType | 否 | 否 | Int64 | 升级类型:0-null,1-manual,2-auto,3-force<br>示例值： 0 |
| UpdateAction | 否 | 否 | Int64 | 升级动作:0-null,1-update,2-revert<br>示例值： 0 |
| RetryCount | 否 | 否 | Int64 | 重试次数,最大三次<br>示例值： 2 |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| UpdateStatus | 否 | 否 | Int64 | Agent升级状态:0-querying,1-updating,2-reverting,-1-updatefail,-2-revertfail, -10-notsupport<br>示例值： 0 |
| ErrCode | 是 | 否 | Int64 | 错误码<br>示例值： 0 |
| ErrMsg | 是 | 否 | String | 错误信息<br>示例值： "OK" |

## DashboardInfo

仪表盘信息

被如下接口引用：DescribeDashboards

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| DashboardId | 是 | 否 | String | 仪表盘id<br>示例值： dashboard-57f5808c-4a55-11eb-b378-0242ac130002 |
| DashboardName | 是 | 否 | String | 仪表盘名字<br>示例值： dashboard_name |
| Data | 是 | 是 | String | 仪表盘数据<br>示例值： {"panels":[],"time":["now-h","now"]} |
| CreateTime | 是 | 否 | String | 创建仪表盘的时间<br>示例值： 2019-01-14 19:19:41 |
| AssumerUin | 是 | 是 | Uint64 | AssumerUin非空则表示创建该日志主题的服务方Uin<br>示例值： 1736994580 |
| RoleName | 是 | 是 | String | RoleName非空则表示创建该日志主题的服务方使用的角色<br>示例值： CLB_QcsRole |
| AssumerName | 是 | 是 | String | AssumerName非空则表示创建该日志主题的服务方名称<br>示例值： CLS_business |
| Tags | 是 | 是 | Array of Tag | 日志主题绑定的标签信息<br>示例值：查看 |
| DashboardRegion | 是 | 是 | String | 仪表盘所在地域： 为了兼容老的地域。<br>示例值： region |
| UpdateTime | 是 | 是 | String | 修改仪表盘的时间<br>示例值： 2021-12-22 16:11:27 |
| DashboardTopicInfos | 是 | 是 | Array of DashboardTopicInfo | 仪表盘对应的topic相关信息<br>示例值：查看 |
| FolderName | 是 | 是 | String | 文件夹名称<br>示例值： 客户端仪表盘 |
| FolderId | 是 | 是 | String | 文件夹id<br>示例值： 10001 |

## FieldValueRatioInfos

描述字段值的占比情况

被如下接口引用：DescribeLogFastAnalysis

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Value | 是 | 否 | String | 字段值<br>示例值： 2 |
| Count | 是 | 否 | Int64 | 字段值所占的数量<br>示例值： 23 |
| Ratio | 是 | 否 | Float | 字段值所占的比例<br>示例值： 0.9 |

## HighLightItem

日志内容高亮描述信息

被如下接口引用：GetAlarmLog、GetConfigurationTemplateApplyLog、SearchLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 高亮的日志Key<br>示例值： Message |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Values | 是 | 否 | Array of String | 高亮的语法<br>示例值： [ "finish request"] |

## PreviewLogStatistic

预览数据详情

被如下接口引用：CreateDataTransform、DescribeDataTransformPreviewDataInfo

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| DstTopicId | 否 | 否 | String | 目标日志主题<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| LogContent | 是 | 否 | String | 日志内容<br>示例值： {"AgentStartTime":"6557160"} |
| FailReason | 否 | 否 | String | 失败错误信息， 空字符串""表示正常<br>示例值： error |
| LineNum | 是 | 否 | Int64 | 行号。从0开始<br>示例值： 0 |
| Time | 否 | 否 | String | 日志时间，格式：2024-05-07 17:13:17.105 入参时无效 出参时有效，为日志中的时间格式<br>示例值： 2024-05-07 17:13:17.105 |
| DstTopicName | 否 | 是 | String | 目标topic-name<br>示例值： Name |
| DstLogSetName | 否 | 否 | String | 目标logsetname<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |

## RebuildIndexTaskInfo

索引重建任务信息

被如下接口引用：DescribeRebuildIndexTasks

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TaskId | 是 | 否 | String | 索引重建任务ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Status | 是 | 否 | Uint64 | 索引重建任务当前状态，0:索引重建任务已创建，1:创建索引重建资源，2:索引重建资源创建完成，3:重建中，4:暂停，5:重建索引成功，6:重建成功（可检索），7:重建失败，8:撤销，9:删除元数据和索引<br>示例值： 0 |
| StartTime | 是 | 否 | Uint64 | 重建任务开始时间戳<br>示例值： 1649670501 |
| EndTime | 是 | 否 | Uint64 | 重建任务结束时间戳<br>示例值： 1649670501 |
| RemainTime | 是 | 否 | Int64 | 重投预估剩余时间，单位秒<br>示例值： 10000 |
| CreateTime | 是 | 否 | Uint64 | 重建任务创建时间戳<br>示例值： 1649670501 |
| Progress | 是 | 否 | Float | 重投完成度，百分比<br>示例值： 0.1 |
| UpdateTime | 是 | 否 | Uint64 | 重建任务更新时间<br>示例值： 0 |
| StatusMessage | 是 | 否 | String | 附加状态描述信息（目前仅描述失败时失败原因）<br>示例值： message |

## ResourceInfo

需要检索的资源信息

被如下接口引用：DescribeExceptionResources

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 需要检索的topic_id<br>示例值： 8211a509-fee5-43dd-9ec0-6a6e7bc92552 |

## SendDetailItem

单个用户(Uin)的发送情况

被如下接口引用：CheckAlarmChannel

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Id | 是 | 否 | Uint64 | Uin或者Gin<br>示例值： 906685969 |
| Type | 是 | 否 | String | Uin或者Group或者Http或者Wecom<br>示例值： Uin |
| SuccessCount | 是 | 否 | Int64 | 成功数<br>示例值： 0 |

## DashboardSubscribeInfo

仪表盘订阅信息

被如下接口引用：DescribeDashboardSubscribes

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Id | 是 | 否 | Uint64 | 仪表盘订阅id。<br>示例值： 11 |
| Name | 是 | 否 | String | 仪表盘订阅名称。<br>示例值： 部门日报 |
| DashboardId | 是 | 否 | String | 仪表盘id。<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Cron | 是 | 否 | String | 仪表盘订阅时间。<br>示例值： "0 0 10 L * ?" |
| SubscribeData | 是 | 否 | DashboardSubscribeData | 仪表盘订阅数据。<br>示例值：查看 |
| CreateTime | 是 | 否 | String | 仪表盘订阅记录创建时间。<br>示例值： 2022-03-18 22:47:06 |
| UpdateTime | 是 | 否 | String | 仪表盘订阅记录更新时间。<br>示例值： 2022-03-18 22:47:06 |
| LastTime | 是 | 是 | String | 仪表盘订阅记录最后一次发送成功时间。<br>示例值： 2022-03-19 22:47:06 |
| Uin | 是 | 否 | Uint64 | 主账号Id。<br>示例值： 277001 |
| SubUin | 是 | 否 | Uint64 | 主账号下的子账号Id。<br>示例值： 277001 |
| LastStatus | 是 | 否 | String | 仪表盘订阅记录最后一次发送的状态。success：全部发送成功，fail：未发送， partialSuccess：部分发送成功。<br>示例值： "ok" |

## ExternalDataSourceInfo

外部数据源信息

被如下接口引用：DescribeExternalDataSources

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Id | 是 | 否 | String | 外部数据源ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Name | 是 | 是 | String | 名称<br>示例值： external_data_name |
| Describes | 是 | 是 | String | 备注<br>示例值： 外部数据源备注 |
| Datasource | 是 | 是 | Uint64 | 数据源类型：1表示Mysql数据源，2表示cos数据源<br>示例值： 1 |
| SubjectionTopicId | 是 | 是 | String | 隶属日志主题ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| SubjectionTopicName | 是 | 是 | String | 隶属日志主题名称<br>示例值： demo_topic_name |
| TopicId | 是 | 是 | String | 当前请求日志主题ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| LogsetId | 是 | 是 | String | 日志集<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| SQLInfo | 是 | 是 | ExternalDataSourceSQLInfo | mysql配置信息<br>示例值：查看 |
| COSInfo | 是 | 是 | ExternalDataSourceCOSInfo | cos配置信息<br>示例值：查看 |
| CreateTime | 是 | 是 | Int64 | 创建时间（秒级时间戳）<br>示例值： 1681990662 |
| UpdateTime | 是 | 是 | Int64 | 更新时间（秒级时间戳）<br>示例值： 1681990662 |

## KafkaRechargeInfo

Kafka导入配置信息

被如下接口引用：DescribeKafkaRecharges

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Id | 是 | 是 | String | Kafka数据订阅配置的ID。<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TopicId | 是 | 是 | String | 日志主题ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Name | 是 | 是 | String | Kafka导入任务名称<br>示例值： kafka导入任务 |
| KafkaType | 是 | 是 | Uint64 | 导入Kafka类型，0: CKafka，1: 用户自建Kafka<br>示例值： 0 |
| KafkaInstance | 是 | 是 | String | CKafka实例ID，KafkaType为0时必填<br>示例值： ckafka-le45va27 |
| ServerAddr | 是 | 是 | String | 服务地址<br>示例值： kafkaconsumer-region.cloud.demo.com |
| IsEncryptionAddr | 是 | 是 | Bool | ServerAddr是否为加密连接。true表示是加密连接，false表示不是加密连接。<br>示例值： true |
| Protocol | 是 | 否 | KafkaProtocolInfo | 加密访问协议，IsEncryptionAddr参数为true时必填<br>示例值：查看 |
| UserKafkaTopics | 是 | 是 | String | 用户需要导入的Kafka相关topic列表，多个topic之间使用半角逗号隔开<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| ConsumerGroupName | 是 | 是 | String | 用户Kafka消费组名称<br>示例值： cls_consumergroup_1 |
| Status | 是 | 是 | Int64 | 状态 status 1: 运行中, 2: 暂停<br>示例值： 1 |
| Offset | 是 | 是 | Int64 | 导入数据位置，-2:最早（默认），-1：最晚<br>示例值： 0 |
| CreateTime | 是 | 是 | String | 创建时间<br>示例值： 2021-11-08 14:40:22 |
| UpdateTime | 是 | 是 | String | 更新时间<br>示例值： 2021-11-08 14:40:22 |
| LogRechargeRule | 是 | 是 | LogRechargeRuleInfo | 日志导入规则<br>示例值：查看 |

## AlarmTarget

告警对象

被如下接口引用：CheckAlarmRule、CreateAlarm、ModifyAlarm

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 是 | String | 日志主题ID。<br>示例值： 8211a509-fee5-43dd-9ec0-6a6e7bc92552 |
| Query | 是 | 是 | String | 查询语句。<br>示例值： select count(*) as success_counts |
| Number | 是 | 是 | Int64 | 告警对象序号；从1开始递增。<br>示例值： 1 |
| StartTimeOffset | 是 | 是 | Int64 | 查询范围起始时间相对于告警执行时间的偏移，单位为分钟，取值为非正，最大值为0，最小值为-1440。<br>示例值： -5 |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| EndTimeOffset | 是 | 是 | Int64 | 查询范围终止时间相对于告警执行时间的偏移，单位为分钟，取值为非正，须大于StartTimeOffset，最大值为0，最小值为-1440。<br>示例值： 0 |
| LogsetId | 是 | 是 | String | 日志集ID。<br>示例值： 8211a509-fee5-43dd-9ec0-6a6e7bc92552 |
| SyntaxRule | 否 | 是 | Uint64 | 检索语法规则，默认值为0。 0：Lucene语法，1：CQL语法。 详细说明参见检索条件语法规则<br>示例值： 1 |

## ContainerStdoutInfo

自建k8s-容器标准输出信息

被如下接口引用：CreateConfigExtra、DescribeConfigExtras、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| AllContainers | 是 | 否 | Bool | 是否所有容器。true表示是所有容器，false表示不包含所有容器。<br>示例值： true |
| Container | 否 | 是 | String | container为空表所有的，不为空采集指定的容器<br>示例值： container_name |
| Namespace | 否 | 是 | String | namespace可以多个，用分隔号分割,例如A,B；为空或者没有这个字段，表示所有namespace<br>示例值： kube-system |
| IncludeLabels | 否 | 是 | Array of String | pod标签信息<br>示例值： ["key1=value1"] |
| WorkLoads | 否 | 是 | Array of ContainerWorkLoadInfo | 工作负载信息<br>示例值：查看 |
| ExcludeNamespace | 否 | 是 | String | 需要排除的namespace可以多个，用分隔号分割,例如A,B<br>示例值： kube-system |
| ExcludeLabels | 否 | 是 | Array of String | 需要排除的pod标签信息<br>示例值： ["key1=value1"] |
| ContainerFlag | 否 | 是 | Uint64 | 容器名称标记。必填字段，不填默认值为0。 默认值0，0:选中Container标记。 1:排除Container标记。<br>示例值： 0 |

## PartitionInfoForMonitor

kafka协议消费partition监控信息

被如下接口引用：DescribeKafkaConsumerTopics

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| PartitionId | 否 | 是 | Int64 | 分区id<br>示例值： 1 |

## WebCallback

回调地址

被如下接口引用：CheckAlarmChannel、CreateAlarmNotice、DescribeAlarmNotices、ModifyAlarmNotice

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Url | 是 | 否 | String | 回调地址，最大支持1024个字节。 也可使用WebCallbackId引用集成配置中的URL，此时该字段请填写为空字符串。<br>示例值： https://api.demo.com/cgi-bin/webhook/send?key=4c3cf1e0-a6e7-47f4-b264-5d2kjnbhjgx |
| Method | 否 | 是 | String | 回调方法。可选值： POST（默认值） PUT 注意： 参数CallbackType为Http时为必选，其它回调方式无需填写。<br>示例值： POST |
| Headers | 否 | 是 | Array of String | 请求头。<br>示例值： ["Content-Type: application/json"] |
| Body | 否 | 是 | String | 请求内容。CallbackType为Http时为必选。<br>示例值： htmlBody |
| CallbackType | 是 | 否 | String | 回调的类型。可选值：Http WeCom DingTalk Lark<br>示例值： Http |
| Index | 否 | 否 | Int64 | 序号。 入参无效。 出参有效。<br>示例值： 1 |
| StartTime | 否 | 是 | String | 允许接收信息的开始时间。当做入参时，默认值为00:00:00<br>示例值： 10:04:05 |
| EndTime | 否 | 是 | String | 允许接收信息的结束时间。当做入参时，默认值为23:59:59<br>示例值： 20:04:05 |

## ChannelTestResult

渠道检测结果

被如下接口引用：CheckAlarmChannel

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Index | 是 | 否 | Int64 | 序号<br>示例值： 1 |
| ErrorCode | 是 | 否 | Int64 | 错误码，0是正确，-1001表示无效请求。<br>示例值： 0 |
| ErrorMessage | 是 | 是 | String | 错误信息<br>示例值： err_message |
| SendTotal | 是 | 是 | SendDetail | 发送结果<br>示例值：查看 |

