

云日志服务

用户指南

文档版本 01
发布日期 2025-09-17



版权所有 © 华为云计算技术有限公司 2025。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目录

- 1 授权 IAM 用户使用云日志服务 LTS..... 1
- 2 购买 LTS 资源包..... 3
- 3 日志管理..... 11
 - 3.1 日志管理概述..... 11
 - 3.2 管理日志组..... 13
 - 3.3 管理日志流..... 17
 - 3.4 查看日志管理..... 23
 - 3.5 设置多账号日志汇聚..... 29
- 4 日志接入..... 32
 - 4.1 日志接入概述..... 32
 - 4.2 使用 ICAgent 插件采集主机日志..... 36
 - 4.2.1 ICAgent 采集主机场景日志概述..... 36
 - 4.2.2 安装 ICAgent（区域内主机）..... 37
 - 4.2.3 安装 ICAgent（区域外主机）..... 43
 - 4.2.4 管理 ICAgent..... 53
 - 4.2.5 管理 LTS 主机组..... 61
 - 4.2.6 云主机 ECS 文本日志接入 LTS..... 69
 - 4.2.7 裸金属服务 BMS 文本日志接入 LTS..... 82
 - 4.2.8 ServiceStage 云主机日志接入 LTS..... 93
 - 4.3 使用 ICAgent 插件采集容器日志..... 102
 - 4.3.1 ICAgent 采集容器场景日志概述..... 102
 - 4.3.2 云容器引擎 CCE 应用日志接入 LTS..... 103
 - 4.3.3 ServiceStage 容器应用日志接入 LTS..... 117
 - 4.3.4 自建 K8s 应用日志接入 LTS..... 126
 - 4.4 配置 ICAgent 结构化解析..... 139
 - 4.5 使用云服务接入 LTS..... 165
 - 4.5.1 云服务接入 LTS 概述..... 165
 - 4.5.2 应用运维管理 AOM 接入 LTS..... 169
 - 4.5.3 API 网关 APIG 接入 LTS..... 169
 - 4.5.4 Astro 轻应用接入 LTS..... 170
 - 4.5.5 云堡垒机 CBH 接入 LTS..... 170
 - 4.5.6 内容分发网络 CDN 接入 LTS..... 171

4.5.7 云防火墙 CFW 接入 LTS.....	171
4.5.8 云审计服务 CTS 接入 LTS.....	171
4.5.9 分布式缓存服务 DCS 接入 LTS.....	172
4.5.10 文档数据库服务 DDS 接入 LTS.....	172
4.5.11 DDoS 防护 AAD 接入 LTS.....	172
4.5.12 分布式消息服务 Kafka 版接入 LTS.....	172
4.5.13 数据复制服务 DRS 接入 LTS.....	172
4.5.14 数据仓库服务 GaussDB(DWS)接入 LTS.....	173
4.5.15 弹性负载均衡 ELB 接入 LTS.....	173
4.5.16 企业路由器 ER 接入 LTS.....	174
4.5.17 函数工作流 FunctionGraph 接入 LTS.....	174
4.5.18 云数据库 GaussDB 接入 LTS.....	174
4.5.19 图引擎服务 GES 接入 LTS.....	174
4.5.20 云数据库 TaurusDB 接入 LTS.....	174
4.5.21 云数据库 GeminiDB 接入 LTS.....	174
4.5.22 云数据库 GeminiDB Mongo 接入 LTS.....	174
4.5.23 云数据库 GeminiDB Cassandra 接入 LTS.....	174
4.5.24 华为 HiLens 接入 LTS.....	175
4.5.25 设备接入 IoTDA 接入 LTS.....	175
4.5.26 AI 开发平台 ModelArts 接入 LTS.....	175
4.5.27 MapReduce 服务 MRS 接入 LTS.....	175
4.5.28 云数据库 RDS for MySQL 接入 LTS.....	175
4.5.29 云数据库 RDS for PostgreSQL 接入 LTS.....	175
4.5.30 云数据库 RDS for SQLServer 接入 LTS.....	175
4.5.31 应用与数据集成平台 ROMA Connect 接入 LTS.....	175
4.5.32 视频直播 Live 接入 LTS.....	175
4.5.33 消息通知服务 SMN 接入 LTS.....	176
4.5.34 安全云脑 SecMaster 接入 LTS.....	176
4.5.35 对象存储服务 OBS 接入 LTS（邀测）.....	176
4.5.36 虚拟私有云 VPC 接入 LTS.....	180
4.5.37 Web 应用防火墙 WAF 接入 LTS.....	181
4.6 使用 API 接入 LTS.....	181
4.6.1 API 接入概述.....	181
4.6.2 上报日志接口参考.....	184
4.6.3 上报高精度日志接口参考.....	187
4.7 使用 SDK 接入 LTS.....	191
4.7.1 云日志服务 Java SDK.....	191
4.7.2 云日志服务 Log4j2 SDK.....	208
4.7.3 云日志服务 LogBack SDK.....	214
4.7.4 云日志服务 Go SDK.....	221
4.7.5 云日志服务 C++ SDK.....	227
4.7.6 云日志服务 Python SDK.....	231

4.7.7 云日志服务 Net SDK.....	234
4.7.8 云日志服务鸿蒙 SDK.....	239
4.7.9 云日志服务 Web SDK.....	243
4.7.10 云日志服务 iOS SDK.....	248
4.7.11 云日志服务 Android SDK.....	254
4.7.12 云日志服务百度小程序 SDK.....	261
4.7.13 云日志服务微信小程序 SDK.....	265
4.7.14 云日志服务钉钉小程序 SDK.....	270
4.7.15 云日志服务支付宝小程序 SDK.....	275
4.7.16 云日志服务快应用 SDK.....	280
4.7.17 LTS SDK 接入错误码.....	285
4.8 跨 IAM 账号接入 LTS.....	287
4.9 使用 KAFKA 协议上报日志到 LTS.....	290
4.10 使用 Flume 采集器上报日志到 LTS.....	295
4.11 使用匿名写入采集日志.....	305
4.12 自建中间件（邀测）.....	311
4.12.1 中间件-MySQL 日志.....	311
4.12.2 中间件-Redis 日志.....	314
4.12.3 中间件-Kafka 日志.....	316
4.12.4 中间件-Nginx 日志.....	317
4.12.5 中间件-ElasticSearch 日志.....	320
4.12.6 中间件-Zookeeper 日志.....	323
4.12.7 中间件-IIS 日志.....	324
4.12.8 服务器-DNS 日志.....	326
4.12.9 中间件-Flink 日志.....	327
5 日志搜索与分析（管道符方式）.....	330
5.1 日志搜索与分析概述.....	330
5.2 设置云端结构化解析日志.....	331
5.2.1 设置日志云端结构化解析.....	331
5.2.2 设置云端结构化字段和 tag 字段.....	340
5.2.3 设置云端结构化自定义日志时间.....	342
5.2.4 设置云端结构化模板.....	344
5.3 创建 LTS 日志索引.....	348
5.4 搜索与分析日志.....	361
5.5 搜索语法介绍.....	370
5.6 创建快速分析.....	375
5.7 保存搜索与分析语句.....	381
5.8 查看上下文.....	383
5.9 查看实时日志.....	384
5.10 SQL 函数.....	385
5.10.1 聚合函数.....	385
5.10.2 字符串函数.....	394

5.10.3 日期和时间函数.....	409
5.10.4 正则式函数.....	432
5.10.5 同比和环比函数.....	435
5.10.6 窗口函数.....	438
5.10.7 类型转换函数.....	439
5.10.8 数学计算函数.....	440
5.10.9 URL 函数.....	441
5.10.10 IP 地址函数.....	447
5.10.11 电话号码函数.....	456
5.10.12 数组函数.....	458
6 日志搜索与分析（即将下线 SQL 分析方式）.....	468
6.1 日志搜索与分析概述.....	468
6.2 设置云端结构化解析日志.....	469
6.2.1 设置日志云端结构化解析.....	470
6.2.2 设置云端结构化字段和 tag 字段.....	478
6.2.3 设置云端结构化自定义日志时间.....	480
6.2.4 设置云端结构化模板.....	482
6.3 创建 LTS 日志索引.....	486
6.4 搜索日志.....	500
6.5 搜索语法介绍.....	509
6.6 分析日志.....	518
6.7 创建快速分析.....	520
6.8 保存搜索与分析语句.....	522
6.9 查看上下文.....	524
6.10 查看实时日志.....	525
6.11 SQL 分析语法介绍.....	526
6.11.1 SQL 查询语法概述.....	526
6.11.2 SQL 聚合函数.....	530
6.11.3 SQL 同比和环比函数.....	531
6.11.4 SQL JSON 函数.....	533
6.11.5 SQL IP 函数.....	534
6.11.6 SQL 数学函数.....	537
6.11.7 SQL 时间函数.....	540
6.11.8 SQL 最值函数.....	546
6.11.9 SQL 字符串函数.....	547
6.11.10 SQL SPLIT 函数.....	550
6.11.11 SQL 比较运算符.....	552
6.11.12 SQL IP 地址函数.....	553
6.11.13 SQL 归约函数.....	555
6.11.14 SQL 其他函数.....	555
6.11.15 SQL JOIN 语法.....	556
6.11.16 SQL 查询样例.....	557

7 日志可视化.....	560
7.1 日志可视化概述.....	560
7.2 使用统计图表将日志可视化.....	561
7.2.1 统计图表概述.....	561
7.2.2 LTS 表格.....	562
7.2.3 LTS 柱状图.....	563
7.2.4 LTS 折线图.....	565
7.2.5 LTS 饼图.....	566
7.2.6 LTS 数字图.....	569
7.2.7 LTS 数字折线图.....	570
7.2.8 LTS 地图.....	571
7.2.9 LTS 漏斗图.....	572
7.3 使用仪表盘将日志可视化.....	573
7.3.1 创建日志仪表盘.....	573
7.3.2 添加日志仪表盘过滤器.....	578
7.3.3 日志仪表盘模板（管道符方式）.....	580
7.3.3.1 APIG 仪表盘模板.....	580
7.3.3.2 CCE 仪表盘模板.....	585
7.3.3.3 CDN 仪表盘模板.....	594
7.3.3.4 CFW 仪表盘模板.....	598
7.3.3.5 CSE 仪表盘模板.....	601
7.3.3.6 CoreDNS 仪表盘模板.....	606
7.3.3.7 DCS 仪表盘模板.....	608
7.3.3.8 DDS 仪表盘模板.....	609
7.3.3.9 DMS 仪表盘模板.....	609
7.3.3.10 DSL 仪表盘模板.....	610
7.3.3.11 ELB 仪表盘模板（共享型）.....	611
7.3.3.12 ELB 仪表盘模板（独享型）.....	618
7.3.3.13 ER 仪表盘模板.....	625
7.3.3.14 LTS 仪表盘模板.....	626
7.3.3.15 METRIC 仪表盘模板.....	633
7.3.3.16 NGINX-INGRESS 仪表盘模板.....	634
7.3.3.17 NGINX 仪表盘模板.....	640
7.3.3.18 WAF 仪表盘模板.....	645
7.3.3.19 采集诊断仪表盘模板.....	650
7.3.3.20 VPC 仪表盘模板.....	653
7.3.4 日志仪表盘模板（SQL 分析方式）.....	655
7.3.4.1 APIG 仪表盘模板.....	655
7.3.4.2 CCE 仪表盘模板.....	660
7.3.4.3 CDN 仪表盘模板.....	670
7.3.4.4 CFW 仪表盘模板.....	674
7.3.4.5 CSE 仪表盘模板.....	677