## FullTextInfo

全文索引配置

被如下接口引用：CreateIndex、DescribeIndex、DescribeIndexs、ModifyIndex

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| CaseSensitive | 是 | 否 | Bool | 是否大小写敏感。true表示大小写敏感，false代表大小写不敏感。<br>示例值： false |
| Tokenizer | 是 | 否 | String | 全文索引的分词符，其中的每个字符代表一个分词符； 仅支持英文符号、\n\t\r及转义符\； 注意：\n\t\r本身已被转义，直接使用双引号包裹即可作为入参，无需再次转义。使用API Explorer进行调试时请使用JSON参数输入方式，以避免\n\t\r被重复转义<br>示例值： @&? |
| ContainZH | 否 | 是 | Bool | 是否包含中文。true代表包含中文，false代表不包含中文。<br>示例值： false |

## RegexIndexInfo

获取K-V正则索引信息

被如下接口引用：GenKVRegex

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Start | 是 | 否 | Int64 | 起始位置<br>示例值： 0 |
| End | 是 | 否 | Int64 | 结束位置<br>示例值： 50 |

## ScheduledSqlTaskProcessInfo

ScheduledSql任务执行详情

被如下接口引用：DescribeScheduledSqlProcessInfo

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TaskId | 是 | 否 | String | 任务ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| ProcessId | 是 | 否 | String | 实例ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| ScheduledSqlContent | 是 | 否 | String | 加工语句<br>示例值： * |
| ProcessStartTime | 是 | 否 | String | 执行时间-开始时间<br>示例值： 2024-03-13 12:00:00 |
| ProcessEndTime | 是 | 否 | String | 执行时间-结束时间<br>示例值： 2024-03-13 12:01:00 |
| ProcessDuration | 是 | 否 | Int64 | 执行时间-耗时<br>示例值： 247 |
| TimeWindowStartTime | 是 | 否 | String | SQL时间窗口-开始时间<br>示例值： 2024-03-13 11:59:00 |
| TimeWindowEndTime | 是 | 否 | String | SQL时间窗口-结束时间<br>示例值： 2024-03-13 12:00:00 |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ReadLogCount | 是 | 否 | Uint64 | 处理数据量-输入行数<br>示例值： 1212 |
| WriteLogCount | 是 | 否 | Uint64 | 处理数据量-输出行数<br>示例值： 12 |
| Status | 是 | 否 | Int64 | 调度结果，1:运行中 2:成功 3:失败<br>示例值： 2 |
| SrcTopicId | 是 | 否 | String | 源topicid<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| StatusFailedMsg | 是 | 否 | String | 失败原因字段<br>示例值： fail_reason |

## TopicIdLogStatistic

数据加工日志主题统计数据详情

被如下接口引用：DescribeDataTransformProcessInfo

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 日志主题id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| ReadLines | 是 | 否 | Int64 | 读取的源日志主题的行数<br>示例值： 1 |
| WriteLines | 是 | 否 | Int64 | 加工后输出到目标日志主题的行数<br>示例值： 1 |
| FailedLines | 是 | 否 | Int64 | 加工失败的行数<br>示例值： 1 |
| FilterLines | 是 | 是 | Uint64 | 加工过滤的行数<br>示例值： 1 |

## AlarmAnalysisConfig

告警多维分析一些配置信息

被如下接口引用：CreateAlarm、DescribeAlarms、ModifyAlarm

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 否 | 是 | String | 键。支持以下key： SyntaxRule：语法规则，value支持 0：Lucene语法；1： CQL语法。 QueryIndex：执行语句序号。value支持 -1：自定义； 1：执行语句1； 2：执行语句2。 CustomQuery：检索语句。 QueryIndex为-1时有效且必填，value示例： "* |
| Value | 否 | 是 | String | 值。 键对应值如下： SyntaxRule：语法规则，value支持 0：Lucene语法；1： CQL语法。 QueryIndex：执行语句序号。value支持 -1：自定义； 1：执行语句1； 2：执行语句2。 CustomQuery：检索语句。 QueryIndex为-1时有效且必填，value示例： "* |

## UserConfigInfo

附加配置信息

被如下接口引用：DescribeUserConfig、ModifyUserConfig

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 附加配置key<br>示例值： key |
| Value | 是 | 否 | String | 附加配置内容<br>示例值： value |

## KeyRegexInfo

需要过滤日志的key，及其对应的regex

被如下接口引用：CreateConfig、CreateConfigExtra、CreateCosRecharge、DescribeAgentConfigs、DescribeConfigExtras、DescribeConfigs、DescribeCosRecharges、DescribeMachineGroupConfigs、ModifyConfig、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 需要过滤日志的key<br>示例值： response_code |
| Regex | 是 | 否 | String | key对应的过滤规则regex<br>示例值： 400 |

## CollectConfig

采集配置信息

被如下接口引用：CreateConfigExtra、DescribeConfigExtras、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Name | 是 | 否 | String | 指定采集类型的采集配置名称信息。 当CollectInfo中Type为0：表示元数据配置，name为元数据名称。 目前支持"container_id"，"container_name"，"image_name"，"namespace"，"pod_uid"，"pod_name"，"pod_ip"。 当CollectInfo中Type为1：指定pod label，name为指定pod label名称。<br>示例值： pod_name |

## ParquetKeyInfo

Parquet内容描述

被如下接口引用：CreateShipper、DescribeShipperPreview、DescribeShippers、ModifyShipper

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| KeyName | 是 | 否 | String | 键值名称<br>示例值： content |
| KeyType | 是 | 否 | String | 数据类型，目前支持6种类型：string、boolean、int32、int64、float、double<br>示例值： string |
| KeyNonExistingField | 是 | 是 | String | 解析失败赋值信息<br>示例值： key1 |

## AnalysisDimensional

多维分析的分析维度

被如下接口引用：CreateAlarm、DescribeAlarms、ModifyAlarm

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Name | 是 | 是 | String | 分析名称<br>示例值： status 502 pv |
| Type | 是 | 是 | String | 分析类型：query，field ，original<br>示例值： query |
| Content | 是 | 是 | String | 分析内容<br>示例值： * |
| ConfigInfo | 否 | 否 | Array of AlarmAnalysisConfig | 多维分析配置。 当Analysis的Type字段为query（自定义）时，支持{ "Key": "SyntaxRule", // 语法规则 "Value": "1" //0：Lucene语法 ，1： CQL语法 } 当Analysis的Type字段为field（top5）时, 支持{ "Key": "QueryIndex", "Value": "-1" // -1：自定义， 1：执行语句1， 2：执行语句2 },{ "Key": "CustomQuery", //检索语句。 QueryIndex为-1时有效且必填 "Value": "* |

## KafkaConsumerInfo

kafka协议消费主题信息

被如下接口引用：DescribeKafkaConsumerTopics

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 是 | String | 消费主题<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| ToTopicId | 是 | 是 | String | kafka的主题id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| InstanceId | 是 | 是 | String | kafka实例id<br>示例值： ckafka-le45va27 |
| ConsumerGroupList | 是 | 是 | Array of ConsumerGroupInfoForMonitor | 消费组信息<br>示例值：查看 |

## MachineInfo

机器状态信息

被如下接口引用：DescribeMachines

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Ip | 是 | 否 | String | 机器的IP<br>示例值： 10.0.0.1 |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Status | 是 | 否 | Uint64 | 机器状态，0:异常，1:正常<br>示例值： 0 |
| OfflineTime | 是 | 否 | String | 机器离线时间，空为正常，异常返回具体时间<br>示例值： 2021-01-12 20:14:10 |
| AutoUpdate | 是 | 否 | Int64 | 机器是否开启自动升级。0:关闭，1:开启<br>示例值： 0 |
| Version | 是 | 否 | String | 机器当前版本号。<br>示例值： 2.1.0 |
| UpdateStatus | 是 | 否 | Int64 | 机器升级功能状态。 0：升级成功；1：升级中；-1：升级失败。<br>示例值： 0 |
| ErrCode | 是 | 否 | Int64 | 机器升级结果标识。 0：成功；1200：升级成功；其他值表示异常。<br>示例值： 0 |
| ErrMsg | 是 | 否 | String | 机器升级结果信息。 “ok”：成功；“update success”：升级成功；其他值为失败原因。<br>示例值： OK |
| InstanceID | 否 | 是 | String | 机器实例ID<br>示例值： loglistener-1462d334-0fb6-5f36-28d8-209e1b17ea2d |

## ConsumerContent

投递任务出入参 Content

被如下接口引用：CreateConsumer、DescribeConsumer、DescribeConsumerPreview、ModifyConsumer

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| EnableTag | 是 | 是 | Bool | 是否投递 TAG 信息。 当EnableTag为true时，表示投递TAG元信息。<br>示例值： false |
| MetaFields | 是 | 是 | Array of String | 需要投递的元数据列表，目前仅支持：SOURCE，FILENAME，TIMESTAMP，HOSTNAME__和__PKGID<br>示例值： ["__SOURCE__", "__FILENAME__", "__TIMESTAMP__"] |
| TagJsonNotTiled | 否 | 是 | Bool | 当EnableTag为true时，必须填写TagJsonNotTiled字段。 TagJsonNotTiled用于标识tag信息是否json平铺。 TagJsonNotTiled为true时不平铺，示例： TAG信息：{"TAG":{"fieldA":200,"fieldB":"text"}} 不平铺：{"TAG":{"fieldA":200,"fieldB":"text"}} TagJsonNotTiled为false时平铺，示例： TAG信息：{"TAG":{"fieldA":200,"fieldB":"text"}} 平铺：{"TAG.fieldA":200,"TAG.fieldB":"text"}<br>示例值： false |
| TimestampAccuracy | 否 | 是 | Int64 | 投递时间戳精度，可选项 [1：秒；2：毫秒] ，默认是1。<br>示例值： 1 |
| JsonType | 否 | 是 | Int64 | 投递Json格式。 JsonType为0：和原始日志一致，不转义。示例： 日志原文：{"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} 投递到Ckafka：{"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} JsonType为1：转义。示例： 日志原文：{"a":"aa", "b":{"b1":"b1b1", "c1":"c1c1"}} 投递到Ckafka：{"a":"aa","b":"{"b1":"b1b1", "c1":"c1c1"}"}<br>示例值： 0 |

## LogRechargeRuleInfo

日志导入规则

被如下接口引用：CreateKafkaRecharge、DescribeKafkaRecharges、ModifyKafkaRecharge、PreviewKafkaRecharge

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| RechargeType | 是 | 否 | String | 导入类型，支持json_log：json格式日志，minimalist_log: 单行全文，fullregex_log: 单行完全正则<br>示例值： json_log |
| LogRegex | 否 | 是 | String | 整条日志匹配规则，只有RechargeType为fullregex_log时有效<br>示例值： (\w+).* |
| UnMatchLogSwitch | 否 | 否 | Bool | 解析失败日志是否上传，true表示上传，false表示不上传<br>示例值： true |
| UnMatchLogKey | 否 | 是 | String | 解析失败日志的键名称<br>示例值： LogParseFailure |
| UnMatchLogTimeSrc | 否 | 是 | Uint64 | 解析失败日志时间来源，0: 系统当前时间，1: Kafka消息时间戳<br>示例值： 0 |
| EncodingFormat | 是 | 否 | Uint64 | 解析编码格式，0: UTF-8（默认值），1: GBK<br>示例值： 0 |
| DefaultTimeSwitch | 是 | 否 | Bool | 使用默认时间，true：开启（默认值）， flase：关闭<br>示例值： true |
| DefaultTimeSrc | 否 | 是 | Uint64 | 默认时间来源，0: 系统当前时间，1: Kafka消息时间戳<br>示例值： 0 |
| TimeKey | 否 | 是 | String | 时间字段<br>示例值： TIMESTAMP |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TimeRegex | 否 | 是 | String | 时间提取正则表达式<br>示例值： (\d{4})-(\d{2})-(\d{2}) (\d{2}):(\d{2}):(\d{2}) |
| TimeFormat | 否 | 是 | String | 时间字段格式<br>示例值： %Y-%m-%d %H:%M:%S |
| TimeZone | 否 | 是 | String | 时间字段时区<br>示例值： UTC+08:00 |
| Metadata | 否 | 是 | Array of String | 元数据信息，Kafka导入支持kafka_topic,kafka_partition,kafka_offset,kafka_timestamp<br>示例值： ["kafka_topic","kafka_partition","kafka_offset","kafka_timestamp"] |
| Keys | 否 | 是 | Array of String | 日志Key列表，RechargeType为full_regex_log时必填<br>示例值： ["name","address"] |

## DashboardSubscribeData

仪表盘订阅相关数据

被如下接口引用：CreateDashboardSubscribe、DescribeDashboardSubscribes、ModifyDashboardSubscribe、SearchDashboardSubscribe

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| DashboardTime | 否 | 是 | Array of String | 仪表盘订阅时间，为空标识取仪表盘默认的时间。<br>示例值： ["now-5m","now"] |
| StyleLayout | 否 | 是 | Uint64 | 仪表盘样式布局。0：网格布局，1：单列布局。<br>示例值： 0 |
| TemplateVariables | 否 | 是 | Array of DashboardTemplateVariable | 仪表盘订阅模板变量。<br>示例值：查看 |
| NoticeModes | 是 | 否 | Array of DashboardNoticeMode | 仪表盘订阅通知方式。<br>示例值：查看 |
| Timezone | 否 | 是 | String | 时区。参考：https://en.wikipedia.org/wiki/List_of_tz_database_time_zones<br>示例值： zone |
| SubscribeLanguage | 否 | 是 | String | 语言。 zh 中文、en`英文。<br>示例值： zh |
| JumpDomain | 否 | 是 | String | 调用链接域名。http:// 或者 https:// 开头，不能/结尾<br>示例值： https://console.cloud.demo.com |

## FunctionInfo

数据加工函数基本信息

被如下接口引用：DescribeFunctions

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| FuncName | 是 | 否 | String | 函数名称<br>示例值： fields_set |
| FuncNameDesc | 是 | 否 | String | 函数描述<br>示例值： 函数功能及使用说明 |
| FuncUseDesc | 是 | 否 | String | 用来说明函数功能<br>示例值： 函数语法描述 |
| FuncSyntaxDesc | 是 | 否 | String | 语法描述<br>示例值： 函数使用示例 |
| FuncDemo | 是 | 否 | String | demo展示<br>示例值： 例1使用逗号作为分隔符，提取日志中的值。 |
| FuncType | 是 | 否 | String | 函数类型<br>示例值： line_function |
| IsVariadic | 是 | 否 | Bool | true是可变长度参数的加工函数<br>示例值： true |
| MaxArgumentSize | 是 | 否 | Int64 | 可变参的参数个数最大限制<br>示例值： 10 |
| ReturnType | 是 | 否 | String | 函数返回结果类型，用来校验嵌套函数中，返回结果是否和函数参数类型匹配。 不同的func_type返回的对象类型不同 string/int/bool/condition/func<br>示例值： string |
| Arguments | 是 | 否 | Array of FunctionArgument | 函数参数描述<br>示例值：查看 |

## ConfigurationContentInfo

配置模板投递日志的内容格式配置

被如下接口引用：CreateConfigurationTemplate、DescribeConfigurationTemplates、ModifyConfigurationTemplate

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Format | 是 | 否 | String | 内容格式，只支持json<br>示例值： json |
| Json | 否 | 是 | JsonInfo | json格式内容描述<br>示例值：查看 |