7.3.4.6 CoreDNS 仪表盘模板.....	681
7.3.4.7 DCS 仪表盘模板.....	683
7.3.4.8 DDS 仪表盘模板.....	684
7.3.4.9 DMS 仪表盘模板.....	685
7.3.4.10 DSL 仪表盘模板.....	686
7.3.4.11 ELB 仪表盘模板（共享型）.....	687
7.3.4.12 ELB 仪表盘模板（独享型）.....	694
7.3.4.13 ER 仪表盘模板.....	701
7.3.4.14 LTS 仪表盘模板.....	702
7.3.4.15 METRIC 仪表盘模板.....	706
7.3.4.16 NGINX-INGRESS 仪表盘模板.....	708
7.3.4.17 NGINX 仪表盘模板.....	713
7.3.4.18 VPC 仪表盘模板.....	718
7.3.4.19 WAF 仪表盘模板.....	720
7.3.4.20 采集诊断仪表盘模板.....	725
8 日志告警.....	729
8.1 日志告警概述.....	729
8.2 配置日志告警规则.....	730
8.3 配置日志告警通知规则.....	751
8.3.1 在 LTS 页面创建消息模板.....	751
8.3.2 创建告警通知规则.....	757
8.4 查看 LTS 告警列表.....	760
9 日志转储.....	762
9.1 日志转储概述.....	762
9.2 日志转储至 OBS.....	763
9.3 日志转储至 DIS.....	773
9.4 日志转储至 DMS.....	776
9.5 日志转储至 DWS.....	780
9.6 日志转储至 DLI.....	783
10 日志消费与加工.....	786
10.1 DSL 加工（邀测）.....	786
10.1.1 DSL 加工概述.....	786
10.1.2 创建 DSL 加工任务.....	787
10.2 DSL 数据加工语法（邀测）.....	789
10.2.1 操作符函数.....	789
10.2.1.1 解析函数.....	789
10.2.1.2 资源函数.....	797
10.2.1.3 字典函数.....	801
10.2.1.4 列表函数.....	805
10.2.1.5 编码解码函数.....	809
10.2.1.6 IP 解析函数.....	823

10.2.1.7 特定结构化数据函数.....	832
10.2.1.8 正则表达式函数.....	836
10.2.1.9 日期时间函数.....	840
10.2.1.10 字符串函数.....	863
10.2.1.10.1 概述.....	863
10.2.1.10.2 多字符串和排序、反转、替换.....	865
10.2.1.10.3 常见操作.....	870
10.2.1.10.4 查找判断、切分和格式化.....	874
10.2.1.10.5 字符集判断.....	883
10.2.1.11 算术函数.....	889
10.2.1.12 转换函数.....	899
10.2.1.13 操作符函数.....	906
10.2.1.14 事件检查函数.....	926
10.2.2 全局操作函数.....	931
10.2.2.1 映射富化函数.....	931
10.2.2.2 字段值提取函数.....	941
10.2.2.3 字段操作函数.....	953
10.2.2.4 事件操作函数.....	960
10.2.2.5 流程控制函数.....	966
10.3 使用定时 SQL 进行日志加工.....	972
10.4 使用 FunctionGraph 服务提供的函数模板进行日志加工.....	976
10.5 日志生成指标（邀测）.....	976
10.6 使用消费组管理消费数据（邀测）.....	978
10.6.1 创建消费组.....	978
10.6.2 使用 Java SDK 管理消费组.....	980
10.6.3 使用 Go SDK 管理消费组.....	985
10.6.4 使用 Python SDK 管理消费组.....	989
10.6.5 Flink 消费.....	992
10.6.6 LTS-Flink-Connector 参数说明.....	996
11 LTS 配置中心管理.....	1001
11.1 设置 LTS 日志采集配额和使用量预警.....	1001
11.2 设置 LTS 日志内容分词.....	1003
11.3 设置 ICAgent 日志采集开关.....	1005
12 查看 LTS 审计事件.....	1007

1 授权 IAM 用户使用云日志服务 LTS

如果您需要对您所拥有的云日志服务LTS（Log Tank Service）进行精细的权限管理，可以使用[统一身份认证服务](#)（Identity and Access Management，简称IAM），通过IAM可以进行如下操作：

- 根据企业的业务组织，在您的华为云账号中，给企业中不同职能部门的员工创建IAM用户，让员工拥有唯一安全凭证，并使用LTS资源。
- 根据企业用户的职能，设置不同的访问权限，以达到用户之间的权限隔离。
- 将LTS资源委托给更专业、高效的其他华为云账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

如果华为云账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用LTS服务的其它功能。

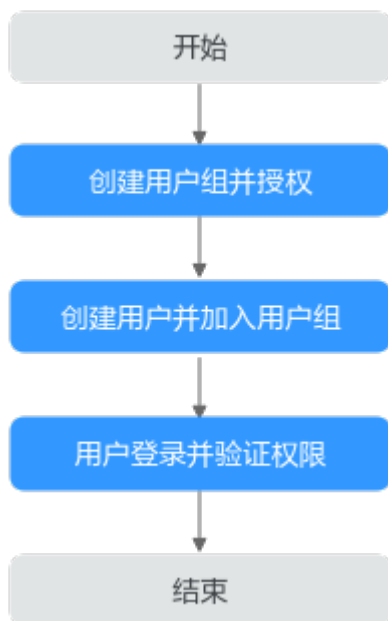
本章节为您介绍对用户授权的方法，操作流程如[图1-1](#)所示。

前提条件

给用户组授权之前，请您了解用户组可以添加的LTS权限，并结合实际需求进行选择LTS支持的系统权限，请参见：[权限管理](#)。

示例流程

图 1-1 给用户授权 LTS 权限流程



1. 登录统一身份认证服务IAM控制台。在IAM控制台创建用户组，并授予云日志服务操作权限“LTS FullAccess”。详细操作请参考[创建用户组并授权](#)。
2. 在IAM控制台创建用户，并将其加入[步骤1](#)中创建的用户组。详细操作请参考[创建用户并加入用户组](#)。
3. 使用新创建的用户登录控制台，切换至授权区域，进入云日志服务控制台可以正常操作并使用LTS。详细操作请参考[用户登录](#)并验证权限。

2 购买 LTS 资源包

云日志服务LTS支持购买资源包进行计费，在费用结算时，优先从资源包中抵扣用量。先购买资源包，后抵扣费用，适用于业务用量相对稳定的场景。更多计费信息[价格计算器](#)。

使用规则

- 资源包到期后，按需资源不会自动关闭，将会以按需付费的方式继续使用。
- 超出额度部分的按需用量，将会按需收费。
- 资源包计费周期的起点是用户设置的生效时间，清零和恢复时间点是购买时长到期后当天的23:59:59。
- 资源包到期后，剩余资源不会结余，将会自动清零。更多信息请参考[资源包扣减规则](#)。

购买 LTS 资源包（新版，白名单功能）

📖 说明

新版仅支持白名单用户使用，如有需要，请[提交工单](#)申请开通。

1. 登录[云日志服务控制台](#)。
2. 在日志管理页面，单击资源包卡片上的“购买资源包”，进入购买资源包页面。

图 2-1 资源包

资源包

[查看已购资源包](#)

资源包类型支持读写流量包、索引流量包、标准存储包，可以根据您的日志量及存储时长推荐...

购买资源包

3. 在“区域”下拉框选择对应区域，云日志服务所在的区域，建议选择与业务应用系统相同的地域。
4. 请根据业务需要选择计算方式，支持智能推荐和手动选择。

智能推荐

请参考[表2-1](#)设置参数。

图 2-2 智能推荐



表 2-1 智能推荐规格参数说明

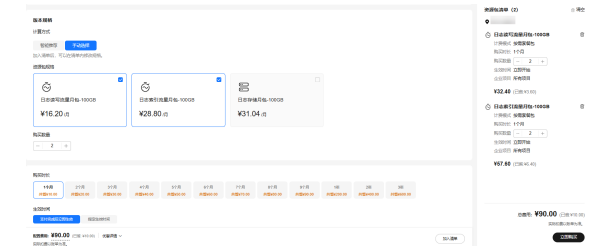
参数	说明
计算方式	计算方式为智能推荐。 基于日志资源统计中最近7天的用量，为您推荐合适的资源包类型和数量。（加入清单后可以修改资源包数量。） 资源包类型支持读写流量包、索引流量包、标准存储包、冷存储包。 说明 冷存储是白名单功能，如有需要请提交工单申请开通。功能开通后，请参考管理日志流开启冷存储。 购买成功后，可以在费用中心的资源包查看已购买资源包详情，详细操作请参考资源包。 ● 存储包：在LTS控制台的购买资源包页面，存储量的单位是GB/月。在费用中心控制台，存储量的单位是GB/小时。例如在LTS购买1个数量100GB的标准存储包（100GB/月），在费用中心的资源包页面存储总量有100GB*1个*24小时*30天=72000GB。 ● 读写流量包：若只购买1个数量100GB的读写流量包，在费用中心的资源包页面总量显示100GB。 ● 索引流量包：若只购买1个数量100GB的索引流量包，在费用中心的资源包页面总量显示100GB。
购买时长	选择资源包的使用时长，支持按月、按年购买。

参数	说明
生效时间	● 支付完成后立即生效：即购买成功后，优先从资源包中抵扣用量。 ● 指定生效时间：设置指定生效的具体时间。 资源包生效时间非整点时，例如2023/07/11 10:30:00开始生效，开始抵扣时间为下一个整点2023/07/11 11:00:00。
企业项目	● 所有企业项目均可使用。 ● 限定企业项目使用套餐包。选择特定企业项目，也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。 如果您没有开通企业管理服务，将无法看到企业项目选项。开通方法请参见如何开通企业项目。

1. 设置完成后，单击“加入清单”。

2. 在左侧的资源包清单下方，您可以根据业务需要调整购买数量，确认清单信息无误后，单击“立即购买”。
- 单次订单中资源包的购买数量总和不超过5000个。
- 截图费用仅供参考，请以实际账单为准。

图 2-3 购买资源包



3. 在配置详情页面，单击“去支付”。

4. 在购买云日志服务页面选择支付方式，单击“确认”，完成后，即可单击“返回云日志服务控制台”。
5. 在资源包卡片上方，单击“查看已购资源包”，进入资源包页面查看已购买资源包信息。详细操作请参考[资源包](#)。

手动选择

请参考[表2-2](#)设置参数。

图 2-4 手动选择

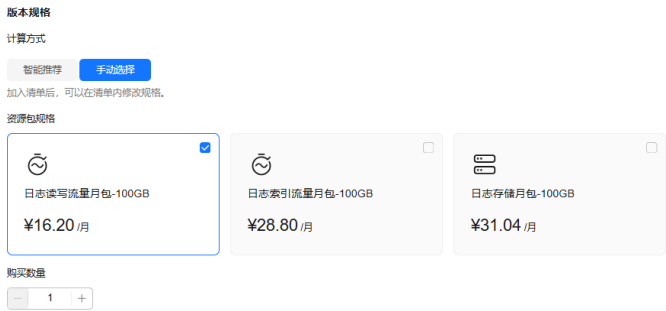
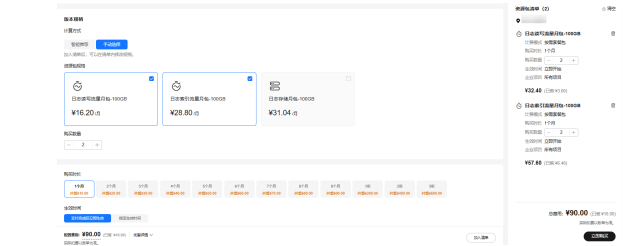


表 2-2 手动选择参数说明

参数	说明
计算方式	计算方式为 手动选择 。
资源包规格	自定义勾选资源包类型。资源包类型：日志读写流量、日志索引流量、标准存储。 ● 日志读写流量月包-100GB ● 日志索引流量月包-100GB ● 日志存储月包-100GB 购买成功后，可以在费用中心的资源包查看已购买资源包详情，详细操作请参考资源包。 ● 存储包：在LTS控制台的购买资源包页面，存储量的单位是GB/月。在费用中心控制台，存储量的单位是GB/小时。 例如在LTS购买1个数量100GB的标准存储包（100GB/月），在费用中心的资源包页面存储总量有100GB*1个*24小时*30天=72000GB。 ● 读写流量包：若只购买1个数量100GB的日志读写流量包，在费用中心的资源包页面总量显示100GB。 ● 索引流量包：若只购买1个数量100GB的日志索引流量包，在费用中心的资源包页面总量显示100GB。
购买数量	资源包的数量，支持设置1-3000。 加入清单后可以修改资源包数量。
购买时长	选择资源包的使用时长，支持按月、按年购买。
生效时间	● 支付完成后立即生效：即购买成功后，优先从资源包中抵扣用量。 ● 指定生效时间：设置指定生效的具体时间。 资源包生效时间非整点时，例如2023/07/11 10:30:00开始生效，开始抵扣时间为下一个整点2023/07/11 11:00:00。
企业项目	● 所有企业项目均可使用。 ● 限定企业项目使用套餐包。选择特定企业项目，也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。 如果您没有开通企业管理服务，将无法看到企业项目选项。开通方法请参见如何开通企业项目。

1. 设置完成后，单击“加入清单”。
2. 在左侧的资源包清单下方，您可以根据业务需要调整购买数量，确认清单信息无误后，单击“立即购买”。
 单次订单中资源包的购买数量总和不超过5000个。
 截图费用仅供参考，请以实际账单为准。

图 2-5 购买资源包



3. 在配置详情页面，单击“去支付”。
4. 在购买云日志服务页面选择支付方式，单击“确认”，完成后，即可单击“返回云日志服务控制台”。
5. 在资源包卡片上方，单击“查看已购资源包”，进入资源包页面查看已购买资源包信息。详细操作请参考[资源包](#)。

购买 LTS 资源包（旧版）

- 步骤1 登录[云日志服务控制台](#)。
- 步骤2 在日志管理页面，单击“购买资源包”，进入购买资源包页面。
- 步骤3 在“区域”下拉框选择对应区域，云日志服务所在的区域，建议选择与业务应用系统相同的地域。
- 步骤4 请根据业务需要选择计算方式，支持智能推荐、一键测算、手动选择，请参考[表2-3](#)、[表2-4](#)、[表2-5](#)设置参数。

说明

资源包中的冷存储包是白名单功能，同时需要开启智能冷存储，详情请参考[管理日志流](#)，如有需要请[提交工单](#)申请开通。

表 2-3 智能推荐规格参数说明

参数	说明
计算方式	智能推荐。 基于日志资源统计中最近7天的用量，为您推荐合适的资源包类型和数量。 资源包类型支持读写流量包、索引流量包、标准存储包、冷存储包（白名单功能）。 根据用户选择的数量进行计算，例如购买2个资源包，规格为规格*规格数量=100GB*2=200GB。 - 在LTS控制台的购买资源包页面，存储量的单位是GB/月。在费用中心控制台，存储量的单位是GB/小时。例如在LTS购买1个数量100GB的标准存储包（100GB/月），在费用中心的资源包页面存储总量有100GB*1个*24小时*30天=72000GB。 - 若只购买1个数量100GB的读写流量包，在费用中心的资源包页面总量显示100GB。 - 若只购买1个数量100GB的索引流量包，在费用中心的资源包页面总量显示100GB。

参数	说明
购买时长	选择资源包的使用时长，支持按月、按年购买。
生效时间	- 支付完成后立即生效，即购买成功后，优先从资源包中抵扣用量。 - 指定生效时间：设置指定生效的具体时间，例如2023/07/11 10:30:00。 资源包生效时间非整点时，例如2023/07/11 10:30:00开始生效，开始抵扣时间为下一个整点2023/07/11 11:00:00。
定向使用	- 所有企业项目均可使用。 - 限定企业项目使用套餐包，选择特定企业项目。 如果您没有开通企业管理服务，将无法看到企业项目选项。 开通方法请参见如何开通企业项目。

表 2-4 一键测算参数说明

参数	说明
计算方式	一键测算：按照每天新增日志量（GB/天或TB/天）、日志存储时长（天）、标准存储层数据保存时间(天)进行测算读写流量包、索引流量包、标准存储包、冷存储包的使用量。 （冷存储包是白名单功能，开通冷存储功能后，才需要设置标准存储层数据保存时间。） 读写流量包和索引流量包与每天新增日志量有关，标准存储包与每天新增日志量、日志存储时长、标准存储层数据保存时间(天)有关。 冷存储包与每天新增日志量、日志存储时长、标准存储层数据保存时间(天)有关。
测算结果	例如选择每天新增日志量10GB/天，日志存储时长30天，标准存储层数据保存时间7天，云日志服务提供默认规格是100GB，一键测算一个月的结果如下：（若是选择TB/天，先将TB换算为GB） - 读写流量包100GB*1，读写有5倍的压缩率，因此所需要的数量：$(10GB*30天)/5=60GB$，不足100GB的算1个。 - 索引流量包100GB*3，所需要的数量：$(10GB*30天)/100GB=3$个。 - 标准存储包100GB*1，所需要的数量：$10GB*7天=70GB$，不足100GB的算1个。 - 冷存储包100GB * 3，冷存储的天数为30-7=23天，所需要的数量：$(10GB*23天)/100GB=3$个，不足100GB的算1个。
购买时长	选择资源包的使用时长，支持按月、按年购买。

参数	说明
生效时间	- 支付完成后立即生效，即购买成功后，优先从资源包中抵扣用量。 - 指定生效时间：设置指定生效的具体时间，例如 2023/07/11 10:30:00。 资源包生效时间非整点时，例如2023/07/11 10:30:00开始生效，开始抵扣时间为下一个整点2023/07/11 11:00:00。
定向使用	- 所有企业项目均可使用。 - 限定企业项目使用套餐包，选择特定企业项目。 如果您没有开通企业管理服务，将无法看到企业项目选项。开通方法请参见如何开通企业项目。

表 2-5 手动选择参数说明

参数	说明
计算方式	手动选择。根据业务需要，自定义勾选资源包类型。
资源包类型	支持读写流量包、索引流量包、标准存储包、冷存储包（白名单功能）。
规格	默认支持100GB。
购买数量	资源包的数量，支持设置1~3000。
规格说明	根据用户选择的数量进行计算，例如购买2个资源包，规格为规格*规格数量=100GB*2=200GB。 - 在LTS控制台的购买资源包页面，存储量的单位是GB/月。 在费用中心控制台，存储量的单位是GB/小时。例如在LTS购买1个数量100GB的标准存储包（100GB/月），在费用中心的资源包页面存储总量有100GB*1个*24小时*30天=72000GB。 - 若只购买1个数量100GB的读写流量包，在费用中心的资源包页面总量显示100GB。 - 若只购买1个数量100GB的索引流量包，在费用中心的资源包页面总量显示100GB。
购买时长	选择资源包的使用时长，支持按月、按年购买。
生效时间	- 支付完成后立即生效，即购买成功后，优先从资源包中抵扣用量。 - 指定生效时间：设置指定生效的具体时间，例如 2023/07/11 10:30:00。 资源包生效时间非整点时，例如2023/07/11 10:30:00开始生效，开始抵扣时间为下一个整点2023/07/11 11:00:00。

参数	说明
定向使用	- 所有企业项目均可使用。 - 限定企业项目使用套餐包，选择特定企业项目。如果您没有开通企业管理服务，将无法看到企业项目选项。开通方法请参见如何开通企业项目。

步骤5 设置完成后，单击“加入清单”。

步骤6 确认清单信息无误后，单击“立即购买”。

步骤7 在购买详情页面，根据业务需要调整购买数量购买时长，单击“去支付”。

单次订单中资源包的购买数量总和不超过5000个。

步骤8 在购买云日志服务页面选择支付方式，单击“确认”，完成后，即可单击“返回云日志服务控制台”。

步骤9 在资源包卡片上方，单击“查看已购资源包”，进入资源包页面查看已购买资源包信息。详细操作请参考[资源包](#)。

----结束

3 日志管理

3.1 日志管理概述

云日志服务LTS以日志组和日志流为基本单位进行日志管理。使用云日志服务LTS之前，请先创建日志组和日志流。

日志组是云日志服务进行日志管理的基本单位，用于对日志流进行分组，一个日志组下面可以创建多个日志流。日志组本身不存储任何日志数据，仅方便您管理日志流。日志采集后，以日志流为单位，将不同类型的日志分类存储在不同的日志流上，方便对日志进一步分组管理。