## Ckafka

CKafka的描述-需要投递到的kafka信息

被如下接口引用：CreateConsumer、DescribeConsumer、ModifyConsumer

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Vip | 是 | 是 | String | Ckafka 的 Vip<br>示例值： 10.10.2.42 |
| Vport | 是 | 是 | String | Ckafka 的 Vport<br>示例值： 9092 |
| InstanceId | 是 | 是 | String | Ckafka 的 InstanceId<br>示例值： ckafka-8j4robetb |
| InstanceName | 是 | 是 | String | Ckafka 的 InstanceName<br>示例值： kafka协议消费监控用 |
| TopicId | 是 | 是 | String | Ckafka 的 TopicId<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TopicName | 是 | 是 | String | Ckafka 的 TopicName<br>示例值： mrxy-business |

## LogContextInfo

日志上下文信息

被如下接口引用：DescribeLogContext

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Source | 是 | 否 | String | 日志来源设备<br>示例值： 100.105.60.255 |
| Filename | 是 | 否 | String | 采集路径<br>示例值： /usr/local/services/cls_cgi_api3-1.0/log/cls_cgi.log.20210425 |
| Content | 是 | 否 | String | 日志内容<br>示例值： {"CONTENT": "[INFO] [2020-11-23 18:34:14.983] [1da50ffe-4e87-42a6-be05-dc9e200cd284] [LogFilter.java] Query:  {}"} |
| PkgId | 是 | 否 | String | 日志包序号<br>示例值： 528C1318606EFEB8-1A0 |
| PkgLogId | 是 | 否 | Int64 | 日志包内一条日志的序号<br>示例值： 196609 |
| BTime | 是 | 否 | Int64 | 日志时间戳<br>示例值： 2021-04-25 14:25:00.000 |
| HostName | 是 | 是 | String | 日志来源主机名称<br>示例值： hostName-1 |
| RawLog | 是 | 是 | String | 原始日志(仅在日志创建索引异常时有值)<br>示例值： {"content":"log"} |
| IndexStatus | 是 | 是 | String | 日志创建索引异常原因(仅在日志创建索引异常时有值)<br>示例值： IndexAllFail:fields like 'a' and 'a.b' can not appear in the same topic |

## ShipperInfo

投递规则

被如下接口引用：DescribeShippers

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ShipperId | 是 | 否 | String | 投递规则ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Bucket | 是 | 否 | String | 投递的bucket地址<br>示例值： bucket_name |
| Prefix | 是 | 否 | String | 投递的前缀目录<br>示例值： var/log |
| ShipperName | 是 | 否 | String | 投递规则的名字<br>示例值： shipper_name |
| Interval | 是 | 否 | Uint64 | 投递的时间间隔，单位 秒<br>示例值： 300 |
| MaxSize | 是 | 否 | Uint64 | 投递的文件的最大值，单位 MB<br>示例值： 256 |
| Status | 是 | 否 | Bool | 是否生效。true表示生效，false表示失效。<br>示例值： true |
| FilterRules | 是 | 是 | Array of FilterRuleInfo | 投递日志的过滤规则<br>示例值：查看 |
| Partition | 是 | 否 | String | 投递日志的分区规则，支持strftime的时间格式表示<br>示例值： %Y%m%d |
| Compress | 是 | 是 | CompressInfo | 投递日志的压缩配置<br>示例值：查看 |
| Content | 是 | 是 | ContentInfo | 投递日志的内容格式配置<br>示例值：查看 |
| CreateTime | 是 | 否 | String | 投递日志的创建时间<br>示例值： 2024-09-19 16:15:35 |
| FilenameMode | 是 | 是 | Uint64 | 投递文件命名配置，0：随机数命名，1：投递时间命名，默认0（随机数命名）<br>示例值： 0 |
| StartTime | 是 | 是 | Int64 | 投递数据范围的开始时间点<br>示例值： 1670642281 |
| EndTime | 是 | 是 | Int64 | 投递数据范围的结束时间点<br>示例值： 1670642281 |
| Progress | 是 | 是 | Float | 历史数据投递的进度（仅当用户选择的数据内中历史数据时才有效）<br>示例值： 5.6 |
| RemainTime | 是 | 是 | Int64 | 历史数据全部投递完成剩余的时间（仅当用户选择的数据中有历史数据时才有效）<br>示例值： 23 |
| CustomUin | 是 | 是 | Uint64 | 跨账号uin<br>示例值： 1670641097 |
| HistoryStatus | 是 | 是 | Int64 | 历史任务状态： 0：实时任务 1：任务准备中 2：任务运行中 3：任务运行异常 4：任务运行结束<br>示例值： 1 |

## ConsumerGroupInfoForMonitor

kafka协议消费消费组监控接口

被如下接口引用：DescribeKafkaConsumerTopics

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ConsumerGroupName | 是 | 是 | String | 消费组名称<br>示例值： cls_consumer_group |
| PartitionListForMonitor | 是 | 是 | Array of PartitionInfoForMonitor | 消费组监控信息<br>示例值：查看 |

## ContainerWorkLoadInfo

自建k8s-工作负载信息

被如下接口引用：CreateConfigExtra、DescribeConfigExtras、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Container | 否 | 是 | String | 容器名<br>示例值： container_name |
| Kind | 是 | 否 | String | 工作负载的类型<br>示例值： daemonset |
| Name | 是 | 否 | String | 工作负载的名称<br>示例值： tke-log-agent |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Namespace | 否 | 是 | String | 命名空间<br>示例值： kube-system |
| ContainerFlag | 否 | 是 | Uint64 | 容器名称标记。必填字段，不填默认值为0。 0:选中Container标记。默认值 1:排除Container标记。<br>示例值： 0 |

## MonitorTime

告警策略中监控任务的执行时间点

被如下接口引用：CheckAlarmRule、CreateAlarm、DescribeAlarms、ModifyAlarm

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Type | 是 | 否 | String | 执行周期， 可选值：Period、Fixed、Cron。 Period：固定频率 Fixed：固定时间 Cron：Cron表达式<br>示例值： Period |
| Time | 是 | 否 | Int64 | 执行的周期，或者定制执行的时间节点。单位为分钟，取值范围为1~1440。 当type为Period,Fixed时，time字段生效。<br>示例值： 1 |

## DataTransformFailureInfo

数据加工日志失败信息

被如下接口引用：DescribeDataTransformFailLogInfo

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| LogContent | 是 | 否 | String | 源日志<br>示例值： content |
| Reason | 是 | 否 | String | 加工失败原因<br>示例值： -110 |

## ExceptionResources

异常资源信息列表

被如下接口引用：DescribeExceptionResources

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 日志主题Id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| IndexExceptions | 否 | 否 | Array of IndexException | 索引配置异常列表<br>示例值：查看 |
| ConfigExceptions | 否 | 否 | Array of ConfigException | 采集配置异常列表<br>示例值：查看 |
| UploadLogExceptions | 否 | 否 | Array of UploadLogException | 日志上传异常列表<br>示例值：查看 |

## ExternalDataSourceCOSInfo

外部表COS信息

被如下接口引用：CreateExternalDataSource、DescribeExternalDataSourcePreview、DescribeExternalDataSources、ModifyExternalDataSource

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Region | 是 | 否 | String | COS地域<br>示例值： region |
| Bucket | 是 | 否 | String | COS存储桶<br>示例值： bucket-1 |
| FileAddress | 是 | 否 | String | COS文件地址(CSV格式)，仅支持CSV格式文件<br>示例值： /region-open-kkk-1730194324/%Y/%m/%d/%H/ |
| CompressType | 否 | 是 | Uint64 | 压缩方式 0:不压缩;1:gzip; 2:lzop;3:snappy<br>示例值： 0 |
| FieldsInfo | 是 | 否 | Array of FieldInfo | COS文件(CSV格式)内容 表头字段与字段类型信息<br>示例值：查看 |

## Filter

过滤器

被如下接口引用：DescribeAlarmNotices、DescribeAlarms、DescribeAlertRecordHistory、DescribeConfigExtras、DescribeConfigs、DescribeConfigurationTemplates、DescribeDashboardSubscribes、DescribeDashboards、DescribeDataTransformInfo、DescribeExternalDataSources、DescribeFolders、DescribeIndexs、DescribeLogsets、DescribeMachineGroups、DescribeShippers、DescribeTopics

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 需要过滤的字段。<br>示例值： taskId |
| Values | 是 | 否 | Array of String | 需要过滤的值。<br>示例值： ["57f5808c-4a55-11eb-b378-0242ac130002"] |

## ScheduledSqlTaskInfo

ScheduledSql任务详情

被如下接口引用：DescribeScheduledSqlInfo

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TaskId | 是 | 否 | String | ScheduledSql任务id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Name | 是 | 否 | String | ScheduledSql任务名称<br>示例值： task |
| SrcTopicId | 是 | 否 | String | 源日志主题id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| SrcTopicName | 是 | 否 | String | 源日志主题名称<br>示例值： topic_name |
| DstResource | 是 | 否 | ScheduledSqlResouceInfo | 定时SQL分析目标主题<br>示例值：查看 |
| CreateTime | 是 | 否 | String | 任务创建时间<br>示例值： 2023-07-28 11:36:00 |
| UpdateTime | 是 | 否 | String | 任务更新时间<br>示例值： 2023-07-28 11:36:00 |
| Status | 是 | 否 | Int64 | 任务状态，1:运行 2:停止 3:异常-找不到源日志主题 4:异常-找不到目标主题 5: 访问权限问题 6:内部故障 7:其他故障<br>示例值： 2 |
| EnableFlag | 是 | 否 | Int64 | 任务启用状态，1开启, 2关闭<br>示例值： 1 |
| ScheduledSqlContent | 是 | 否 | String | 查询语句<br>示例值： select count(*) |
| ProcessStartTime | 是 | 否 | String | 调度开始时间<br>示例值： 2023-07-28 11:36:00 |
| ProcessType | 是 | 否 | Int64 | 调度类型，1:持续运行 2:指定时间范围<br>示例值： 1 |
| ProcessEndTime | 是 | 否 | String | 调度结束时间，当process_type=2时为必传字段<br>示例值： 2023-07-28 11:36:00 |
| ProcessPeriod | 是 | 否 | Int64 | 调度周期(分钟)<br>示例值： 5 |
| ProcessTimeWindow | 是 | 否 | String | 查询的时间窗口. @m-15m, @m，意为近15分钟<br>示例值： @m-15m, @m |
| ProcessDelay | 是 | 否 | Int64 | 执行延迟(秒)<br>示例值： 5 |
| SrcTopicRegion | 是 | 否 | String | 源topicId的地域信息<br>示例值： region |
| SyntaxRule | 是 | 是 | Uint64 | 语法规则，0：Lucene语法，1：CQL语法<br>示例值： 0 |

## Tag

创建资源实例时同时绑定的标签对说明

被如下接口引用：CreateAlarm、CreateAlarmNotice、CreateDashboard、CreateLogset、CreateMachineGroup、CreateTopic、DescribeAlarmNotices、DescribeAlarms、DescribeConfigMachineGroups、DescribeDashboards、DescribeLogsets、DescribeMachineGroups、DescribeTopics、ModifyAlarm、ModifyAlarmNotice、ModifyDashboard、ModifyLogset、ModifyMachineGroup、ModifyTopic

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 是 | String | 标签键<br>示例值： name |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Value | 是 | 是 | String | 标签值<br>示例值： 业务A |

## NoticeReceiver

告警通知接收者信息

被如下接口引用：CheckAlarmChannel、CreateAlarmNotice、DescribeAlarmNotices、ModifyAlarmNotice

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ReceiverType | 是 | 否 | String | 接受者类型。可选值： Uin - 用户ID Group - 用户组ID 暂不支持其余接收者类型<br>示例值： Group |
| ReceiverIds | 是 | 否 | Array of Int64 | 接收者。 当ReceiverType为Uin时，ReceiverIds的值为用户uid。子用户信息查询 当ReceiverType为Group时，ReceiverIds的值为用户组id。CAM用户组<br>示例值： [1736994580] |
| ReceiverChannels | 是 | 否 | Array of String | 通知接收渠道。 Email - 邮件 Sms - 短信 WeChat - 微信 Phone - 电话<br>示例值： ['Sms'] |
| StartTime | 是 | 否 | String | 允许接收信息的开始时间。格式：15:04:05。必填<br>示例值： 10:04:05 |
| EndTime | 是 | 否 | String | 允许接收信息的结束时间。格式：15:04:05。必填<br>示例值： 20:04:05 |
| Index | 否 | 否 | Int64 | 位序。 入参时无效。 出参时有效。<br>示例值： 1 |

## ContainerFileInfo

自建k8s-容器文件路径信息

被如下接口引用：CreateConfigExtra、DescribeConfigExtras、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Namespace | 是 | 否 | String | namespace可以多个，用分隔号分割,例如A,B<br>示例值： kube-system |
| Container | 是 | 否 | String | 容器名称<br>示例值： container1 |
| LogPath | 是 | 否 | String | 日志文件夹<br>示例值： /var/log/tmep |
| FilePattern | 是 | 否 | String | 日志名称<br>示例值： *.log |
| IncludeLabels | 否 | 是 | Array of String | pod标签信息<br>示例值： ["key1=value1"] |
| WorkLoad | 否 | 是 | ContainerWorkLoadInfo | 工作负载信息<br>示例值：查看 |
| ExcludeNamespace | 否 | 是 | String | 需要排除的namespace可以多个，用分隔号分割,例如A,B<br>示例值： kube-system |
| ExcludeLabels | 否 | 是 | Array of String | 需要排除的pod标签信息<br>示例值： ["key1=value1"] |
| ContainerFlag | 否 | 是 | Uint64 | 容器名称标记。必填字段，不填默认值为0。 默认值0，0:选中Container标记。 1:排除Container标记。<br>示例值： 0 |

## AlarmClassification

告警分类信息

被如下接口引用：CreateAlarm、DescribeAlarms、ModifyAlarm

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 分类键<br>示例值： key_sample |
| Value | 是 | 否 | String | 分类值<br>示例值： value_sample |

## AlarmTargetInfo

日志告警监控对线

被如下接口引用：DescribeAlarms

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| LogsetId | 是 | 否 | String | 日志集ID。<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| LogsetName | 是 | 否 | String | 日志集名称。<br>示例值： LogsetDemo |
| TopicId | 是 | 否 | String | 日志主题ID。<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| TopicName | 是 | 否 | String | 日志主题名称。<br>示例值： TopicDemo |
| Query | 是 | 否 | String | 查询语句。<br>示例值： "mi" OR "CommunicationExcep |
| Number | 是 | 否 | Int64 | 告警对象序号。<br>示例值： 1 |
| StartTimeOffset | 是 | 否 | Int64 | 查询范围起始时间相对于告警执行时间的偏移，单位为分钟，取值为非正，最大值为0，最小值为-1440。<br>示例值： -240 |
| EndTimeOffset | 是 | 否 | Int64 | 查询范围终止时间相对于告警执行时间的偏移，单位为分钟，取值为非正，须大于StartTimeOffset，最大值为0，最小值为-1440。<br>示例值： 0 |
| SyntaxRule | 是 | 否 | Uint64 | 检索语法规则，默认值为0。 0：Lucene语法，1：CQL语法。 详细说明参见检索条件语法规则<br>示例值： 0 |
| QueryInteractiveConfig | 否 | 否 | String | 查询语句交互模式配置信息。<br>示例值： {"Value":{}} |

## SendDetail

渠道检测发送详情

被如下接口引用：CheckAlarmChannel

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Sms | 是 | 是 | Array of SendDetailItem | 发送次数<br>示例值：查看 |
| Email | 是 | 是 | Array of SendDetailItem | 发送次数<br>示例值：查看 |
| WeChat | 是 | 是 | Array of SendDetailItem | 发送次数<br>示例值：查看 |
| Phone | 是 | 是 | Array of SendDetailItem | 发送次数<br>示例值：查看 |
| Callback | 是 | 是 | Array of SendDetailItem | 发送次数<br>示例值：查看 |

## CsvInfo

csv内容描述

被如下接口引用：CreateShipper、DescribeShipperPreview、DescribeShippers、ModifyShipper

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| PrintKey | 是 | 否 | Bool | csv首行是否打印key。true表示首行打印key，false表示首行不打印key<br>示例值： true |
| Keys | 是 | 是 | Array of String | 每列key的名字<br>示例值： ["SOURCE","FILENAME","HOSTNAME"] |
| Delimiter | 是 | 否 | String | 各字段间的分隔符<br>示例值：， |
| EscapeChar | 是 | 否 | String | 若字段内容中包含分隔符，则使用该转义符包裹改字段，只能填写单引号、双引号、空字符串<br>示例值： ' |
| NonExistingField | 是 | 否 | String | 对于上面指定的不存在字段使用该内容填充<br>示例值： not_exist |

## MachineGroupMetadataInfo

机器组元数据信息

被如下接口引用：DescribeAgentMachineGroupMetadata

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| MetaTags | 是 | 否 | Array of MetaTagInfo | 机器组元数据列表<br>示例值：查看 |

## CollectInfo

采集配置信息

被如下接口引用：CreateConfigExtra、DescribeConfigExtras、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| CollectConfigs | 否 | 是 | Array of CollectConfig | 指定采集类型的采集配置信息。 当Type为0时，CollectConfigs不允许为空。 当Type为1时，CollectConfigs为空时，表示选择所有Pod Label；否则CollectConfigs为指定Pod Label。<br>示例值：查看 |
| Type | 是 | 是 | Uint64 | 采集类型，必填字段。 0：元数据配置。 1：指定Pod Label。<br>示例值： 0 |

## KeyValueInfo

键值或者元字段索引的字段信息

被如下接口引用：CreateIndex、DescribeIndex、DescribeIndexs、ModifyIndex

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 需要配置键值或者元字段索引的字段名称，仅支持字母、数字、下划线和-./@，且不能以下划线开头 注意： 1，元字段（tag）的Key无需额外添加__TAG__.前缀，与上传日志时对应的字段Key一致即可，云控制台展示时将自动添加__TAG__.前缀 2，键值索引（KeyValue）及元字段索引（Tag）中的Key总数不能超过300 3，Key的层级不能超过10层，例如a.b.c.d.e.f.g.h.j.k 4，不允许同时包含json父子级字段，例如a及a.b<br>示例值： address |
| Value | 是 | 否 | ValueInfo | 字段的索引描述信息<br>示例值：查看 |

## KafkaInfo

消费kafka相关信息

被如下接口引用：DescribeKafkaConsume、OpenKafkaConsume

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicName | 是 | 否 | String | 可消费topic名称<br>示例值： topic_name |
| UserName | 是 | 否 | String | ACL模式用户名<br>示例值： user_name |
| Password | 是 | 否 | String | ACL模式密码<br>示例值： password |
| InstanceId | 是 | 否 | String | 可消费kafka实例ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| BootstrapServers | 是 | 否 | String | SASL接入点信息<br>示例值： 1.1.1.1 |

## SearchLogErrors

多日志主题检索错误信息

被如下接口引用：DescribeLogHistogram、SearchLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 是 | String | 日志主题ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| ErrorCode | 是 | 是 | Int64 | 错误码。（已废弃）<br>示例值： -45000 |
| ErrorMsg | 是 | 是 | String | 错误信息<br>示例值： SyntaxError [field: key, can not search on this field, since it is not indexed] |
| ErrorCodeStr | 是 | 是 | String | 错误码<br>示例值： FailedOperation.QueryError |

## TopicIndexInfo

topic的索引配置明细

被如下接口引用：DescribeIndexs

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 日志主题id<br>示例值： e8722b2a-ab75-47f8-8530-8ac542133654 |
| Status | 是 | 是 | Bool | 索引是否生效。true表示索引生效，false表示索引失效。<br>示例值： false |
| Rule | 是 | 是 | RuleInfo | 索引配置信息 注意：此字段可能返回 null，表示取不到有效值。<br>示例值：查看 |
| ModifyTime | 是 | 否 | String | 索引修改时间，初始值为索引创建时间。<br>示例值： 2021-01-13 21:00:08 |
| TopicName | 是 | 否 | String | 日志主题名称<br>示例值： name1 |
| LogsetId | 是 | 否 | String | 日志集id<br>示例值： oooo |
| LogsetName | 是 | 否 | String | 日志集名称<br>示例值： name2 |
| IncludeInternalFields | 是 | 是 | Bool | 内置保留字段（FILENAME，HOSTNAME__及__SOURCE）是否包含至全文索引 false:不包含 true:包含<br>示例值： false |
| MetadataFlag | 是 | 是 | Uint64 | 元数据字段（前缀为__TAG__的字段）是否包含至全文索引 0:仅包含开启键值索引的元数据字段 1:包含所有元数据字段 2:不包含任何元数据字段<br>示例值： 0 |
| CoverageField | 是 | 是 | String | 自定义日志解析异常存储字段。<br>示例值： RAWLOG |

## EventLog

Windows事件日志采集配置

被如下接口引用：CreateConfig、CreateConfigExtra、CreateCosRecharge、DescribeAgentConfigs、DescribeConfigExtras、DescribeConfigs、DescribeCosRecharges、DescribeMachineGroupConfigs、ModifyConfig、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| EventChannel | 是 | 否 | String | 事件通道，支持Application，Security，Setup，System，ALL<br>示例值： Application |
| TimeType | 是 | 否 | Uint64 | 时间类型，1:用户自定义，2:当前时间<br>示例值： 1 |
| Timestamp | 否 | 否 | Uint64 | 时间，用户选择自定义时间类型时，需要指定时间<br>示例值： 1736994580 |
| EventIDs | 否 | 否 | Array of String | 事件ID过滤列表 选填，为空表示不做过滤 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<br>示例值： 1-200,-100 |

## HistogramInfo

直方图详细信息

被如下接口引用：DescribeLogHistogram

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Count | 是 | 否 | Int64 | 统计周期内的日志条数<br>示例值： 15 |
| BTime | 是 | 否 | Int64 | 按 period 取整后的 unix timestamp： 单位毫秒<br>示例值： 2021-04-25 14:25:00.000 |

## ExceptionInfo

异常问题请求结构

被如下接口引用：DescribeExceptionResources

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Enable | 否 | 否 | Bool | 表示该异常问题是否需要展示。true表示需要展示异常问题，false表示不需要展示异常问题<br>示例值： false |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ExceptionId | 否 | 否 | Int64 | 异常资源ID： 1：索引配置异常 2：采集配置异常 3：日志上传异常<br>示例值： 1 |

## ExportInfo

日志导出信息

被如下接口引用：DescribeExports

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： ee20bb16-3025-4048-b81a-dd436373062e |
| ExportId | 是 | 否 | String | 日志导出任务ID<br>示例值： export-57196a6a-7622-47be-bc92-d2ebea959a0f |
| Query | 是 | 否 | String | 日志导出查询语句<br>示例值： status:200 |
| FileName | 是 | 否 | String | 日志导出文件名<br>示例值： log_2075178708_ee20bb16-3025-4048-b81a-dd436373062f_20210107_571 |
| FileSize | 是 | 否 | Uint64 | 日志文件大小<br>示例值： 0 |
| Order | 是 | 否 | String | 日志导出时间排序。asc表示正序排序，desc表示倒序排序。<br>示例值： desc |
| Format | 是 | 否 | String | 日志导出格式。json表示日志导出格式为json。<br>示例值： json |
| Count | 是 | 否 | Uint64 | 日志导出数量<br>示例值： 100 |
| Status | 是 | 否 | String | 日志下载状态。Processing:导出正在进行中，Completed:导出完成，Failed:导出失败，Expired:日志导出已过期(三天有效期), Queuing 排队中<br>示例值： Expired |
| From | 是 | 否 | Int64 | 日志导出起始时间<br>示例值： 1607499207000 |
| To | 是 | 否 | Int64 | 日志导出结束时间<br>示例值： 1607499208000 |
| CosPath | 是 | 否 | String | 日志导出路径<br>示例值： /region-open-kkk-1730194324/%Y/%m/%d/%H/ |
| CreateTime | 是 | 否 | String | 日志导出创建时间<br>示例值： 2020-08-08 12:12:12 |
| SyntaxRule | 是 | 否 | Uint64 | 语法规则。 默认值为0。 0：Lucene语法，1：CQL语法。<br>示例值： 1 |

## AlarmNotice

告警通知模板类型

被如下接口引用：DescribeAlarmNotices

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Name | 是 | 否 | String | 告警通知模板名称。<br>示例值： alarm |
| Type | 是 | 否 | String | 告警模板的类型。可选值： Trigger - 告警触发 Recovery - 告警恢复 All - 告警触发和告警恢复<br>示例值： All |
| NoticeReceivers | 是 | 是 | Array of NoticeReceiver | 告警通知模板接收者信息。<br>示例值：查看 |
| WebCallbacks | 是 | 是 | Array of WebCallback | 告警通知模板回调信息。<br>示例值：查看 |
| AlarmNoticeId | 是 | 是 | String | 告警通知模板ID。<br>示例值： notice-dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| CreateTime | 是 | 是 | String | 创建时间。<br>示例值： 2022-11-07 16:30:53 |
| UpdateTime | 是 | 是 | String | 最近更新时间。<br>示例值： 2022-11-07 16:30:53 |
| Tags | 是 | 是 | Array of Tag | 告警通知模板绑定的标签信息。<br>示例值：查看 |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| JumpDomain | 否 | 是 | String | 调用链接域名。http:// 或者 https:// 开头，不能/结尾<br>示例值： https://console.cloud.demo.com |
| NoticeRules | 否 | 否 | Array of NoticeRule | 通知规则。<br>示例值：查看 |

## LogItems

LogItem的数组

被如下接口引用：GetAlarmLog、GetConfigurationTemplateApplyLog、SearchLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Data | 是 | 否 | Array of LogItem | 分析结果返回的KV数据对<br>示例值：查看 |

## ClbTopicExtendConfig

clb定制化的业务配置

被如下接口引用：CreateTopicExtendConfig、DescribeTopicExtendConfig、ModifyTopicExtendConfig

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Vip | 否 | 是 | String | LB关键信息，VIP<br>示例值： 127.0.0.1 |
| VpcId | 否 | 是 | Int64 | LB关键信息，VpcId<br>示例值： 23232323 |
| TopicId | 否 | 是 | String | clb服务端的公共topic<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| UserTopicId | 否 | 是 | String | clb用户的topic<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| UserUin | 否 | 是 | Uint64 | clb用户的uin信息<br>示例值： 13232323 |
| UserAppId | 否 | 是 | Uint64 | clb用户的appid信息<br>示例值： 2323232 |
| UserTmpSecretId | 否 | 是 | String | 临时证书加密密钥ID。最长不超过1024字节。<br>示例值： secretId |
| UserTmpSecretKey | 否 | 是 | String | 临时证书加密密钥Key。最长不超过1024字节。<br>示例值： secretKey |
| UserToken | 否 | 是 | String | token, 最长不超过4096字节。<br>示例值： userToken |
| TmpKeyExpired | 否 | 是 | Uint64 | 临时证书有效的时间，返回 Unix 时间戳，精确到秒<br>示例值： 1547696355 |
| LbKey | 否 | 是 | String | 唯一标识clb的一种业务<br>示例值： clb_topic |
| LogSample | 否 | 是 | String | 公共topic的采样比<br>示例值： [{"key":"key1","rule":"!=200","rate":10},{"key":"DEFAULT","r |
| UserSample | 否 | 是 | String | 用户topic的采样比<br>示例值： [{"key":"key1","rule":"!=200","rate":10},{"key":"DEFAULT","r |
| UserHealthTopicId | 否 | 是 | String | LB健康检查日志 Topic ID, 和topicId属于另外一种公共的topic<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| UserSampleStatus | 否 | 是 | Bool | topic的采集配置是否生效,true为生效,false为失效<br>示例值： false |
| UserTopicStatus | 否 | 是 | Uint64 | 1代表用户topic已删除，0代表用户topic未删除。<br>示例值： 0 |
| Collection | 否 | 是 | Bool | lbkey是否要采集到公共topic, true为要采集， false为不采集，默认为false<br>示例值： false |
| LbID | 否 | 是 | String | LB相关业务ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| ShowServerAddr | 否 | 是 | Bool | 是否展示ServerAddr字段，true 展示，false 不展示<br>示例值： false |

## ExtractRuleInfo

日志提取规则

被如下接口引用：CreateConfig、CreateConfigExtra、CreateCosRecharge、DescribeAgentConfigs、DescribeConfigExtras、DescribeConfigs、DescribeCosRecharges、DescribeMachineGroupConfigs、ModifyConfig、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TimeKey | 否 | 是 | String | 时间字段的key名字，TikeKey和TimeFormat必须成对出现<br>示例值： time |
| TimeFormat | 否 | 是 | String | 时间字段的格式，参考c语言的strftime函数对于时间的格式说明输出参数<br>示例值： YYDDWW |
| Delimiter | 否 | 是 | String | 分隔符类型日志的分隔符，只有LogType为delimiter_log时有效<br>示例值：， |
| LogRegex | 否 | 是 | String | 整条日志匹配规则，只有LogType为fullregex_log时有效<br>示例值： \w |
| BeginRegex | 否 | 是 | String | 行首匹配规则，只有LogType为multiline_log或fullregex_log时有效<br>示例值： \w |
| Keys | 否 | 是 | Array of String | 取的每个字段的key名字，为空的key代表丢弃这个字段，只有LogType为delimiter_log时有效，json_log的日志使用json本身的key。限制100个。<br>示例值： ["name","address"] |
| FilterKeyRegex | 否 | 是 | Array of KeyRegexInfo | 日志过滤规则列表（旧版），需要过滤日志的key，及其对应的regex。 注意：2.9.3及以上版本LogListener ，建议使用AdvanceFilterRules配置日志过滤规则。<br>示例值：查看 |
| UnMatchUpLoadSwitch | 否 | 是 | Bool | 解析失败日志是否上传，true表示上传，false表示不上传<br>示例值： true |
| UnMatchLogKey | 否 | 是 | String | 失败日志的key，当UnMatchUpLoadSwitch为true时必填<br>示例值： log_key |
| Backtracking | 否 | 是 | Int64 | 增量采集模式下的回溯数据量，默认：-1（全量采集）；其他非负数表示增量采集（从最新的位置，往前采集${Backtracking}字节（Byte）的日志）最大支持1073741824（1G）。 注意： COS导入不支持此字段。<br>示例值： -1 |
| IsGBK | 否 | 是 | Int64 | 是否为Gbk编码。 0：否；1：是。 注意 目前取0值时，表示UTF-8编码 COS导入不支持此字段。<br>示例值： 0 |
| JsonStandard | 否 | 是 | Int64 | 是否为标准json。 0：否； 1：是。 标准json指采集器使用业界标准开源解析器进行json解析，非标json指采集器使用CLS自研json解析器进行解析，两种解析器没有本质区别，建议客户使用标准json进行解析。<br>示例值： 0 |
| Protocol | 否 | 是 | String | syslog传输协议，取值为tcp或者udp，只有在LogType为service_syslog时生效，其余类型无需填写。 注意： 该字段适用于：创建采集规则配置、修改采集规则配置。 COS导入不支持此字段。<br>示例值： tcp |
| Address | 否 | 是 | String | syslog系统日志采集指定采集器监听的地址和端口 ，形式：[ip]:[port]，只有在LogType为service_syslog时生效，其余类型无需填写。 注意： 该字段适用于：创建采集规则配置、修改采集规则配置。 COS导入不支持此字段。<br>示例值： 127.0.0.1:9000 |
| ParseProtocol | 否 | 是 | String | rfc3164：指定系统日志采集使用RFC3164协议解析日志。 rfc5424：指定系统日志采集使用RFC5424协议解析日志。 auto：自动匹配rfc3164或者rfc5424其中一种协议。 只有在LogType为service_syslog时生效，其余类型无需填写。 注意： 该字段适用于：创建采集规则配置、修改采集规则配置 COS导入不支持此字段。<br>示例值： RFC3164 |
| MetadataType | 否 | 否 | Int64 | 元数据类型。0: 不使用元数据信息；1:使用机器组元数据；2:使用用户自定义元数据；3:使用采集配置路径。 注意： COS导入不支持此字段。<br>示例值： 0 |
| PathRegex | 否 | 是 | String | 采集配置路径正则表达式。 请用"()"标识路径中目标字段对应的正则表达式，解析时将"()"视为捕获组，并以__TAG__.{i}:{目标字段}的形式与日志一起上报，其中i为捕获组的序号。 若不希望以序号为键名，可以通过命名捕获组"(?<{键名}>{正则})"自定义键名，并以__TAG__.{键名}:{目标字段}的形式与日志一起上报。最多支持5个捕获组 注意： MetadataType为3时必填。 COS导入不支持此字段。<br>示例值： /(?.?)/(?.?)/access.log |
| MetaTags | 否 | 否 | Array of MetaTagInfo | 用户自定义元数据信息。 注意： MetadataType为2时必填。 COS导入不支持此字段。<br>示例值：查看 |
| EventLogRules | 否 | 否 | Array of EventLog | Windows事件日志采集规则，只有在LogType为windows_event_log时生效，其余类型无需填写。<br>示例值：查看 |