了解更多日志管理的概念信息，请参考[基本概念](#)。

图 3-1 日志管理



- 根据业务需要在指定区域创建日志组，请参考[管理日志组](#)。
- 需要通过日志流存储日志数据，请参考[管理日志流](#)。
- LTS支持标准存储和冷存储，默认为标准存储，如果需要开启冷存储，请参考[智能冷存储](#)。

说明

冷存储是白名单功能，如有需要请[提交工单](#)申请开通。

- LTS支持多账号日志汇聚多个账号的日志组/日志流集中管理，请参考[设置多账号日志汇聚](#)。
- 在日志管理首页查看资源统计、我的收藏/我的收藏（本地缓存）、最近访问等信息，请参考[查看日志管理](#)。

应用场景

假设某互联网公司有两个部门：部门F-内容平台、部门H-企业服务，各部门的业务场景和日志分类参考如下，以下示例仅供参考，请以实际业务为准。

表 3-1 业务场景介绍

部门	日志说明
部门F-内容平台	• 业务场景：包含推荐系统（微服务）、内容管理系统（单体架构）、CDN加速服务。 • 日志类型： - 推荐系统（微服务）：用户行为日志（user_behavior_log）、模型训练日志（model_training_log）。 - 内容管理系统（单体架构）：后台操作日志（cms_operation_log）。 - CDN：访问日志（cdn_access_log）、边缘节点日志（edge_node_log）。
部门H-企业服务	• 业务场景：多租户软件系统（微服务+容器化），例如用户A、用户B等。 • 日志类型： - 租户通用日志：应用访问日志（app_access_log）、数据库慢查询日志（slow_query_log）。 - 特殊需求：用户A要求独立查看其租户日志。

表 3-2 日志分组规划

部门	日志组	日志流	标签1（标签键=标签值）	标签2（标签键=标签值）
部门F	F_content_platform_logset	recommend_user_behavior_log	部门=部门F	业务线=推荐系统
		recommend_model_training_log	部门=部门F	业务线=推荐系统
		cms_operation_log	部门=部门F	业务线=内容管理

部门	日志组	日志流	标签1（标签键=标签值）	标签2（标签键=标签值）
		cdn_access_log	部门=部门F	业务线=CDN
部门H	H_enterprise_services_logset	tenantA_app_access_log	部门=部门H	租户=用户A
		tenantB_app_access_log	部门=部门H	租户=用户B
	H_public_services_logset	slow_query_log	部门=部门H	类型=数据库

使用云日志服务LTS管理上述日志时，LTS支持为日志组、日志流添加标签，每个标签由“标签键”和“标签值”组成，用于快速识别、检索和管理目标对象。参考表3-2为不同部门创建日志组和日志流，并分别设置标签信息。

- 一个部门创建一个日志组，方便统一管理同一个部门的所有日志。在日志组下创建多个日志流，用于代表不同业务线的日志，如果后续新增其他的项目，再到对应日志组下面新建一个日志流即可。
部门H创建两个日志组是为了将用户日志和数据库日志区分，方便维护。
- 标签1可以用于区分不同的部门，标签2用于区分同部门不同业务线的日志，结合LTS权限管理可以控制每个部门的人员仅可查看所属部门的数据。因此，结合标签可以实现：
 - a. 多维度权限隔离：通过定义不同的标签，可以实现部门隔离、业务线隔离、租户隔离，满足不同租户对日志数据的管理要求。
 - b. 动态扩展与灵活调整：若用户B从部门H迁移至新成立的部门L，只需修改用户B对应日志流（tenantB_app_access_log）的标签为部门=部门L，日志数据无需物理迁移。
 - c. 快速搜索与分析：通过标签类型（例如数据库、计费等）可快速搜索其他部门的相关日志（如慢查询、对账日志、交易日志等），用于全公司数据库性能分析或统计各个部门的费用情况。

3.2 管理日志组

日志组（LogGroup）是云日志服务进行日志管理的基本单位，用于对日志流进行分组，一个日志组下面可以创建多个日志流。日志组本身不存储任何日志数据，仅方便您管理日志流，每个账号下可以创建100个日志组。

一个日志组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/服务的日志流归属到同一个日志组下。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组下的日志流即可，其它项目的日志流不会对其产生干扰。

LTS支持按照业务需求对不同的日志组添加对应的标签，方便运维人员管理业务。

前提条件

如果您使用华为账号创建的IAM用户进行操作，IAM用户需要具备足够的权限才能使用云日志服务。具体操作请参见[授权IAM用户使用云日志服务LTS](#)。

创建日志组

- 步骤1** 登录[云日志服务控制台](#)。
- 步骤2** 进入“日志管理”页面，单击“创建日志组”。
- 步骤3** 在“创建日志组”页面中，参考[表3-3](#)填写日志组相关信息。

表 3-3 日志组参数说明

参数	说明
日志组名称	日志组是云日志服务进行日志管理的基本单位，用于对日志流进行分类。如果日志较多，需要分门别类，建议您为日志组做好命名，方便后续快速查找日志。 LTS会自动生成一个默认的日志组名称，建议您根据业务自定义日志组的名称，创建成功后支持修改日志组名称。命名规范如下： • 日志组名称只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。长度为1~64个字符。 • 日志组名称不能重复。
企业项目	企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。 默认选择default，建议根据业务选择企业项目，也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。 • 企业项目需要开通后才能使用，请参考如何开通企业项目。 • 支持将该企业项目资源迁出，详细请参考迁出企业项目资源。
日志存储时间(天)	日志组的存储时间，即日志上报到LTS后日志存储的时间。 日志数据默认存储30天，可以在1~365天之间设置。 云日志服务LTS根据配置的日志存储时间定时清理日志内容，例如日志存储时间为30天，上报到LTS的日志只保存30天，30天后开始删除日志内容。 说明 目前白名单用户的日志存储时间支持1095天，如有需要，请提工单申请。详细操作请参考提交工单

参数	说明
标签	按照业务需求对不同的日志组添加对应的标签。单击“添加”，分别填写标签键key和标签值value，开启应用到日志流后，日志组下的所有日志流即可自动加上该标签。如需添加多个标签可重复该步骤，最多支持添加20个标签。 标签键key限制条件： - 标签键可以包含任意语种的字母、数字和空格，以及_!:=+~@字符，但首尾不能包含空格，且不能以_sys_开头。 - 标签键长度不能超过128个字符。 - 标签键名称不可重复。 标签值value限制条件： - 标签值可以包含任意语种的字母、数字和空格，以及_!:=+~@字符。 - 标签值长度不能超过255个字符。 标签策略： 若您的组织已经设定云日志服务的相关标签策略，则需按照标签策略规则为日志组、日志流、日志接入、主机组、告警规则添加标签。标签如果不符合标签策略的规则，则可能会导致日志组、日志流、日志接入、主机组、告警规则创建失败，请联系组织管理员了解标签策略详情。标签策略详细介绍请参考标签策略概述，标签管理详细介绍请参考标签管理。 删除标签： 如需删除标签，请单击标签操作列的“删除”。 警告 删除标签无法恢复，请谨慎操作。 如果配置转储时使用了该标签，删除标签后，请同步修改转储配置信息。
备注	自定义填写备注信息，字符长度0-1024个字符。

步骤4 单击“确定”，日志组创建成功，即可在日志组列表下方生成一条日志组信息。

图 3-2 日志组

日志组列表

创建日志组

日志组名称	备注	所属项目	日志流	标签	创建时间	操作
✓ idp-log-12345	idplog112	default	4		2025/07/29 11:51:56 GMT...	修改 删除 更多
✓ idp-log-goldlog	idplog	default	4		2025/07/21 17:43:46 GMT...	修改 删除 更多
✓ its-group-FR455		default	1		2025/06/24 21:50:51 GMT...	修改 删除 更多

- 在日志组列表中，可以查看日志组名称、标签、日志流数量等信息。
- 单击日志组名称，可跳转到日志详情页面。

----结束

修改日志组

日志组创建完成后，如果您需要修改日志组名称、日志存储时间、标签等信息，参考如下操作：

- 步骤1 在日志组列表中，单击待修改日志组操作列下的“修改”。
- 步骤2 在弹出的修改日志组页面中，修改日志组名称、日志存储时间、标签等信息。
- 步骤3 完成后，单击“确定”。
- 步骤4 修改成功后，鼠标悬浮在日志组名称上，显示修改后日志组名称、日志组原始名称、日志组ID。

----结束

删除日志组

如果日志组不再需要使用，可以删除日志组。日志组删除后，日志组中的日志流、日志数据将被同时删除。

如果日志组绑定了日志转储任务，删除日志组之前，需要先删除该日志组关联的日志转储任务。



警告

删除日志组会导致用户日志相关业务异常，日志组删除后无法恢复，请谨慎操作。

- 步骤1 在日志组列表中，单击待删除日志组操作列下的“删除”。
- 步骤2 在弹出框中输入“DELETE”或单击“一键输入”，单击“确定”，完成日志组删除。

图 3-3 删除日志组



----结束

搜索日志组/日志流

在日志组列表中，支持通过如下筛选条件进行搜索：

- 日志组/日志流：日志组名称或日志流名称。
- 日志组原始名称/日志流原始名称：日志组或日志流修改名称之前的原始名称，即第一次创建日志组或日志流时定义的名称。
- 日志组名称/ID：日志组名称或日志组ID（鼠标悬浮在日志组名称上即可查看日志组ID）。
- 日志流名称/ID：日志流名称或日志流ID（鼠标悬浮在日志流名称上即可查看日志流ID）。
- 企业项目：创建日志组选择的企业项目。
- 日志组标签：日志组设置的标签。
- 备注：创建日志组时自定义的备注信息。

其他操作

- **查看日志组详情**：在日志组列表中，单击目标日志组操作列的“更多 > 详情”，可查看该日志组名称、日志组ID、创建时间等详情。
- **ICAgent采集诊断**：需要先开启“ICAgent诊断开关”，请参考[设置ICAgent日志采集开关](#)。单击目标日志组操作列下的“更多 > ICAgent采集诊断”，可查看该日志组的ICAgent异常监控、ICAgent采集监控、ICAgent整体状态。更多信息请参考[采集诊断仪表盘模板](#)。
- **导出日志组及日志流信息**：单击搜索框旁边的，导出当前账号下的所有日志组及日志流信息到本地查看。
- **查看流量监控**：单击目标日志组操作列的“更多 > 流量监控”，可查看该日志组的资源用量明细和用量Top排序。（白名单功能，开通可观测For LTS后才能看到，详细请参考[设置LTS日志采集配额和使用量预警](#)。）
- 支持分别按日志组名称、日志流数量或创建时间对日志组进行升序或降序。

相关文档

LTS支持通过调用API接口实现创建日志组、修改日志组、删除日志组等，详细请参考[日志组管理](#)。

3.3 管理日志流

云日志服务以日志流（LogStream）作为日志管理维度。每个日志流只能属于一个日志组，每个日志组中可创建多个日志流。

日志采集后，以日志流为单位，将不同类型的日志分类存储在不同的日志流上，方便对日志进一步分类管理。如果日志较多，需要分门别类，建议您创建多个日志流，并给日志流做好命名，方便后续快速查找日志。例如，您可以将操作日志、访问日志等接入不同的日志流，查询日志时可以进入对应的日志流快速查看日志。支持按照业务需求对不同的日志流添加对应的标签，方便运维人员管理业务。

1个日志组中最多可以创建100个日志流，如果无法创建日志流，建议删除不再需要使用的日志流后重试，或者在新的日志组中创建日志流。

前提条件

已创建日志组。

创建日志流

- 步骤1 登录[云日志服务控制台](#)。
- 步骤2 单击日志组名称对应的 。
- 步骤3 单击“创建日志流”，在创建日志流页面，参考[表3-4](#)填写日志流相关信息。

表 3-4 日志流参数说明

参数	说明
日志组名称	默认显示所属日志组名称，不支持修改。
日志流类型	说明 白名单功能，如果需要使用，请提交工单申请开通。 根据业务需要选择日志流类型。 - 标准型包含LTS完整功能，适用于构建完整可观测性系统。 - 搜索型仅支持关键词搜索。不支持快速分析/统计图表/日志分析/SQL告警，适用于日志搜索场景，其优点是索引流量费用是标准型的31%。更多信息请参考计费项。
日志流名称	LTS会自动生成一个默认的日志流名称，建议您根据业务自定义日志流的名称，创建成功后支持修改日志流名称。命名规范如下： - 只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。 - 长度为1-64个字符。 - 日志流名称不能重复。
企业项目	选择业务需要的企业项目，默认为default。也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。 - 企业项目需要开通后才能使用，请参考如何开通企业项目。 - 支持将该企业项目资源迁出，详细请参考迁出企业项目资源。如果创建转储任务时选择多个日志流，则这些日志流无法进行企业项目迁移。
日志存储	若关闭日志存储，则无法开启日志存储时间。 - 开启“日志存储”：日志将会被存入搜索引擎，能使用日志全量功能。 - 关闭“日志存储”：日志不会存储到LTS，可节约索引流量和存储费用，只能使用日志生成指标、转储功能，不能使用日志搜索分析、告警、消费加工等其他功能。

参数	说明
日志存储时间 (天)	日志流的存储时间，即日志上报到LTS后日志存储的时间。 日志数据默认存储30天，可以在1~365天之间设置。 • 打开日志流的“日志存储时间”：日志的存储时间使用日志流设置的日志存储时间。 • 关闭日志流的“日志存储时间”：日志的存储时间使用日志组设置的日志存储时间。 • 超出存储时间的日志将会被自动删除，您可以按需将日志数据转储至OBS桶中进行长期存储。 说明 目前仅支持白名单用户提交工单申请开通日志存储时间为1095天，但是存量Region需要根据底层资源评估是否支持1095天。
智能冷存储	白名单功能，开启日志存储时间后，根据业务需要设置智能冷存储功能。详细请参考智能冷存储。 开启智能冷存储需要日志存储时间大于7天。
标准存储层数据 保存时间	开启智能冷存储后，需要设置标准存储层数据的保存时间。 标准存储数据至少需保存7天才能转换为智能冷存储层数据。当日志保存时间大于标准存储层数据保存时间，且小于日志存储时间时，日志数据将自动转换为智能冷存储层数据存储。
匿名写入	匿名写入默认关闭，适用于安卓/iOS/小程序/浏览器端上报日志才能使用匿名写入。 需要提交工单开通白名单。 打开匿名写入则表示该日志流打开匿名写入权限，不会经过有效鉴权，可能产生脏数据。关于SDK使用请参考使用SDK接入LTS。 说明 使用匿名写入采集日志功能仅支持华北-北京四、华东-上海一、华南-广州区域的白名单用户使用，如有需要，请提交工单，其他区域暂不支持申请开通。
分区数量	分区数量仅对消费日志有效。每个分区支持5MB/S写流量，10MB/S读流量；当读写流量超过分区的读写能力时，LTS会自动扩容读写分区数量，您也可以手动扩容读写分区数量，来提供更高的读写能力。输入值范围1-64。 单击“查看当前分区详情”进入分区详情页面，查看分区、状态、创建时间。 说明 白名单功能，如果需要使用，请提交工单申请开通。

参数	说明
标签	按照业务需求对不同的日志流添加对应的标签，单击“添加”，分别填写标签键key和标签值value。如需添加多个标签可重复该步骤，最多支持添加20个标签。 标签键key限制条件： - 标签键可以包含任意语种的字母、数字和空格，以及_!:=+-@字符，但首尾不能包含空格，且不能以_sys_开头。 - 标签键长度不能超过128个字符。 - 标签键名称不可重复。 标签值value限制条件： - 标签值可以包含任意语种的字母、数字和空格，以及_!:=+-@字符。 - 标签值长度不能超过255个字符。 标签策略： 若您的组织已经设定云日志服务的相关标签策略，则需按照标签策略规则为日志组、日志流、日志接入、主机组添加标签。标签如果不符合标签策略的规则，则可能会导致日志组、日志流、日志接入、主机组创建失败，请联系组织管理员了解标签策略详情。标签策略详细介绍请参考标签策略概述，标签管理详细介绍请参考标签管理。 删除标签： 如需删除标签，请单击标签操作列的“删除”。 警告 删除标签无法恢复，请谨慎操作。 如果配置转储时使用了该标签，删除标签后，请同步修改转储配置信息。关于转储信息请参考日志转储至OBS。
备注	自定义填写备注信息，字符长度0-1024个字符。

步骤4 单击“确定”，完成日志流的创建，在日志组下方生成一条日志流信息。

图 3-4 创建日志流



- 支持查看日志流计费状态，日志计费请参考[价格计算器](#)。
- 按日志流维度上报话单功能目前在友好用户内测中，您可以[提交工单](#)申请开通。

----结束

智能冷存储

说明

冷存储是白名单功能，如有需要请[提交工单](#)申请开通。

云日志服务LTS提供冷存储和标准存储功能。冷存储费用低，响应速度慢，对长期存储的数据进行冷存储。

- 在日志流创建或修改页面，开启智能冷存储功能，将标准存储层数据转为冷存储。
- 在开启智能冷存储功能后，如果标准存储时间增大，冷存储数据的存储时间仅对最新数据生效，冷存储数据不会重新转为标准存储。
- 在开启智能冷存储功能后，在日志分析、仪表盘、定时SQL、管道符搜索方式的查询时间范围查询日志时，不支持查询涉及冷存储的时间段，需将查询时间设置为标准存储的时间范围。

表 3-5 冷存储和标准存储区别

对比项	标准存储	冷存储
适用场景	数据高频查询场景	数据低频查询场景
查询与分析性能	高，秒级返回	较低，存在8S延迟
日志分析、仪表盘等性能	高，秒级返回	不支持此功能
存储价格	0.000479元/GB/小时	0.000208元/GB/小时

修改日志流

日志流默认的存储时间和日志组保持一致。

步骤1 在日志流列表中，单击待修改日志流操作列的“修改”。

步骤2 在弹出框中，支持修改日志流名称、日志存储时间、标签等信息。

支持修改分区数量。单击“查看当前分区详情”进入分区详情页面，查看分区、状态、创建时间。

步骤3 单击“确定”。

步骤4 修改完成后，鼠标悬浮在日志流名称上，显示修改后日志流名称、日志流原始名称、日志流ID。

----结束

删除日志流

如果日志流不再需要使用，可以删除日志流，日志流删除后，日志流中的日志数据将被同时删除。

删除日志流前请确认该日志流是否被其他业务使用，否则删除后可能影响正常的日志上报。例如日志接入配置、日志转储任务、日志告警规则等。

 警告

删除日志流会导致用户日志相关业务异常，日志流删除后无法恢复，请谨慎操作。

- 步骤1 在日志流列表中，单击待删除日志流操作列的“删除”。
- 步骤2 在弹出框中输入“DELETE”或单击“一键输入”，单击“确定”，完成日志流删除。

图 3-5 删除日志流



----结束

其他操作

- 支持通过如下筛选条件进行搜索日志流：
 - 日志流名称/ID：日志流名称或日志流ID（鼠标悬浮在日志流名称上即可查看日志流ID）。
 - 日志流原始名称/ID：日志流修改名称之前的原始名称（第一次创建日志组或日志流时定义的原始名称）、日志流ID（鼠标悬浮在日志流名称上即可查看日志流ID）。
 - 企业项目：创建日志流选择的企业项目。
 - 标签：日志流设置的标签。
 - 备注：创建日志流时自定义的备注信息。
- 收藏日志流：单击日志流中操作列“更多 > 编辑收藏”，收藏日志流，即可在我的收藏/我的收藏（本地缓存）中展示已收藏的日志流。
- 查看日志流详情：单击日志流中操作列下“更多 > 详情”，可查看日志流详情。包括日志流名称、日志流ID、创建时间等信息。
- ICAgent采集诊断：需要先开启“ICAgent诊断开关”，请参考[设置ICAgent日志采集开关](#)。单击目标日志流操作列“更多 > ICAgent采集诊断”，可查看该日志流的ICAgent异常监控、ICAgent采集监控、ICAgent整体状态。更多信息请参考[采集诊断仪表盘模板](#)。
- 查看流量监控：单击目标日志流操作列的“更多 > 流量监控”，可查看该日志流的资源用量明细和用量Top排序。（白名单功能，开通可观测后才能看到，详细请参考[设置LTS日志采集配额和使用量预警](#)。）