## ScheduledSqlResouceInfo

ScheduledSql的资源信息

被如下接口引用：CreateScheduledSql、DescribeScheduledSqlInfo、ModifyScheduledSql

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 目标主题id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| LogsetId | 否 | 否 | String | 日志集id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Uin | 否 | 否 | Int64 | 主账号Uin<br>示例值： 1736994580 |
| Region | 否 | 否 | String | 主题的地域信息<br>示例值： region |
| BizType | 否 | 否 | Int64 | 主题类型：0为日志主题，1为指标主题<br>示例值： 0 |
| MetricName | 否 | 否 | String | 指标名称<br>示例值： cpu_usage |

## ConditionTestResult

触发条件测试结果

被如下接口引用：CheckAlarmRule

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ErrorCode | 是 | 否 | Int64 | 错误码，可选值：1005代表触发条件表达式语法错误；1006代表触发条件计算错误，请检查参数<br>示例值： 1005 |
| ErrorMessage | 是 | 否 | String | 错误信息<br>示例值： SyntaxError |

## ContentInfo

投递日志的内容格式配置

被如下接口引用：CreateShipper、DescribeShipperPreview、DescribeShippers、ModifyShipper

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Format | 是 | 否 | String | 内容格式，支持json、csv<br>示例值： json |
| Csv | 否 | 是 | CsvInfo | csv格式内容描述<br>示例值：查看 |
| Json | 否 | 是 | JsonInfo | json格式内容描述<br>示例值：查看 |
| Parquet | 否 | 是 | ParquetInfo | parquet格式内容描述<br>示例值：查看 |

## MachineGroupInfo

机器组信息

被如下接口引用：DescribeConfigMachineGroups、DescribeMachineGroups

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： 559d015f-fac7-43cd-ad6e-3c4523fe8767 |
| GroupName | 是 | 否 | String | 机器组名称<br>示例值： group_name |
| MachineGroupType | 是 | 否 | MachineGroupTypeInfo | 机器组类型<br>示例值：查看 |
| CreateTime | 是 | 否 | String | 创建时间<br>示例值： 2021-03-05 00:40:22 |
| Tags | 是 | 是 | Array of Tag | 机器组绑定的标签列表<br>示例值：查看 |
| AutoUpdate | 是 | 是 | String | 是否开启机器组自动更新。true表示开启自动更新，false表示关闭自动更新。<br>示例值： "true" |
| UpdateStartTime | 是 | 是 | String | 升级开始时间，建议业务低峰期升级LogListener<br>示例值： "12:00:00" |
| UpdateEndTime | 是 | 是 | String | 升级结束时间，建议业务低峰期升级LogListener<br>示例值： "13:00:00" |
| ServiceLogging | 是 | 是 | Bool | 是否开启服务日志，用于记录因Loglistener 服务自身产生的log，开启后，会创建内部日志集cls_service_logging和日志主题loglistener_status,loglistener_alarm,loglistener_business，不产生计费。true表示开启服务日志，false表示关闭服务日志。<br>示例值： true |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Flag | 是 | 是 | String | TKE标志位，默认值为空字符串。空字符串表示日志不是来自于TKE，label_k8s表示日志来自于TKE。<br>示例值： label_k8s |
| DelayCleanupTime | 是 | 是 | Int64 | 机器组中机器离线定期清理时间<br>示例值： 30 |
| MetaTags | 是 | 否 | Array of MetaTagInfo | 机器组元数据信息列表<br>示例值：查看 |
| OSType | 是 | 否 | Uint64 | 操作系统类型，0: Linux，1: windows<br>示例值： 0 |

## CompressInfo

投递日志的压缩配置

被如下接口引用：CreateConfigurationTemplate、CreateShipper、DescribeConfigurationTemplates、DescribeShippers、ModifyConfigurationTemplate、ModifyShipper

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Format | 是 | 否 | String | 压缩格式，支持gzip、lzop、snappy和none不压缩<br>示例值： gzip |

## DataTransformTaskInfo

数据加工任务基本详情

被如下接口引用：DescribeDataTransformInfo

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Name | 是 | 否 | String | 数据加工任务名称<br>示例值： task_name |
| TaskId | 是 | 否 | String | 数据加工任务id<br>示例值： 96562452-58da-458c-9d2b-0b94e379c917 |
| EnableFlag | 是 | 否 | Int64 | 任务启用状态，默认为1，正常开启，2关闭<br>示例值： 1 |
| Type | 是 | 否 | Int64 | 加工任务类型，1： DSL， 2：SQL<br>示例值： 1 |
| SrcTopicId | 是 | 否 | String | 源日志主题<br>示例值： c6404313-dc28-402c-970d-9654afd08b1f |
| Status | 是 | 否 | Int64 | 当前加工任务状态（1准备中/2运行中/3停止中/4已停止）<br>示例值： 1 |
| CreateTime | 是 | 否 | String | 加工任务创建时间<br>示例值： 2022-12-08 21:00:35 |
| UpdateTime | 是 | 否 | String | 最近修改时间<br>示例值： 2024-03-20 15:51:01 |
| LastEnableTime | 是 | 否 | String | 最后启用时间，如果需要重建集群，修改该时间<br>示例值： 2024-03-20 15:50:00 |
| SrcTopicName | 是 | 否 | String | 日志主题名称<br>示例值： topicname |
| LogsetId | 是 | 否 | String | 日志集id<br>示例值： logset-1 |
| DstResources | 是 | 否 | Array of DataTransformResouceInfo | 加工任务目的topic_id以及别名<br>示例值：查看 |
| EtlContent | 是 | 否 | String | 加工逻辑函数<br>示例值： fields_set(\"111\",\"111\") |
| BackupTopicID | 是 | 是 | String | 兜底topic_id<br>示例值： 7b40b0cb-034a-4852-9477-9386068cede6 |
| BackupGiveUpData | 是 | 是 | Bool | 超限之后是否丢弃日志数据<br>示例值： false |

## ConfigurationTemplateInfo

配置模板信息

被如下接口引用：DescribeConfigurationTemplates

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TemplateId | 是 | 否 | String | 配置模板id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Version | 是 | 是 | Uint64 | 版本号<br>示例值： 0 |
| Name | 是 | 是 | String | 配置模板名称<br>示例值： Template_1 |
| Describes | 是 | 是 | String | 配置模板描述<br>示例值： 模板配置1 |
| ShipperTemplateInfos | 是 | 是 | Array of ShipperTemplateInfo | 投递cos配置模板<br>示例值：查看 |
| CreateTime | 是 | 是 | Uint64 | 创建时间<br>示例值： 1719307168 |
| UpdateTime | 是 | 是 | Uint64 | 更新时间<br>示例值： 1719307168 |

## SearchLogTopics

多主题检索返回信息

被如下接口引用：DescribeLogHistogram、SearchLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Errors | 是 | 是 | Array of SearchLogErrors | 多日志主题检索对应的错误信息<br>示例值：查看 |
| Infos | 是 | 是 | Array of SearchLogInfos | 多日志主题检索各日志主题信息<br>示例值：查看 |

## ShipperTemplateInfo

投递cos模板配置信息

被如下接口引用：CreateConfigurationTemplate、DescribeConfigurationTemplates、ModifyConfigurationTemplate

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ShipperName | 是 | 否 | String | 投递规则的名字，允许字符0-9、a-z、A-Z、 _、-、中文字符。 必须以${TopicName}-${TaskCreateTime} 为前缀；${TopicName} 字符最多200个，如果超出截断topicname，-${TaskCreateTime}14个，用户可添加41个自定义字符，总长度255。受限于CLS产品规格255。<br>示例值： ShipperName |
| Prefix | 是 | 否 | String | 创建的投递规则投递目录的前缀,允许字符 a-z、A-Z、0-9、_、-、/，并且以非/开头。${TopicName} 字符最多200个，如果超出截断topicname，${TaskCreateTime}14个，年月日时分秒25个，如果用户使用了默认参数，那么用户可添加17个自定义字符，总长度255。 受限于COS产品规格255，COS允许中文路径。<br>示例值： var/log |
| Partition | 否 | 是 | String | 投递日志的分区规则，支持strftime的时间格式表示<br>示例值： %Y%m%d |
| Compress | 否 | 是 | CompressInfo | 投递日志的压缩配置<br>示例值：查看 |
| MaxSize | 否 | 是 | Uint64 | 投递的文件的最大值，单位 MB，默认256，范围 5-256<br>示例值： 256 |
| Interval | 否 | 是 | Uint64 | 投递的时间间隔，单位 秒，默认300，范围 300-900<br>示例值： 60 |
| Content | 否 | 是 | ConfigurationContentInfo | 投递日志的内容格式配置。投递格式：仅支持JSON。<br>示例值：查看 |
| FilterRules | 否 | 是 | Array of FilterRuleInfo | 投递日志的过滤规则，匹配的日志进行投递，各rule之间是and关系，最多5个，数组为空则表示不过滤而全部投递<br>示例值：查看 |
| StartTime | 否 | 是 | Uint64 | 投递数据范围的开始时间点，不能超出日志主题的生命周期起点。如果用户不填写，默认为用户新建投递任务的时间。<br>示例值： 1732106219 |
| EndTime | 否 | 是 | Uint64 | 投递数据范围的结束时间点，不能填写未来时间。如果用户不填写，默认为持续投递，即无限。<br>示例值： 1732106219 |

## ShipperTaskInfo

投递任务信息

被如下接口引用：DescribeShipperTasks

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TaskId | 是 | 否 | String | 投递任务ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| ShipperId | 是 | 否 | String | 投递信息ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| RangeStart | 是 | 否 | Int64 | 本批投递的日志的开始时间戳，毫秒<br>示例值： 978653222 |
| RangeEnd | 是 | 否 | Int64 | 本批投递的日志的结束时间戳， 毫秒<br>示例值： 978653222 |
| StartTime | 是 | 否 | Int64 | 本次投递任务的开始时间戳， 毫秒<br>示例值： 978653222 |
| EndTime | 是 | 否 | Int64 | 本次投递任务的结束时间戳， 毫秒<br>示例值： 978653222 |
| Status | 是 | 否 | String | 本次投递的结果，"success","running","failed"<br>示例值： success |
| Message | 是 | 否 | String | 结果的详细信息<br>示例值： success |

## MachineGroupTypeInfo

机器组类型描述

被如下接口引用：AddMachineGroupInfo、CreateMachineGroup、DeleteMachineGroupInfo、DescribeConfigMachineGroups、DescribeMachineGroups、ModifyMachineGroup

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Type | 是 | 否 | String | 机器组类型。支持 ip 和 label。 ip：表示该机器组Values中存的是采集机器的ip地址 label：表示该机器组Values中存储的是机器的标签<br>示例值： label |
| Values | 否 | 否 | Array of String | 机器描述列表。<br>示例值： [10.1.1.1,10.2.2.2] |

## AlarmRuleTestResult

告警策略检测结果

被如下接口引用：CheckAlarmRule

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Index | 是 | 否 | Int64 | 位序<br>示例值： 1 |
| ErrorCode | 是 | 否 | Int64 | 错误码。0表示请求正常，-1001表示异常请求<br>示例值： 0 |
| ErrorMessage | 是 | 否 | String | 错误信息<br>示例值： err_message |
| AlarmTarget | 是 | 是 | CheckAlarmTarget | 当前检测的元数据<br>示例值：查看 |

## AlertHistoryNotice

告警历史的通知对象结构

被如下接口引用：DescribeAlertRecordHistory

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Name | 是 | 否 | String | 通知渠道组名称<br>示例值： notice_name |
| AlarmNoticeId | 是 | 否 | String | 通知渠道组ID<br>示例值： alarm-57f5808c-4a55-11eb-b378-0242ac130002 |

## DashboardNoticeMode

仪表盘订阅通知方式

被如下接口引用：CreateDashboardSubscribe、DescribeDashboardSubscribes、ModifyDashboardSubscribe、SearchDashboardSubscribe

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ReceiverType | 是 | 否 | String | 仪表盘通知方式。 Uin：云用户 Group：云用户组 Email：自定义Email WeCom: 企业微信回调<br>示例值： Email |
| Values | 否 | 否 | Array of String | 通知方式对应的值。 当ReceiverType不是 Wecom 时，Values必填。<br>示例值： ["168053"] |
| ReceiverChannels | 否 | 是 | Array of String | 仪表盘通知渠道。 支持：["Email","Sms","WeChat","Phone"]。 当ReceiverType是 Email 或 Wecom 时，ReceiverChannels不能赋值。<br>示例值： ["Email","Sms","WeChat","Phone"] |
| Url | 否 | 是 | String | 回调Url。 当ReceiverType是 Wecom 时，Url必填。 当ReceiverType不是 Wecom 时，Url不能填写。<br>示例值： https://console.cloud.demo.com |