- 支持分别按日志流名称或创建时间对日志流进行升序或降序。

相关文档

LTS支持通过调用API接口实现创建日志流、修改日志流、删除日志流、查询日志流信息等，详细请参考[日志流管理](#)。

3.4 查看日志管理

在日志管理首页提供资源统计、日志应用、我的收藏/我的收藏（本地缓存）、最近访问、告警统计、最新告警、功能上线公告等信息的展示。

图 3-6 日志管理首页

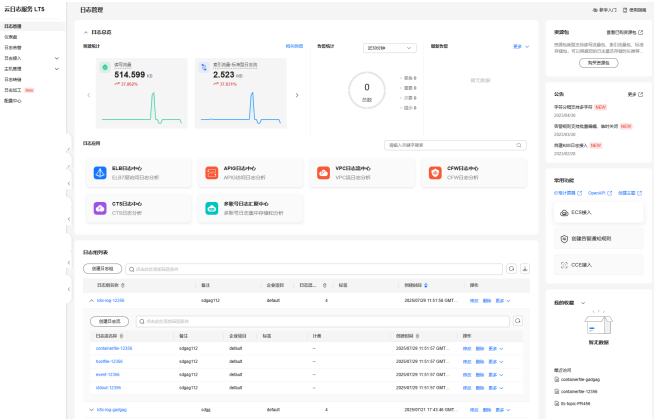


表 3-6 日志管理首页

序号	功能	说明
1	新手入门	新手入门LTS的使用流程。
2	设置主题模式	设置LTS控制台的视觉模式，支持浅色模式、深色模式、跟随系统。
3	资源统计	在日志资源统计页面，支持查看日志组和日志流的读写流量、索引流量、标准存储量、每日读写流量、每日索引流量等资源使用详情。
4	告警统计和最新告警	支持查看告警统计和最新告警。
5	日志应用	- 日志应用支持ELB日志中心、APIG日志中心、VPC日志流中心等，通过日志中心卡片可以快速进入仪表盘和日志接入中心，LTS支持开箱即用的日志仪表盘模板。 - 多账号日志汇聚中心支持多账号日志的集中存储和分析。

序号	功能	说明
6	日志组列表	在日志组列表下方，展示日志组和日志流信息。日志组是云日志服务进行日志管理的基本单位，用于对日志流进行分类，一个日志组下面可以创建多个日志流。将日志数据采集保存到日志流上，通过日志流对日志数据进行搜索分析、日志告警、日志转储、可视化展示、日志加工等操作。
7	资源包	云日志服务LTS支持购买资源包进行计费，在费用结算时，优先从资源包中抵扣用量。
8	公告	公告是对新功能的介绍，展示云日志服务LTS功能的最新动态。
9	常用功能	显示常用使用功能模块，方便快速使用该功能。
10	我的收藏/我的收藏（本地缓存）	我的收藏展示您收藏的日志流，有两种收藏方式：我的收藏和我的收藏（本地缓存）。
11	最近访问	最近访问展示您最近访问的日志流名称，最多可显示3条日志流访问记录。
12	FAQ	FAQ展示常见的问题案例。
13	视频介绍	视频介绍展示常用功能的操作视频，方便您更快的了解使用LTS，例如云日志服务功能介绍、ECS文本日志接入LTS等。

新手入门

步骤1 登录[云日志服务控制台](#)。

步骤2 在日志管理页面右上角，单击展开新手入门。您可以参考如下流程快速使用LTS。

1. [创建日志组](#)
2. [创建日志流](#)
3. [日志接入](#)
4. [日志搜索与分析](#)

图 3-7 新手入门



步骤3 若不需要展示新手入门，在日志管理页面右上角，单击隐藏新手入门。

----结束

设置主题模式

步骤1 单击右侧偏好设置



步骤2 在打开的偏好设置页面，选择LTS控制台显示的主题模式。例如浅色模式、深色模式、跟随系统。

图 3-8 偏好设置



----结束

资源统计

在日志资源统计页面，支持查看资源统计、资源详情。日志资源统计是对日志进行分类统计及日志数据的可视化展示，统计日志资源的数据量仅供参考。

高级版入口仅支持白名单用户使用，在LTS控制台配置中心的“配额设置”页面，开启“可观测For LTS”，即可使用高级版功能。详细请参考[设置LTS日志采集配额和使用量预警](#)。

- 基础版：查看日志资源统计及日志详情数据的可视化展示。
- 高级版：查看日志组或日志流的资源用量明细和用量Top排序。

说明

“可观测For LTS”功能仅支持华北-北京四、华东-上海一、华南-广州、亚太-新加坡区域的白名单用户使用，如有需要，请[提交工单](#)，其他区域暂不支持申请开通。

步骤1 登录[云日志服务控制台](#)。

步骤2 在日志总览下方，单击“相关明细”或“详情-基础版”，进入日志资源统计详情页面。

开启“可观测For LTS”白名单功能后，才会显示“详情-基础版”。

步骤3 资源统计主要展示日志资源数据，默认展示时间为1周（相对）的日志资源数据，您可以根据业务需要选择时间范围。

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。

- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

步骤4 在资源统计下方，查看资源统计数据：

- 读写流量：读写流量根据传输的流量计算，传输流量为压缩后的日志大小，日志一般有5倍压缩率。
- 索引流量-标准型日志流：原始日志数据默认都会建立全文索引，创建索引（对日志分词处理）后，才能搜索日志。在日志写入数据库时一次性收取流量费用。
- 标准存储量：日志标准存储量为压缩后的日志数据、索引数据、副本数据之和，这些空间约等于原始日志数据大小。标准存储量统计是指在选定时间范围内最新的存储总量。
- 原始日志流量：原始日志数据的大小。
- 基础转储流量：开启日志转储计费功能后，才会显示基础转储流量。日志转储不需要字段映射，转储算力消耗少，例如转储OBS/DMS/DIS。
- 高级转储流量：开启日志转储计费功能后，才会显示高级转储流量。日志转储需要字段映射，转储算力消耗较多，例如转储DWS/DLI。

步骤5 在资源详情下方，查看Top100的日志组资源统计和Top100的日志流资源统计。根据选择的时间范围，支持表格或柱状图展示每日读写流量（GB）、每日索引流量-标准型日志流（GB）、每日标准存储量（GB）、每日基础转储流量（GB）、每日高级转储流量（GB）。

- 新创建的日志组/日志流，需间隔至少1小时才能进行资源统计。
- 单击Top100中的日志组名称，可查询该日志组下的日志流资源统计。
- 单击 ，可下载日志组资源统计和日志流资源统计。下载的日志组资源统计和日志流资源统计文件为.CSV格式。

步骤6 返回日志总览页面，单击资源统计旁边的详情-高级版，在弹出的页面查看资源用量明细和用量Top排序。

- 在“资源用量明细”页签：查看日志组和日志流的原始日志流量、索引流量、写流量等信息。如果当前版本不支持某个功能，LTS界面的图表展示暂无数据。
- 在“用量Top排序”页签：查看Top日志组和Top日志流的原始日志流量、索引流量、写流量等信息。如果当前版本不支持某个功能，LTS界面的图表展示暂无数据。

----结束

告警统计和最新告警

在日志总览下方，支持查看告警统计和最新告警。

- 告警统计展示云服务日志的告警总数及各个告警级别的数量。告警统计时间有：近30分钟、近1小时、近6小时、近1天和近1周；告警级别包括紧急、重要、次要和提示。
- 最新告警展示最新创建的告警规则，最多可显示近30分钟的3条告警规则。
- 若需要查看更多告警，您可以单击“更多 > 更多告警”。
- 若需要添加告警，您可以单击“更多 > 添加告警”。

日志应用

在日志管理页面的日志总览下方，支持多种日志应用，即LTS提供开箱即用的日志仪表盘模板，用户接入即可进行日志的快速分析，主要有如下几个应用：

- ELB日志中心：支持ELB日志接入LTS和ELB仪表盘模板。详情请参考[ELB仪表盘模板（共享型）](#)。
- APIG日志中心：支持APIG日志接入LTS和APIG仪表盘模板。详情请参考[APIG仪表盘模板](#)。
- VPC日志流中心：支持VPC日志接入LTS和VPC仪表盘模板。详情请参考[VPC仪表盘模板](#)。
- CFW日志中心：支持CFW日志接入LTS和CFW仪表盘模板。详情请参考[CFW仪表盘模板](#)。
- CTS日志中心：支持CTS日志接入LTS，暂不支持CTS仪表盘模板，支持CTS接入LTS。详情请参考[云审计服务CTS接入LTS](#)。
- 多账号日志汇聚中心：支持您将多个账号下的日志流复制到指定的账号中，实现多账号日志的集中存储和分析。详细请参考[设置多账号日志汇聚](#)。
- 可观测For LTS：提供分钟粒度的日志流量监控数据。白名单功能，详细请参考[设置LTS日志采集配额和使用量预警](#)。

日志组列表

在日志组列表下方，展示日志组和日志流信息，更多信息请参考[管理日志组](#)、[管理日志流](#)。

资源包

云日志服务支持购买资源包进行计费，在费用结算时，优先从资源包中抵扣用量。详细请参考[购买LTS资源包](#)。

公告

公告是对新功能的介绍，展示云日志服务功能的最新动态。

如需查看更多功能介绍，您可以单击[更多](#)。

图 3-9 公告



常用功能

显示常用使用功能模块，方便快速使用该功能。固定展示模块：价格计算器、OpenAPI、创建主题，下方会显示常用的三个功能模块，单击模块名称即可快速跳转到对应功能的使用页面。

图 3-10 常用功能



我的收藏/我的收藏（本地缓存）

我的收藏展示您收藏的日志流，有两种收藏方式：我的收藏和我的收藏（本地缓存）。

- **我的收藏：**将日志流保存至LTS数据库中，默认为关闭状态。当您的账号开通写权限时，可显示该功能和我的收藏（本地缓存）。
- **我的收藏（本地缓存）：**将日志流保存至浏览器本地缓存，默认为关闭状态。清除浏览器记录后收藏记录就会被删除。可写用户和只读用户支持显示我的收藏（本地缓存）。

当您的账号开通写权限时，**我的收藏/我的收藏（本地缓存）**至少有一个是开启状态，否则无法收藏日志流。

您可以通过云日志服务提供的收藏功能个性化定制属于自己的收藏日志流列表，方便您直接、快速的定位到常用的日志流。

以日志组lts-test为例，收藏日志组lts-test下某个日志流的操作步骤如下：

- 步骤1** 在日志管理页面的日志组列表下方，单击日志组lts-test对应的 ，选择待收藏的日志流。
- 步骤2** 单击日志流操作列的“更多 > 编辑收藏”，选择收藏方式，单击“确定”，即可收藏日志流。
- 步骤3** 收藏成功后，在右侧我的收藏/我的收藏（本地缓存）展示框中，即可展示您收藏的日志流信息。

----结束

最近访问

最近访问展示最近访问的日志流。最近访问最多可显示3条日志流访问记录。

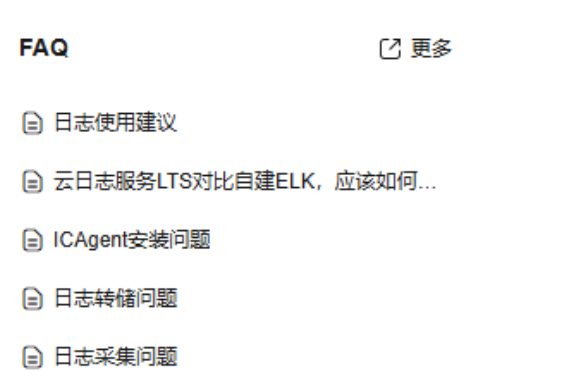
图 3-11 最近访问



FAQ

FAQ（常见问题）展示经常被询问的问题。如需查看更多FAQ，您可以单击[更多](#)。

图 3-12 更多



视频介绍

通过视频介绍云日志服务的功能、ECS文本日志接入LTS、CCE应用日志接入LTS、安装ICAgent。

图 3-13 视频介绍



3.5 设置多账号日志汇聚

多账号日志汇聚中心支持您将多个账号下的日志流复制到指定账号中，实现多账号日志的集中存储和分析，满足安全合规、集中分析等不同场景下的诉求。

为了满足安全审计或运营分析要求，集团公司的安全合规部门或运营部门可以通过日志汇聚功能，将不同账号下各个业务部门的关键日志统一汇聚到一个日志账号中，实现集中存储和分析，便于后续进行大数据处理和可视化分析。

说明

目前此功能支持华东-上海一、华东-上海二、华东-青岛、华东二、中国-香港、亚太-曼谷、亚太-新加坡、亚太-雅加达、华南-广州、华南-广州-友好用户环境、拉美-圣保罗一、华北-北京一、华北-北京四、华北-乌兰察布一、西南-贵阳一、拉美-圣地亚哥、非洲-开罗、亚太-马尼拉、中东-利雅得、非洲-约翰内斯堡、土耳其-伊斯坦布尔、拉美-墨西哥城一、拉美-墨西哥城二，其他局点需要提交工单申请才能使用。详细操作请参考[提交工单](#)。

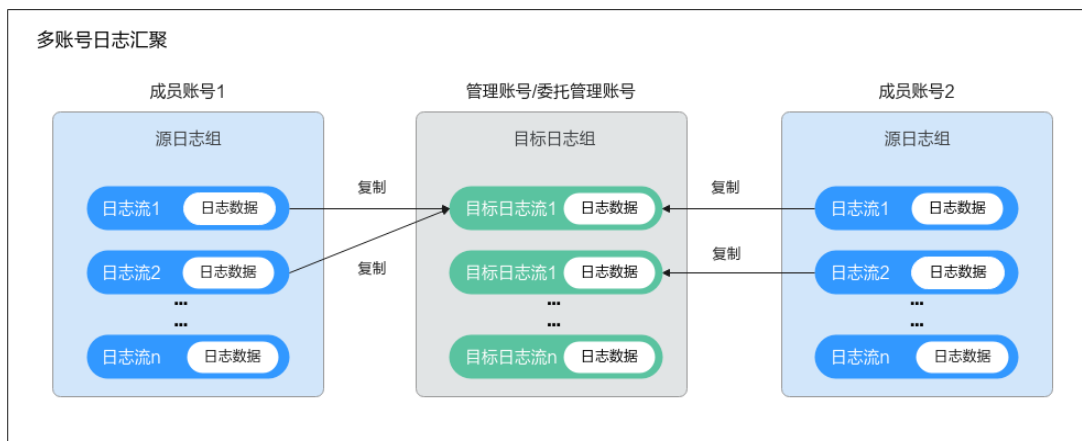
背景信息

组织（以下简称Organizations）云服务为企业用户提供多账号关系的管理能力。Organizations支持用户将多个华为云账号整合到创建的组织中，并可以集中管理组织下的所有账号。用户可以在组织中设置访问策略，帮助用户更好地满足业务的安全性和合规性需求。更多组织信息请参考[什么是组织云服务](#)。

云日志服务LTS联合Organizations推出多账号日志汇聚中心，您可以在Organizations使用管理账号指定某个成员账号成为LTS可信服务的委托管理员账号，然后可以在LTS将组织下某个成员账号的日志流复制到管理账号或者委托管理员账号中，实现多账号日志集中汇聚的目的。

某个成员账号的源日志组/日志流实际上是被复制一份到管理账号或者委托管理员账号对应的目标日志组/日志流中，因此两个账号间的日志组/日志流之间不会互相干扰，可以在各自账号下独立配置转储、消费、加工等任务。

图 3-14 日志汇聚



前提条件

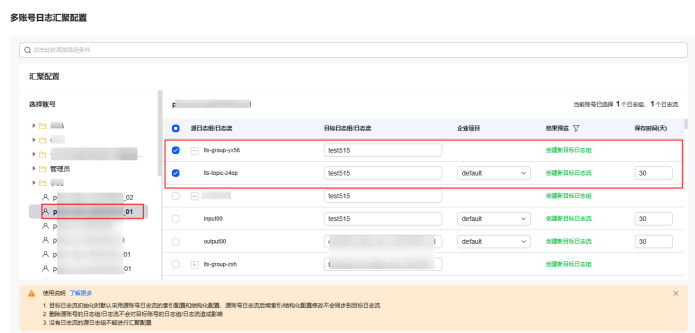
- 已创建组织。更多信息请参考[创建组织](#)。
- 已设置service.LTS为可信服务。更多信息请参考[启用、禁用可信服务](#)。
- 赋予管理账号或者委托管理员账号拥有Organizations服务只读权限。关于委托账号的操作请参考[添加、查看和取消委托管理员](#)。关于授权的操作请联系企业管理人员参考[创建IAM用户并授权管理组织](#)。

设置多账号日志汇聚任务

- 步骤1** 使用管理账号或者委托管理员账号登录管理控制台，选择“云日志服务 LTS”进入日志管理页面。
- 步骤2** 在日志应用下方，单击“多账号日志汇聚中心”。

步骤3 在多账号日志汇聚配置页面，开启日志接收状态，左侧选择某个成员账号，右侧勾选对应的源日志组/日志流，自定义目标日志组/日志流的名称。

图 3-15 汇聚账号



- 若不需要使用日志汇聚配置功能时，可以关闭日志接收状态，所有汇聚配置都会失效，源日志流停止汇聚到目标日志流上。
- 支持创建日志流的数量不超过总数量。更多信息请参考[基础资源限制](#)。

步骤4 设置完成后，单击“确定”，该账号汇聚配置创建中，预计5分钟左右创建完成。然后返回日志管理页面，在日志组列表下方查看到新建的日志组/日志流信息，则代表汇聚目标日志组/日志流创建成功。

图 3-16 汇聚成功



- 目标日志流初始化时，默认采用源账号日志流的索引配置和结构化配置。配置成功后，源账号日志流修改索引/结构化配置时，不会同步到目标日志流。
- 删除源账号的日志组/日志流不会对目标账号的日志组/日志流造成影响。
- 取消勾选日志组/日志流并成功下发配置后，该日志组/日志流将不再继续汇聚。

----结束

相关文档

LTS支持通过调用API接口实现获取日志汇聚开关、修改日志汇聚开关等，详细请参考[多账号日志汇聚](#)。

4 日志接入

4.1 日志接入概述

应用程序和服务在运行过程中会产生大量日志数据，包括系统运行状态、错误信息和用户操作记录等。这些数据在系统运维、故障排查和业务分析等关键环节中发挥着重要作用。

随着日志数据量的增加和格式的多样化，管理和分析这些数据变得更具挑战性。日志接入能够帮助您高效地采集日志数据，并将其保存到LTS中。

日志接入成功后，您还可以通过LTS使用日志搜索、日志分析和日志告警等功能，进一步提升应用运维效率和业务分析洞察力。

了解更多日志接入的概念信息，请参考[基本概念](#)。

费用说明

日志接入会产生费用，更多信息请参考[计费项](#)。

日志接入方式

LTS提供实时日志接入功能，支持通过ICAgent插件、云服务接入、API接入、SDK接入等多种方式将采集到的日志上报到LTS。日志接入LTS后，用户就可以在云日志服务控制台进行一系列操作，例如搜索与分析日志、使用统计图表或仪表盘可视化展示日志统计结果、设置日志告警、设置日志转储等。