## AgentUpdateInfo

Agent自动升级任务信息

被如下接口引用：HeartBeat

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| NeedUpdate | 是 | 否 | Bool | 是否需要升级。true表示需要升级，false表示不需要升级。<br>示例值： true |
| UpdateType | 是 | 是 | Int64 | 升级类型:0-null,1-manual,2-auto,3-force<br>示例值： 0 |
| UpdateAction | 是 | 是 | Int64 | 升级动作:0-null,1-update,2-revert<br>示例值： 0 |
| RetryCount | 是 | 是 | Int64 | 重试次数,最大3次<br>示例值： 0 |
| TargetVersion | 是 | 是 | String | 目标版本<br>示例值： 2.6.5 |
| DownloadUrl | 是 | 是 | String | 安装包下载链接1<br>示例值： https://console.cloud.demo.com/loglistener.tar.gz |
| DownloadUrlSecond | 是 | 是 | String | 安装包下载链接2<br>示例值： https://console.cloud.demo.com/loglistener.tar.gz |
| FileMd5 | 是 | 是 | String | 安装包文件MD5值<br>示例值： f560bf80453db17538aa44e2d5b121a1 |

## Column

日志分析的列属性

被如下接口引用：GetAlarmLog、GetConfigurationTemplateApplyLog、SearchLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Name | 是 | 否 | String | 列的名字<br>示例值： _col0 |
| Type | 是 | 否 | String | 列的属性<br>示例值： json |

## FilterRuleInfo

投递日志的过滤规则

被如下接口引用：CreateConfigurationTemplate、CreateShipper、DescribeConfigurationTemplates、DescribeShippers、ModifyConfigurationTemplate、ModifyShipper

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 过滤规则Key<br>示例值： log_key |
| Regex | 是 | 否 | String | 过滤规则<br>示例值： /w |
| Value | 是 | 否 | String | 过滤规则Value<br>示例值： log_value |

## ResourcesInfo

地域指标信息

被如下接口引用：DescribeResources

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Region | 是 | 否 | String | 地域<br>示例值： region |
| Logsets | 是 | 否 | Uint64 | 日志集数<br>示例值： 1 |
| Topics | 是 | 否 | Uint64 | 日志主题数<br>示例值： 1 |
| Partitions | 是 | 否 | Uint64 | 分区数<br>示例值： 1 |
| Machines | 是 | 否 | Uint64 | 机器数<br>示例值： 1 |
| HeartMachines | 是 | 否 | Uint64 | 心跳正常机器数<br>示例值： 1 |
| MetricTopics | 是 | 是 | Uint64 | 指标主题数<br>示例值： 1 |

## DashboardTopicInfo

仪表盘关联的topic信息

被如下接口引用：DescribeDashboards

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 主题id<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Region | 是 | 否 | String | topic所在的地域<br>示例值： region |

## RuleInfo

索引规则，FullText、KeyValue、Tag参数必须输入一个有效参数

被如下接口引用：CreateIndex、DescribeIndex、DescribeIndexs、ModifyIndex

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| FullText | 否 | 是 | FullTextInfo | 全文索引配置, 为空时代表未开启全文索引<br>示例值：查看 |
| KeyValue | 否 | 是 | RuleKeyValueInfo | 键值索引配置，为空时代表未开启键值索引<br>示例值：查看 |
| Tag | 否 | 是 | RuleTagInfo | 元字段索引配置，为空时代表未开启元字段索引<br>示例值：查看 |
| DynamicIndex | 否 | 是 | DynamicIndex | 键值索引自动配置，为空时代表未开启该功能。 启用后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。<br>示例值：查看 |

## UploadLogException

日志上传异常

被如下接口引用：DescribeExceptionResources

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| EventId | 是 | 否 | Int64 | 异常事件ID 400：参数错误 401：鉴权失败/密钥过期 413：消息格式过大 429：流控-区分超频/超频<br>示例值： 429 |
| Message | 是 | 否 | String | 异常事件描述<br>示例值： err_message |
| LatestTime | 是 | 否 | Uint64 | 最近发生时间<br>示例值： 1671182885 |

## ServiceLogConfigInfo

服务日志topic信息

被如下接口引用：DescribeAgentConfigs

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| LogsetId | 是 | 否 | String | 服务日志的logset信息 |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| | | | | 示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| TopicId | 是 | 否 | String | 服务日志的Topic ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| TopicName | 是 | 否 | String | 服务日志的Topic Name<br>示例值： demoTopic |

## DynamicIndex

键值索引自动配置，启用后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。

被如下接口引用：CreateIndex、DescribeIndex、DescribeIndexs、ModifyIndex

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Status | 否 | 是 | Bool | 动态索引配置开关。true表示开启动态索引，false表示关闭动态索引。<br>示例值： false |

## MultiTopicSearchInformation

多日志主题检索相关信息

被如下接口引用：SearchLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 否 | 否 | String | 要检索分析的日志主题ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Context | 否 | 否 | String | 透传上次接口返回的Context值，可获取后续更多日志，总计最多可获取1万条原始日志，过期时间1小时<br>示例值： Y29udGV4dC04MjMzNWRkMi01YmMxLTQ4NGYtYjQ4MS04MDg0NzAwYjQ1NDUxNjcy |

## CheckAlarmTarget

检查告警的告警对象

被如下接口引用：CheckAlarmRule

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 否 | 是 | String | 日志主题ID。<br>示例值： dfa4ac24-caa1-4e30-a027-349cb29937c4 |
| Query | 否 | 是 | String | 查询语句。<br>示例值： code:200 |
| Number | 否 | 是 | Int64 | 告警对象序号；从1开始递增。<br>示例值： 1 |
| StartTimeOffset | 否 | 是 | Int64 | 查询范围起始时间相对于告警执行时间的偏移，单位为分钟，取值为非正，最大值为0，最小值为-1440。<br>示例值： -5 |
| EndTimeOffset | 否 | 是 | Int64 | 查询范围终止时间相对于告警执行时间的偏移，单位为分钟，取值为非正，须大于StartTimeOffset，最大值为0，最小值为-1440。<br>示例值： 0 |
| LogsetId | 否 | 是 | String | 日志集ID。<br>示例值： dfa4ac24-caa1-4e30-a027-349cb29937c4 |
| SyntaxRule | 否 | 是 | Uint64 | 检索语法规则，默认值为0。 0：Lucene语法，1：CQL语法。 详细说明参见检索条件语法规则<br>示例值： 1 |
| Index | 否 | 是 | Int64 | 序号<br>示例值： 0 |

## JsonInfo

JSON类型描述

被如下接口引用：CreateConfigurationTemplate、CreateShipper、DescribeConfigurationTemplates、DescribeShipperPreview、DescribeShippers、ModifyConfigurationTemplate、ModifyShipper

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| EnableTag | 是 | 否 | Bool | 启用标志。true表示启用TAG标志，false表示关闭TAG标志。<br>示例值： true |
| MetaFields | 是 | 是 | Array of String | 元数据信息列表<br>示例值： [SOURCE、FILENAME、TIMESTAMP、HOSTNAME] |
| JsonType | 否 | 是 | Int64 | 投递Json格式，0：字符串方式投递；1:以结构化方式投递<br>示例值： 1 |

## HostFileInfo

自建k8s-节点文件配置信息

被如下接口引用：CreateConfigExtra、DescribeConfigExtras、ModifyConfigExtra

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| LogPath | 是 | 否 | String | 日志文件夹<br>示例值： /var/log |
| FilePattern | 是 | 否 | String | 日志文件名<br>示例值： file.log |
| CustomLabels | 否 | 是 | Array of String | metadata信息<br>示例值： ["key1=value1","key2=value2"] |

## ParquetInfo

Parquet内容

被如下接口引用：CreateShipper、DescribeShipperPreview、DescribeShippers、ModifyShipper

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ParquetKeyInfo | 是 | 否 | Array of ParquetKeyInfo | ParquetKeyInfo数组<br>示例值：查看 |

## LogItem

日志中的KV对

被如下接口引用：GetAlarmLog、GetConfigurationTemplateApplyLog、SearchLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 日志Key<br>示例值： http_code |
| Value | 是 | 否 | String | 日志Value<br>示例值： 200 |

## MetaTagInfo

元数据信息

被如下接口引用：CreateConfig、CreateConfigExtra、CreateCosRecharge、CreateMachineGroup、DescribeAgentConfigs、DescribeAgentMachineGroupMetadata、DescribeConfigExtras、DescribeConfigMachineGroups、DescribeConfigs、DescribeCosRecharges、DescribeMachineGroupConfigs、DescribeMachineGroups、ModifyConfig、ModifyConfigExtra、ModifyMachineGroup

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 否 | 否 | String | 元数据key<br>示例值： buss_key |
| Value | 否 | 否 | String | 元数据value<br>示例值： buss_value |

## JsonLogInfo

json格式日志内容

被如下接口引用：DescribeLatestJsonLog、DescribeLatestUserLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Log | 是 | 是 | Array of KeyLogInfo | K-V形式日志信息<br>示例值：查看 |
| Tag | 是 | 是 | Array of KeyLogInfo | K-V形式标签信息<br>示例值：查看 |
| Time | 是 | 否 | Uint64 | 时间戳<br>示例值： 1732175558 |

## CosRechargeInfo

cos导入配置信息

被如下接口引用：DescribeCosRecharges

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Id | 是 | 是 | String | COS导入配置ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| TopicId | 是 | 是 | String | 日志主题ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| LogsetId | 是 | 是 | String | 日志集ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Name | 是 | 是 | String | cos导入任务名称<br>示例值： cos任务导入 |
| Bucket | 是 | 是 | String | cos存储桶<br>示例值： bucket1 |
| BucketRegion | 是 | 是 | String | cos存储桶地域<br>示例值： region |
| Prefix | 是 | 是 | String | cos存储桶前缀地址<br>示例值： /var/log |
| LogType | 是 | 是 | String | 采集的日志类型，json_log代表json格式日志，delimiter_log代表分隔符格式日志，minimalist_log代表单行全文； 默认为minimalist_log<br>示例值： minimalist_log |
| Status | 是 | 是 | Uint64 | 状态 status 0: 已创建, 1: 运行中, 2: 已停止, 3: 已完成, 4: 运行失败。<br>示例值： 1 |
| Enable | 是 | 是 | Uint64 | 是否启用: 0： 未启用 ， 1：启用<br>示例值： 1 |
| CreateTime | 是 | 是 | String | 创建时间<br>示例值： 2021-11-08 14:40:22 |
| UpdateTime | 是 | 是 | String | 更新时间<br>示例值： 2021-11-08 14:40:22 |
| Progress | 是 | 是 | Uint64 | 进度条百分值<br>示例值： 0 |
| Compress | 是 | 是 | String | 压缩格式，支持 "", "gzip", "lzop", "snappy"; 默认空<br>示例值： gzip |
| ExtractRuleInfo | 是 | 是 | ExtractRuleInfo | 见： ExtractRuleInfo 结构描述<br>示例值：查看 |

## LogInfo

日志结果信息

被如下接口引用：GetAlarmLog、GetConfigurationTemplateApplyLog、SearchLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Time | 是 | 否 | Int64 | 日志时间，单位ms<br>示例值： 1619162953000 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 4463e7b0-3ec8-41a1-ae48-5d24b22167c2 |
| TopicName | 是 | 否 | String | 日志主题名称<br>示例值： 业务A日志主题 |
| Source | 是 | 否 | String | 日志来源IP<br>示例值： 10.0.0.1 |
| FileName | 是 | 否 | String | 日志文件名称<br>示例值： /var/access.log |
| PkgId | 是 | 否 | String | 日志上报请求包的ID<br>示例值： 0 |
| PkgLogId | 是 | 否 | String | 请求包内日志的ID<br>示例值： 0 |
| Logs | 是 | 是 | Array of LogItem | 日志内容，由多个LogItem (KV结构 ) 组成<br>示例值：查看 |
| HighLights | 是 | 是 | Array of HighLightItem | 日志内容的高亮描述信息<br>示例值：查看 |
| LogJson | 是 | 是 | String | 日志内容的Json序列化字符串<br>示例值： {\"content\":\"log\"} |
| HostName | 是 | 是 | String | 日志来源主机名称<br>示例值： hostName-1 |
| RawLog | 是 | 是 | String | 原始日志(仅在日志创建索引异常时有值)<br>示例值： {\"content\":\"log\"} |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| IndexStatus | 是 | 是 | String | 日志创建索引异常原因(仅在日志创建索引异常时有值)<br>示例值： IndexAllFail:fields like 'a' and 'a.b' can not appear in the same topic |

## AgentKeyInfo

需要升级的机器信息

被如下接口引用：UpgradeAgentNormal

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| IP | 是 | 否 | String | agent IP地址<br>示例值： 127.0.0.1 |
| InstanceID | 是 | 否 | String | 实例ID<br>示例值： loglistener-1462d334-0fb6-5f36-28d8-209e1b17ea2d |

## MultiCondition

多触发条件。

被如下接口引用：DescribeAlarms

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Condition | 否 | 否 | String | 触发条件<br>示例值： [$1.QUERYCOUNT]> 0 |
| ConditionInteractiveConfig | 否 | 否 | String | 交互式触发配置信息。<br>示例值： {"Value":{"type":"CONDITION","level":0,"executeQueryIndex":0,"field":"QUERYCOUNT","comparator":">","value":0}} |
| AlarmLevel | 否 | 否 | Uint64 | 告警级别。0:警告(Warn); 1:提醒(Info); 2:紧急 (Critical)。 - 不填则默认为0。<br>示例值： 0 |

## AlertHistoryRecord

告警历史的记录的结构

被如下接口引用：DescribeAlertRecordHistory

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| RecordId | 是 | 否 | String | 告警历史ID<br>示例值： 7cfbe19f-99af-42ca-8f9f-b59d1f9c6157 |
| AlarmId | 是 | 否 | String | 告警策略ID<br>示例值： alarm-8211a509-fee5-43dd-9ec0-6a6e7bc92552 |
| AlarmName | 是 | 否 | String | 告警策略名称<br>示例值： 日志错误数超限 |
| TopicId | 是 | 否 | String | 监控对象ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| TopicName | 是 | 否 | String | 监控对象名称<br>示例值： Nginx请求日志 |
| Region | 是 | 否 | String | 监控对象所属地域<br>示例值： region |
| Trigger | 是 | 否 | String | 触发条件<br>示例值： $1.d > 300 |
| TriggerCount | 是 | 否 | Int64 | 持续周期，持续满足触发条件TriggerCount个周期后，再进行告警<br>示例值： 1 |
| AlarmPeriod | 是 | 否 | Int64 | 告警通知发送频率，单位为分钟<br>示例值： 15 |
| Notices | 是 | 否 | Array of AlertHistoryNotice | 通知渠道组<br>示例值：查看 |
| Duration | 是 | 否 | Int64 | 告警持续时间，单位为分钟<br>示例值： 60 |
| Status | 是 | 否 | Int64 | 告警状态，0代表未恢复，1代表已恢复，2代表已失效<br>示例值： 1 |
| CreateTime | 是 | 否 | Uint64 | 告警发生时间，毫秒级Unix时间戳<br>示例值： 1619359718000 |
| GroupTriggerCondition | 否 | 是 | Array of GroupTriggerConditionInfo | 告警分组触发时对应的分组信息<br>示例值：查看 |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| AlarmLevel | 否 | 是 | Uint64 | 告警级别，0代表警告(Warn)，1代表提醒(Info)，2代表紧急 (Critical)<br>示例值： 0 |
| MonitorObjectType | 否 | 是 | Uint64 | 监控对象类型。 0:执行语句共用监控对象; 1:每个执行语句单独选择监控对象。<br>示例值： 0 |

## ConfigException

采集配置异常

被如下接口引用：DescribeExceptionResources

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ConfigId | 是 | 否 | String | 采集配置Id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Status | 是 | 否 | Int64 | 状态： 1：非结构化提取模式<br>示例值： 1 |
| ConfigName | 是 | 否 | String | 采集配置名称<br>示例值： 名称 |

## LogConfigInfo

采集日志配置信息

被如下接口引用：DescribeAgentConfigs

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： f6c4fa6f-367a-4f14-8289-1ff6f77ed975 |
| Path | 是 | 否 | String | 采集日志路径列表<br>示例值： /var/log/message |
| LogType | 是 | 否 | String | 日志类型，可选值：minimalist_log代表单行全文；multiline_log代表多行全文；fullregex_log代表单行完全正则；json_log代表JSON日志；multiline_fullregex_log多行完全正则；user_define_log代表组合解析；delimiter_log代表分隔符；<br>示例值： minimalist_log |
| ExtractRule | 是 | 否 | ExtractRuleInfo | 提取规则<br>示例值：查看 |
| LogFormat | 是 | 是 | String | 日志格式化格式，可选值：default代表默认格式；nginx_log代表nginx格式日志；<br>示例值： default |
| ExcludePaths | 是 | 是 | Array of ExcludePathInfo | 黑名单path列表<br>示例值：查看 |
| UserDefineRule | 是 | 是 | String | 用户自定义解析字符串<br>示例值： {"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":" |
| ConfigId | 是 | 是 | String | 采集配置ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| GroupIds | 是 | 是 | Array of String | 使用了元数据的机器组ID列表<br>示例值： ["dd1e5d83-fee5-cd35-ac57-20823c21b8cd"] |
| LogsetId | 是 | 否 | String | 日志集ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |

## ConfigExtraInfo

特殊采集规则配置信息

被如下接口引用：DescribeConfigExtras

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ConfigExtraId | 是 | 否 | String | 采集规则扩展配置ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Name | 是 | 否 | String | 采集规则名称<br>示例值： config_name |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Type | 是 | 否 | String | 类型：container_stdout、container_file、host_file<br>示例值： container_stdout |
| HostFile | 是 | 是 | HostFileInfo | 节点文件配置信息<br>示例值：查看 |
| ContainerFile | 是 | 是 | ContainerFileInfo | 容器文件路径信息<br>示例值：查看 |
| ContainerStdout | 是 | 是 | ContainerStdoutInfo | 容器标准输出信息<br>示例值：查看 |
| LogFormat | 是 | 是 | String | 日志格式化方式<br>示例值： default |
| LogType | 是 | 是 | String | 采集的日志类型，json_log代表json格式日志，delimiter_log代表分隔符格式日志，minimalist_log代表极简日志，multiline_log代表多行日志，fullregex_log代表完整正则，默认为minimalist_log<br>示例值： minimalist_log |
| ExtractRule | 是 | 是 | ExtractRuleInfo | 提取规则，如果设置了ExtractRule，则必须设置LogType<br>示例值：查看 |
| ExcludePaths | 是 | 是 | Array of ExcludePathInfo | 采集黑名单路径列表<br>示例值：查看 |
| UpdateTime | 是 | 否 | String | 更新时间<br>示例值： 2020-02-16 17:01:50 |
| CreateTime | 是 | 否 | String | 创建时间<br>示例值： 2020-02-16 17:01:50 |
| UserDefineRule | 是 | 是 | String | 用户自定义解析字符串<br>示例值： {"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":" |
| GroupId | 是 | 否 | String | 机器组ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| ConfigFlag | 是 | 是 | String | 自建采集配置标<br>示例值： label_k8s |
| LogsetId | 是 | 是 | String | 日志集ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| LogsetName | 是 | 是 | String | 日志集name<br>示例值： logset_name |
| TopicName | 是 | 是 | String | 日志主题name<br>示例值： topic_name |
| CollectInfos | 是 | 是 | Array of CollectInfo | 采集相关配置信息。详情见 CollectInfo复杂类型配置。<br>示例值：查看 |
| AdvancedConfig | 是 | 是 | String | 高级采集配置。 Json字符串， Key/Value定义为如下： ClsAgentFileTimeout(超时属性), 取值范围: 大于等于0的整数， 0为不超时 ClsAgentMaxDepth(最大目录深度)，取值范围: 大于等于0的整数 ClsAgentParseFailMerge(合并解析失败日志)，取值范围: true或false 样例：{"ClsAgentFileTimeout":0,"ClsAgentMaxDepth":10,"ClsAgentParseFailMerge":true}<br>示例值： {"ClsAgentFileTimeout":0,"ClsAgentMaxDepth":10,"ClsAgentParseFailMerge":true} |

## TopicInfo

日志主题信息

被如下接口引用：DescribeTopics

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| LogsetId | 是 | 否 | String | 日志集ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| TopicName | 是 | 否 | String | 日志主题名称<br>示例值： 业务A |
| PartitionCount | 是 | 否 | Int64 | 主题分区个数<br>示例值： 1 |
| Index | 是 | 否 | Bool | 主题是否开启索引（主题类型需为日志主题）。true表示开启索引，false表示关闭索引。<br>示例值： true |
| AssumerUin | 是 | 是 | Uint64 | AssumerUin非空则表示创建该日志主题的服务方Uin<br>示例值： 1736994580 |
| AssumerName | 是 | 是 | String | 云产品标识，主题由其它云产品创建时，该字段会显示云产品名称，例如CDN、TKE<br>示例值： CLS_bus |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| CreateTime | 是 | 否 | String | 创建时间<br>示例值： 2024-10-10 16:17:03 |
| TopicModifyAcl | 是 | 是 | Int64 | 若AssumerUin非空，则表示除服务方外其余调用者修改日志主题的权限<br>示例值： 1 |
| TopicShowAcl | 是 | 是 | Int64 | 若AssumerUin非空，则表示除服务方外其余调用者展示日志主题的权限<br>示例值： 1 |
| Status | 是 | 否 | Bool | 主题是否开启采集，true：开启采集；false：关闭采集。 创建日志主题时默认开启，可通过SDK调用ModifyTopic修改此字段。 控制台目前不支持修改此参数。<br>示例值： true |
| Tags | 是 | 是 | Array of Tag | 日志主题绑定的标签信息<br>示例值：查看 |
| RoleName | 是 | 是 | String | RoleName非空则表示创建该日志主题的服务方使用的角色<br>示例值： CLB_QcsRole |
| AutoSplit | 是 | 是 | Bool | 该主题是否开启自动分裂<br>示例值： true |
| MaxSplitPartitions | 是 | 是 | Int64 | 若开启自动分裂的话，该主题能够允许的最大分区数<br>示例值： 50 |
| StorageType | 是 | 是 | String | 主题的存储类型 hot: 标准存储 cold: 低频存储<br>示例值： hot |
| Period | 是 | 是 | Int64 | 生命周期，单位天，可取值范围1~3600。取值为3640时代表永久保存<br>示例值： 30 |
| SubAssumerName | 是 | 是 | String | 云产品二级标识，日志主题由其它云产品创建时，该字段会显示云产品名称及其日志类型的二级分类，例如TKE-Audit、TKE-Event。部分云产品仅有云产品标识(AssumerName)，无该字段。<br>示例值： TKE-Audio |
| LogsetInfo | 是 | 是 | LogsetInfo | 主题对应的日志集信息<br>示例值：查看 |
| Describes | 是 | 是 | String | 日志主题描述<br>示例值： demo部门demo业务topic |
| SubUin | 是 | 是 | Uint64 | 子用户。<br>示例值： 27356 |
| UserSample | 是 | 是 | String | 用户自定义抽样配置<br>示例值： [{"key":"DEFAULT","rate":10,"rule":""},{"key":"status","rate":20,"rule":"400 |
| UserSampleStatus | 是 | 是 | Bool | 用户采样率状态。true表示开启用户采样率，false表示关闭用户采样率。<br>示例值： true |
| WhitelistInfo | 是 | 是 | TopicWhitelistInfo | 日志主题名名单信息。<br>示例值：查看 |
| HotPeriod | 是 | 是 | Uint64 | 开启日志沉降，标准存储的生命周期， hotPeriod < Period。 标准存储为 hotPeriod, 低频存储则为 Period-hotPeriod。（主题类型需为日志主题）HotPeriod=0为没有开启日志沉降。<br>示例值： 0 |
| KeyId | 是 | 是 | String | kms-cls服务秘钥id<br>示例值： ASF-AFAS-JGJISDJG-SDGF |
| BizType | 是 | 是 | Uint64 | 主题类型。 0: 日志主题 1: 指标主题<br>示例值： 0 |
| IsWebTracking | 是 | 是 | Bool | 免鉴权开关。 false：关闭； true：开启。 开启后将支持指定操作匿名访问该日志主题。详情请参见日志主题。<br>示例值： false |

## ValueInfo

需要开启键值索引的字段的索引描述信息

被如下接口引用：CreateIndex、DescribeIndex、DescribeIndexs、ModifyIndex

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Type | 是 | 否 | String | 字段类型，目前支持的类型有：long、text、double<br>示例值： long |
| Tokenizer | 否 | 否 | String | 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、\n\t\r及转义符\； long及double类型字段需为空； 注意：\n\t\r本身已被转义，直接使用双引号包裹即可作为入参，无需再次转义。使用API Explorer进行调试时请使用JSON参数输入方式，以避免\n\t\r被重复转义<br>示例值： .* |
| SqlFlag | 否 | 否 | Bool | 字段是否开启分析功能。true表示开启分析功能，false表示关闭分析功能。<br>示例值： true |
| ContainZH | 否 | 是 | Bool | 是否包含中文，long及double类型字段需为false。true表示包含中文，false表示不包含中文。<br>示例值： false |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Alias | 否 | 是 | String | 索引键值别名<br>示例值： alias_name |

## CosFieldsData

COS文件（csv格式）字段信息

被如下接口引用：DescribeExternalDataSourcePreview

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| RowInfo | 是 | 是 | Array of String | COS文件（csv格式）行字段信息<br>示例值： ["name","age"] |

## GroupTriggerConditionInfo

分组触发条件

被如下接口引用：DescribeAlertRecordHistory

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | 分组触发字段名称<br>示例值： $1.count |
| Value | 是 | 否 | String | 分组触发字段值<br>示例值： 100 |

## FunctionArgument

函数参数描述

被如下接口引用：DescribeFunctions

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ArgIndex | 是 | 否 | Int64 | 参数序号，根据参数顺序定义<br>示例值： 1 |
| ArgName | 是 | 否 | String | 参数名称<br>示例值： field_name |
| ArgDesc | 是 | 否 | String | 参数描述<br>示例值： 字段名 |
| ArgType | 是 | 否 | String | 可接受的参数类型列表，包括字面常量、数组、条件表达式、函数表达式等任意一种或多种<br>示例值： string |
| ArgValueDefault | 是 | 否 | String | 参数默认值<br>示例值： default_value |
| ArgValueType | 是 | 否 | String | 范围、枚举类型<br>示例值： type1 |
| ArgValueScope | 是 | 否 | Array of String | 参数值域校验范围，这里仅针对常量进行校验，如果arg_value_type是scope类型，则此数组表示前闭后开区间，否则表示枚举的值类型. 如果此值为空，或者空数组，则不进行参数值校验<br>示例值： ["scope"] |
| IsNecessary | 是 | 否 | Bool | 是否必须。true表示必须，false代表不必须。<br>示例值： true |

## RuleKeyValueInfo

键值索引配置

被如下接口引用：CreateIndex、DescribeIndex、DescribeIndexs、ModifyIndex

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| CaseSensitive | 是 | 否 | Bool | 是否大小写敏感。true表示大小写敏感，false表示大小写不敏感。<br>示例值： false |
| KeyValues | 否 | 否 | Array of KeyValueInfo | 需要建立索引的键值对信息；最大只能配置100个键值对<br>示例值：查看 |
| TemplateType | 否 | 否 | String | 索引是否开启动态模板；若开启，则会根据上报的键值对配置索引，但是所有字段类型都是text，大小写敏感，不支持分析，采用默认分词符。已废弃<br>示例值： dynamic |

## KafkaProtocolInfo

Kafka访问协议

被如下接口引用：CheckRechargeKafkaServer、CreateKafkaRecharge、DescribeKafkaRecharges、ModifyKafkaRecharge、PreviewKafkaRecharge

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Protocol | 否 | 是 | String | 协议类型，支持的协议类型包括 plaintext、sasl_plaintext 或 sasl_ssl。建议使用 sasl_ssl，此协议会进行连接加密同时需要用户认证。 入参必填<br>示例值： sasl_plaintext |
| Mechanism | 否 | 是 | String | 加密类型，支持 PLAIN、SCRAM-SHA-256 或 SCRAM-SHA-512。 当Protocol为sasl_plaintext或sasl_ssl时必填<br>示例值： PLAIN |
| UserName | 否 | 是 | String | 用户名。 当Protocol为sasl_plaintext或sasl_ssl时必填<br>示例值： user_name |
| Password | 否 | 是 | String | 用户密码。 当Protocol为sasl_plaintext或sasl_ssl时必填<br>示例值： user_password |

## ConfigInfo

采集规则配置信息

被如下接口引用：DescribeConfigs、DescribeMachineGroupConfigs

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| ConfigId | 是 | 否 | String | 采集规则配置ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| Name | 是 | 是 | String | 采集规则配置名称<br>示例值： config_1 |
| LogFormat | 是 | 是 | String | 日志格式化方式<br>示例值： default |
| Path | 是 | 是 | String | 日志采集路径<br>示例值： /var/log/* |
| LogType | 是 | 是 | String | 采集的日志类型。 json_log代表：JSON-文件日志（详见使用 JSON 提取模式采集日志）； delimiter_log代表：分隔符-文件日志（详见使用分隔符提取模式采集日志）； minimalist_log代表：单行全文-文件日志（详见使用单行全文提取模式采集日志）； fullregex_log代表：单行完全正则-文件日志（详见使用单行-完全正则提取模式采集日志）； multiline_log代表：多行全文-文件日志（详见使用多行全文提取模式采集日志）； multiline_fullregex_log代表：多行完全正则-文件日志（详见使用多行-完全正则提取模式采集日志）； user_define_log代表：组合解析（适用于多格式嵌套的日志，详见使用组合解析提取模式采集日志）； service_syslog代表：syslog 采集（详见采集 Syslog）； windows_event_log代表：Windows事件日志（详见采集 Windows 事件日志）。<br>示例值： json_log |
| ExtractRule | 是 | 是 | ExtractRuleInfo | 提取规则，如果设置了ExtractRule，则必须设置LogType<br>示例值：查看 |
| ExcludePaths | 是 | 是 | Array of ExcludePathInfo | 采集黑名单路径列表<br>示例值：查看 |
| Output | 是 | 否 | String | 采集配置所属日志主题ID即TopicId<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| UpdateTime | 是 | 是 | Datetime | 更新时间<br>示例值： 2019-01-14 19:19:41 |
| CreateTime | 是 | 否 | Datetime | 创建时间<br>示例值： 2019-01-14 19:19:41 |
| UserDefineRule | 是 | 是 | String | 用户自定义解析字符串，详见使用组合解析提取模式采集日志。<br>示例值： "{"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":" |
| ConfigExtraId | 是 | 是 | String | config_extra主键ID<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| ConfigFlag | 是 | 是 | String | 采集配置标签<br>示例值： label_k8s |
| AdvancedConfig | 是 | 是 | String | 高级采集配置。 Json字符串， Key/Value定义为如下： ClsAgentFileTimeout(超时属性), 取值范围: 大于等于0的整数， 0为不超时 ClsAgentMaxDepth(最大目录深度)，取值范围: 大于等于0的整数 ClsAgentParseFailMerge(合并解析失败日志)，取值范围: true或false 样例： {"ClsAgentFileTimeout":0,"ClsAgentMaxDepth":10,"ClsAgentParseFailMerge":true} 控制台默认占位值：{"ClsAgentDefault":0}<br>示例值： "{"ClsAgentFileTimeout":0,"ClsAgentMaxDepth":10,"ClsAgentParseFailMerge":true}" |
| Source | 是 | 是 | Uint64 | 采集配置来源，0: 默认来源，1: TKE<br>示例值： 0 |

## SearchLogInfos

多日志主题检索topic信息

被如下接口引用：DescribeLogHistogram、SearchLog

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Period | 是 | 否 | Int64 | 日志存储生命周期<br>示例值： 30 |
| Border | 是 | 否 | Int64 | 冷热边界时间点（ms）<br>示例值： 1671415200000 |
| Context | 是 | 是 | String | 透传本次接口返回的Context值，可获取后续更多日志，过期时间1小时<br>示例值： Y29udGV4dC04MjMzNWRkMi01YmMxLTQ4NGYtYjQ4MS04MDg0NzAwYjQ1NDUxNjcy |

## DataTransformResouceInfo

数据加工的资源信息

被如下接口引用：CheckFunction、CreateDataTransform、DescribeDataTransformInfo、ModifyDataTransform

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 目标主题id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Alias | 是 | 否 | String | 别名<br>示例值： topicname |
| LogsetId | 否 | 否 | String | 日志集id<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Uin | 否 | 否 | String | 主账号Uin<br>示例值： 130000000212 |

## PartitionInfo

日志主题分区信息

被如下接口引用：DescribePartitions、MergePartition、SplitPartition

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| PartitionId | 是 | 否 | Int64 | 分区ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Status | 是 | 否 | String | 分区的状态（readwrite或者是readonly）<br>示例值： readonly |
| InclusiveBeginKey | 是 | 否 | String | 分区哈希键起始key<br>示例值： 00000000000000000000000000000000 |
| ExclusiveEndKey | 是 | 否 | String | 分区哈希键结束key<br>示例值： ffffffffffffffffffffffffffffffff |
| CreateTime | 是 | 否 | String | 分区创建时间<br>示例值： 2024-10-10 16:17:03 |
| LastWriteTime | 是 | 是 | String | 只读分区数据停止写入时间<br>示例值： 2024-10-10 16:17:03 |

## TopicIdAndRegion

仪表盘 topic与地域信息

被如下接口引用：DescribeDashboards

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 日志主题id<br>示例值： 57f5808c-4a55-11eb-b378-0242ac130002 |
| RegionId | 是 | 否 | Uint64 | 日志主题id所在的地域id。<br>示例值： 19 |

## FieldInfo

字段信息

被如下接口引用：CreateExternalDataSource、DescribeExternalDataSourcePreview、DescribeExternalDataSources、ModifyExternalDataSource

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| FieldName | 是 | 是 | String | 字段名<br>示例值： name |
| FieldType | 是 | 是 | String | 字段类型<br>示例值： string |

## NoticeRule

通知规则

被如下接口引用：CreateAlarmNotice、DescribeAlarmNotices

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Rule | 否 | 否 | String | 匹配规则 JSON串<br>示例值： {"Value":"AND","Type":"Operation","Children":[{"Type":"Condition","Value":"NotifyType","Children":[{"Value":"In","Type":"Compare"},{"Value":"[1,2]","Type":"Value"}]}]} |
| NoticeReceivers | 否 | 否 | Array of NoticeReceiver | 告警通知接收者信息。<br>示例值：查看 |
| WebCallbacks | 否 | 否 | Array of WebCallback | 告警通知模板回调信息，包括企业微信、钉钉、飞书。<br>示例值：查看 |

## TopicWhitelistInfo

日志主题白名单

被如下接口引用：DescribeTopics

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| CloudQueryLanguage | 是 | 是 | Bool | 当cql返回不为空时， true:可以使用CQL语法， false:不可以使用CQL语法。字段不存在时，不能使用CQL语法。<br>示例值： false |

## DashboardTemplateVariable

仪表盘订阅模板变量

被如下接口引用：CreateDashboardSubscribe、DescribeDashboardSubscribes、ModifyDashboardSubscribe、SearchDashboardSubscribe

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Key | 是 | 否 | String | key的值<br>示例值： TopicId |
| Values | 是 | 否 | Array of String | key对应的values取值values<br>示例值： ["dd1e5d83-fee5-cd35-ac57-20823c21b8cd"] |

## ConfigurationTemplateLogsetInfo

配置模板日志集相关信息

被如下接口引用：ApplyConfigurationTemplate

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| LogsetIds | 是 | 否 | Array of String | 日志集id列表<br>示例值： ["dd1e5d83-fee5-cd35-ac57-20823c21b8cd"] |
| MatchRange | 是 | 否 | Array of Uint64 | 动态配置范围。1:新增日志主题,2:已有日志主题。<br>示例值： [1,2] |
| Regex | 是 | 否 | String | 日志主题名称正则匹配表达式。<br>示例值： [a-z]{0,3} |

## CallBackInfo

回调配置

被如下接口引用：CreateAlarm、DescribeAlarms、ModifyAlarm

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Body | 是 | 否 | String | 回调时的Body。 可将各类告警变量放在请求内容中，详见帮助文档。 如下示例： {"data":"data"}<br>示例值： data |
| Headers | 否 | 是 | Array of String | 回调时的HTTP请求头部字段。 例如：下面请求头部字段来告知服务器请求主体的内容类型为JSON。 "Content-Type: application/json"<br>示例值： ["Content-Type: application/json"] |

## ExternalDataSourceSQLInfo

外部表SQL信息

被如下接口引用：CreateExternalDataSource、DescribeExternalDataSourcePreview、DescribeExternalDataSources、ModifyExternalDataSource

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| AccessMode | 否 | 否 | Uint64 | 访问方式。 1：私有链接(内网访问)；2：外网访问； 当DataSource为1时，AccessMode必填<br>示例值： 2 |
| Address | 否 | 是 | String | SQL访问IP地址。 当DataSource为1且AccessMode为2时，Address必填。<br>示例值： 10.2.3.4 |
| Port | 是 | 否 | String | SQL访问端口。 当DataSource为1时，Port必填<br>示例值： 3306 |
| EndPointServiceId | 否 | 是 | String | 私有链接终端节点服务ID。 当DataSource为1时，当AccessMode 为1时， EndPointServiceId字段必填。<br>示例值： vpcsvc-f8fth |
| User | 是 | 否 | String | SQL访问用户名<br>示例值： user_00 |
| Password | 否 | 是 | String | SQL访问密码<br>示例值： password |
| DatabaseName | 是 | 否 | String | SQL访问数据库名<br>示例值： databases |
| TableName | 是 | 否 | String | SQL访问表名<br>示例值： table |

## IndexException

索引配置异常

被如下接口引用：DescribeExceptionResources

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TopicId | 是 | 否 | String | 日志主题ID<br>示例值： dd1e5d83-fee5-cd35-ac57-20823c21b8cd |
| Status | 是 | 否 | Int64 | 状态： 1：未开启索引 2：未开启键值索引<br>示例值： 1 |
| TopicName | 是 | 否 | String | 日志主题名称<br>示例值： 主题名称 |

# 错误码

## 功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

```
{
  "Response": {
    "Error": {
      "Code": "AuthFailure.SignatureFailure",
      "Message": "The provided credentials could not be validated. Please check your signature is cor
rect."
    },
    "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
  }
}
```

Error 中的 Code 表示错误码，Message 表示该错误的具体信息。

## 错误码列表

### 公共错误码

| 错误码 | 说明 |
| --- | --- |
| AuthFailure.InvalidSecretId | 密钥非法（不是云 API 密钥类型）。 |
| AuthFailure.MFAFailure | MFA 错误。 |
| AuthFailure.SecretIdNotFound | 密钥不存在。 请在控制台检查密钥是否已被删除或者禁用，如状态正常，请检查密钥是否填写正确，注意前后不得有空格。 |
| AuthFailure.SignatureExpire | 签名过期。Timestamp 和服务器时间相差不得超过五分钟，请检查本地时间是否和标准时间同步。 |
| AuthFailure.SignatureFailure | 签名错误。 签名计算错误，请对照调用方式中的接口鉴权文档检查签名计算过程。 |
| AuthFailure.TokenFailure | token 错误。 |
| AuthFailure.UnauthorizedOperation | 请求未 CAM 授权。 |
| DryRunOperation | DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。 |

| 错误码 | 说明 |
| --- | --- |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidAction | 接口不存在。 |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| LimitExceeded | 超过配额限制。 |
| MissingParameter | 缺少参数错误。 |
| NoSuchVersion | 接口版本不存在。 |
| RequestLimitExceeded | 请求的次数超过了频率限制。 |
| ResourceInUse | 资源被占用。 |
| ResourceInsufficient | 资源不足。 |
| ResourceNotFound | 资源不存在。 |
| ResourceUnavailable | 资源不可用。 |
| UnauthorizedOperation | 未授权操作。 |
| UnknownParameter | 未知参数错误。 |
| UnsupportedOperation | 操作不支持。 |
| UnsupportedProtocol | http(s)请求协议错误，只支持 GET 和 POST 请求。 |
| UnsupportedRegion | 接口不支持所传地域。 |

## 业务错误码

| 错误码 | 说明 |
| --- | --- |
| InvalidParameter.InvalidEtlContent | 无效的数据加工语句。 |
| OperationDenied | 操作被拒绝。 |
| LimitExceeded.MachineGroupIp | 机器组IP超过限制。 |
| OperationDenied.NoticeHasAlarm | 通知模版已绑定告警，无法删除。 |

| 错误码 | 说明 |
| --- | --- |
| FailedOperation.ReadQpsLimit | 读qps超过限制。 |
| ResourceNotFound.LogsetNotExist | 指定的日志集不存在。 |
| MissingParameter | 缺少参数错误。 |
| LimitExceeded.LogSize | 日志大小超过限制。 |
| ResourceNotFound.ExportNotExist | 日志导出不存在。 |
| InvalidParameterValue | 参数取值错误。 |
| ResourceNotFound.AlarmNotExist | 告警策略不存在。 |
| ResourceNotFound.AlarmNoticeNotExist | 告警策略通知模板不存在。 |
| OperationDenied.TopicHasDeliverFunction | topic绑定了函数投递。 |
| ResourceNotFound.MachineGroupNotExist | 机器组不存在。 |
| InvalidParameter.AlarmNoticeConflict | 告警策略通知模板已经存在。 |
| ResourceNotFound.TopicNotExist | 日志主题不存在。 |
| FailedOperation.SyntaxError | 查询语句解析错误。 |
| InvalidParameter.AlarmConflict | 告警策略已经存在。 |
| InvalidParameter.ShipperConflict | 投递规则命名冲突。 |
| LimitExceeded.MachineGroup | 机器组超过限制。 |
| InvalidParameter.Content | 无效的Content。 |
| LimitExceeded.Partition | 分区超过限制。 |
| FailedOperation.PeriodModifyForbidden | 修改的生命周期被禁止。 |
| ResourceNotFound | 资源不存在。 |
| LimitExceeded.Tag | tag超过限制。 |
| FailedOperation | 操作失败。 |
| ResourceNotFound.ShipperTaskNotExist | 投递任务不存在。 |
| LimitExceeded.Config | 采集规则配置超过最大值限制。 |
| ResourceNotFound.ConfigNotExist | 指定的采集规则配置不存在。 |

| 错误码 | 说明 |
| --- | --- |
| OperationDenied.AccountNotExists | 账户不存在。 |
| OperationDenied.ACLFailed | ACL校验失败。 |
| InvalidParameter.DataFromTaskNotExist | 数据加工任务不存在。 |
| OperationDenied.AlarmNotSupportForSearchLow | 低频不支持告警。 |
| FailedOperation.MissingContent | 无效的Content。 |
| InvalidParameter.TopicConflict | 指定日志集下已经有同名的日志主题。 |
| FailedOperation.InvalidContext | 检索游标已失效或不存在。 |
| InvalidParameter.InValidIndexRuleForSearchLow | 低频不支持配置kv和tag索引。 |
| FailedOperation.BindedAlarm | 告警策略通知模板已经绑定到了某个告警策略上。 |
| InvalidParameter | 参数错误。 |
| ResourceNotFound.PartitionNotExist | 分区不存在。 |
| OperationDenied.AnalysisSwitchClose | 字段没有开启分析功能。 |
| FailedOperation.InvalidAlarm | 该告警策略状态异常，请检查下日志主题ID是否都存在。 |
| FailedOperation.GetlogReachLimit | 检索日志触发最大条数限制。 |
| LimitExceeded | 超过配额限制。 |
| OperationDenied.AccountIsolate | 账户欠费。 |
| LimitExceeded.Export | 日志导出数量超出限制。 |
| InvalidParameter.DashboardNameConflict | 仪表盘命名冲突。 |
| LimitExceeded.Topic | 日志主题数目超过限制。 |
| LimitExceeded.LogSearch | 并发查询超过限制，单topic并发最大值15。 |
| OperationDenied.TopicHasDataFormTask | topic绑定了数据加工。 |
| AuthFailure.UnauthorizedOperation | 请求未授权。 |
| FailedOperation.WriteTrafficLimit | 写流量超过限制。 |
| FailedOperation.TopicIsolated | 日志主题已隔离。 |

| 错误码 | 说明 |
| --- | --- |
| FailedOperation.InvalidPeriod | 离线存储保存时间不得少于7天。 |
| ResourceNotFound.IndexNotExist | 索引不存在。 |
| InternalError | 内部错误。 |
| InvalidParameter.MachineGroupConflict | 同名机器组已经存在。 |
| FailedOperation.WriteQpsLimit | 写qps超过限制。 |
| AuthFailure | CAM签名/鉴权错误。 |
| FailedOperation.LogsetConflict | 相同的日志集已存在。 |
| ResourceNotFound.ShipperNotExist | 投递规则不存在。 |
| LimitExceeded.MachineGroupLabels | 机器组Label超过限制。 |
| UnsupportedOperation | 操作不支持。 |
| InvalidParameter.IndexConflict | 指定日志主题已经存在索引规则。 |
| LimitExceeded.Logset | 日志集数量超出限制。 |
| FailedOperation.TopicClosed | 日志主题已关闭。 |
| InvalidParameter.LogsetConflict | 相同的日志集已存在。 |
| FailedOperation.SearchTimeout | 查询超时。 |
| FailedOperation.QueryError | 查询语句运行失败。 |
| InvalidParameter.DataFromTaskConflict | 数据加工任务冲突。 |
| InvalidParameter.ConfigConflict | 相同的采集配置规则已经存在。 |
| LimitExceeded.Shipper | 投递规则超出限制。 |
| ResourceNotFound.AgentVersionNotExist | agent version不存在。 |
| OperationDenied.AccountDestroy | 账户已销毁。 |
| FailedOperation.ShipperTaskNotToRetry | 投递任务不允许重试。 |
| FailedOperation.LogsetNotEmpty | 日志集下存在日志主题。 |