云日志服务LTS数据采集的总体概览如下图：

图 4-1 数据采集

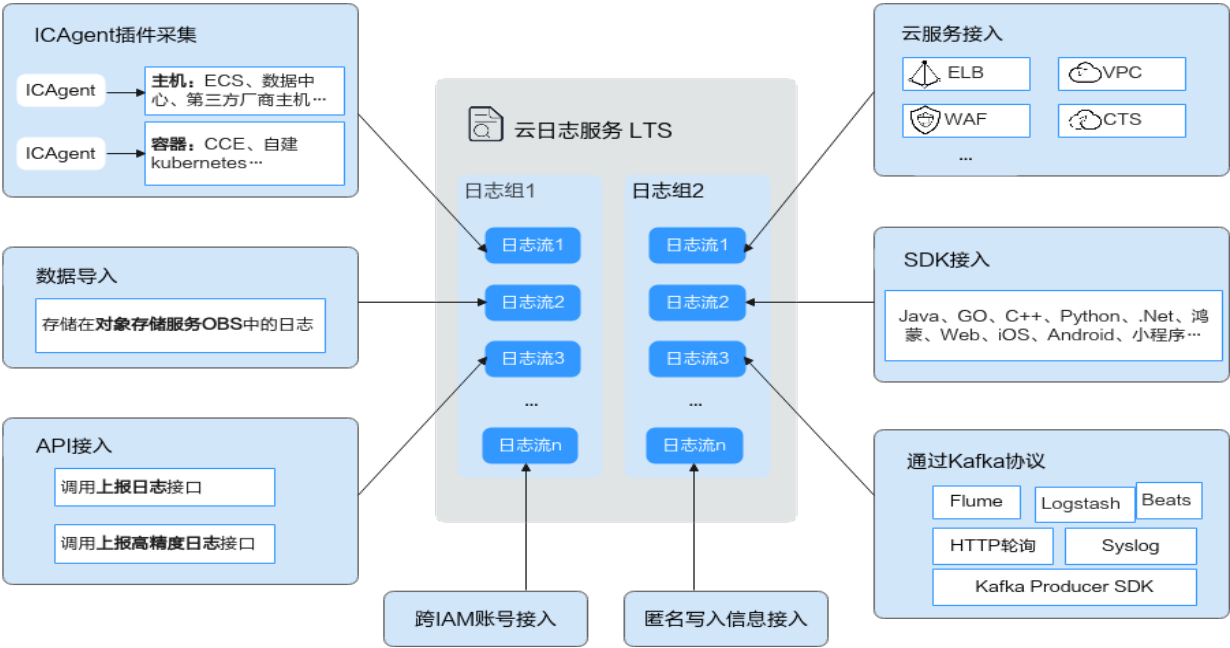


表 4-1 数据采集方式

采集方式	说明
ICAgent插件采集	ICAgent是云日志服务进行日志采集的插件工具，运行在需要采集日志的主机中。通过ICAgent插件，可以实现无侵入主机文本或容器应用场景的日志，将日志接入到LTS，实现日志统一管理。 - 采集主机场景日志，ICAgent插件安装运行在待采集日志的主机上，请参考使用ICAgent插件采集主机日志。 - 采集容器场景日志，ICAgent插件安装运行在待采集日志的集群中，请参考使用ICAgent插件采集容器日志。
云服务接入LTS	LTS支持采集ELB、WAF、APIG、CFW等云服务日志数据，例如性能日志（资源利用率）、网络日志（流量记录）等信息。通过LTS统一管理云服务日志，您可以快速搜索日志、分析日志，配置日志告警，及时掌握云服务的运行状态，提升运维效率，请参考 使用云服务接入LTS 。
数据导入	如果您有存储在OBS桶中的日志文件，可以使用数据导入的方式，将对象存储服务桶内的文件一次性或定期导入到云日志服务日志数据导入到LTS，请参考 对象存储服务OBS接入LTS（公测） 。 说明 目前此功能在公测中，暂不支持申请开通。
SDK接入	LTS提供Java SDK、Go SDK、Web SDK、iOS SDK、Android SDK等多种语言SDK。通过封装SDK，直接进行二次开发提升开发效率，各语言SDK功能对齐，降低多端适配成本，请参考 使用SDK接入LTS 。

采集方式	说明
API接入	通过调用LTS提供的上报日志接口和上报高精度日志接口，即可将日志数据接入LTS，实现日志集中管理。 • 上报日志接口：上报在相近时间点且按顺序产生的日志。 • 上报高精度日志接口：上报在不同时间且非顺序产生的日志。
跨IAM账号接入	通过创建委托，您可以将委托账号的日志流映射到被委托方账号的日志流下，被委托账号即当前云日志服务登录账号。
通过标准Kafka协议接入	您可以通过标准Kafka协议将日志上报到云日志服务，这种方式仅依赖于Kafka协议，支持各类Kafka Producer SDK或采集工具。比如，如果您正在使用Logstash作为日志采集工具，只需修改配置文件，就可以将日志直接上报到云日志服务，无需额外安装其他组件。请参考 使用KAFKA协议上报日志到LTS 。
使用匿名写入采集日志	如果您需要收集和分析用户在浏览器、小程序上的信息，例如用户的浏览器、浏览行为记录、购买行为记录、停留时间等匿名写入信息，可以使用匿名写入功能。请参考 使用匿名写入采集日志 。

ICAgent 插件介绍

ICAgent是云日志服务的日志采集工具，运行在需要采集日志的主机中。使用ICAgent插件采集日志具有如下优势：

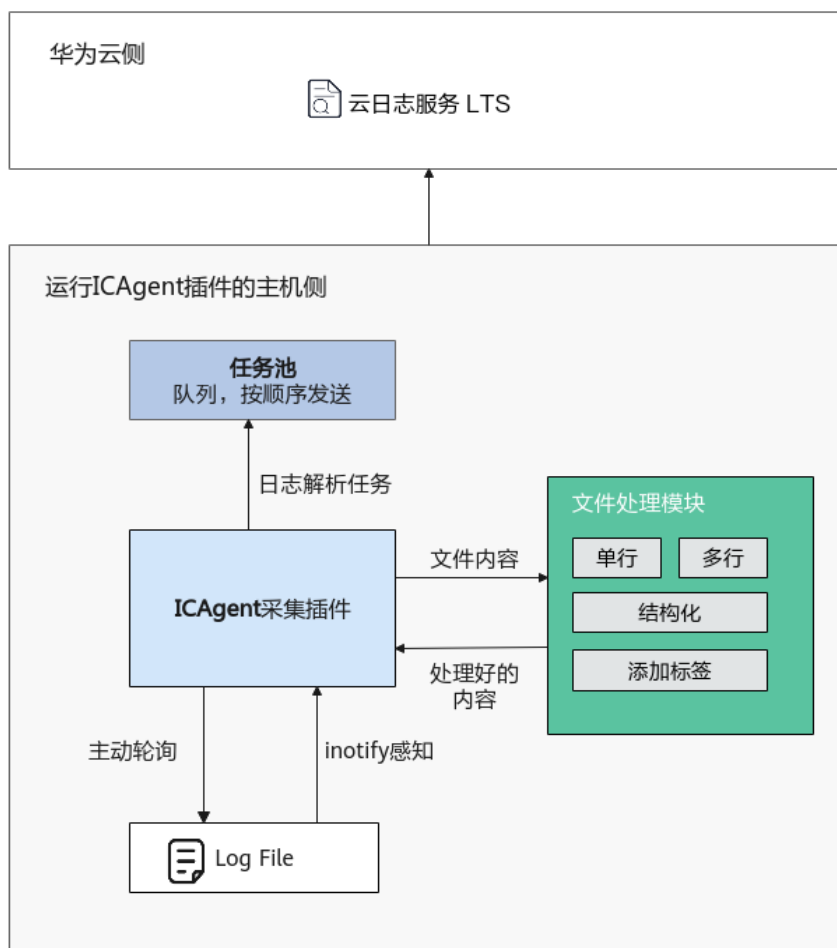
- 直接采集现有的日志文件，无需修改应用程序代码，不会影响应用程序运行。
- 稳定处理日志采集过程中的各种异常。当遇到网络异常、服务端异常等问题时会采用主动重试，本地持久化采集游标，记录采集日志位置等措施保障数据安全。
- 基于云日志服务的集中管理能力。安装ICAgent后，只需要在云日志服务上配置主机组、ICAgent采集配置等信息即可。
- 完善的自我保护机制。为保证运行在服务器上的ICAgent，不会明显影响您服务器上其他服务的性能，ICAgent在CPU、内存及网络使用方面都做了严格的限制和保护机制。

ICAgent采集原理

ICAgent侧收到LTS接入页面下发的采集配置后，通过轮询和监控的方式感知日志文件变化，会对所关联的日志文件进行实时监听。定时解析采集配置信息，获取到采集路径后在节点上进行路径匹配，将节点上对应的文件加入监控任务中，通过轮询和inotify（Linux内核提供了一种文件系统事件监控机制）感知目标日志文件的变化。

ICAgent侧发现文件变更后，读取文件的内容，分块发送到处理模块，根据用户的采集配置进行单行、多行、结构化、拆分、添加标签等日志处理动作，再将处理好的任务提交到发送任务池后，上报到LTS。

图 4-2 ICAgent 采集原理



ICAgent 安装说明

如果需要采集节点日志、容器日志、标准输出日志等，则需要在采集日志的机器上安装 ICAgent。

请参考[安装 ICAgent（区域内主机）](#)和[安装 ICAgent（区域外主机）](#)。

- 区域内主机就是主机所在区域和用户登录云日志服务控制台所在的区域相同，例如华北-北京四。
- 区域外主机就是主机所在区域和用户登录云日志服务控制台所在区域不同，例如华为云其他区域、第三方都属于区域外主机。

ICAgent 结构化解析规则

ICAgent 除了提供日志采集功能外，还支持在采集过程中对日志数据进行结构化处理。在日志接入前，您可以提前了解 ICAgent 结构化解析规则，然后根据日志内容选择合适的解析规则。详情请参考[配置 ICAgent 结构化解析](#)。

除此之外，LTS 还提供了云端结构化解析的能力，详情请参考[设置云端结构化解析日志](#)。但在 ICAgent 插件采集场景更推荐 ICAgent 结构化解析。一个日志流只能配置一种日志数据结构化方式，例如，日志流在配置 ICAgent 结构化解析后，不能再配置云端结构化解析。

- ICAgent结构化解析是通过用户主机上安装的ICAgent进行结构化，支持插件组合解析，单个日志流的多个采集配置支持不同结构化解析规则，**推荐使用ICAgent结构化解析的方式**，更多内容请参考[配置ICAgent结构化解析](#)。
- 云端结构化解析是通过不同的日志提取方式将日志流中的日志进行结构化，云端结构化解析会消耗LTS服务端算力，未来会按照日志量大小收取日志加工流量费用。

在不同的接入场景下选择日志数据的结构化解析方式可以参考图4-3。

图 4-3 结构化解析方式的选择



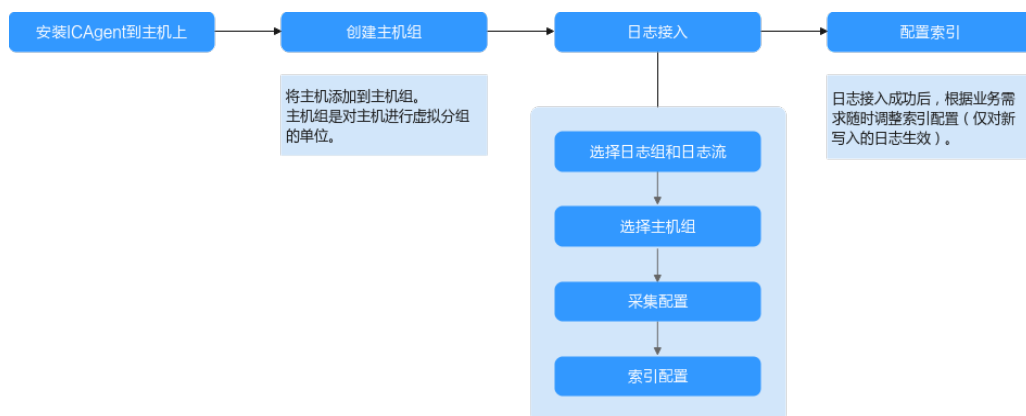
4.2 使用 ICAgent 插件采集主机日志

4.2.1 ICAgent 采集主机场景日志概述

云日志服务LTS支持通过ICAgent采集主机上的日志，适用于物理服务器、云服务器ECS等传统计算环境。

通过ICAgent采集主机场景日志的流程请参考图4-4。

图 4-4 ICAgent 使用流程



1. **安装 ICAgent 到主机上**：ICAgent 是云日志服务的日志采集工具，运行在需要采集日志的主机中。根据主机所在位置分为区域内主机和区域外主机，详细操作请参考[安装 ICAgent（区域内主机）](#)、[安装 ICAgent（区域外主机）](#)。
2. **创建主机组**：将已安装 ICAgent 的主机加入主机组。详细请参考[管理 LTS 主机组](#)。
3. **日志接入**：LTS 以主机组为单位下发采集配置到 ICAgent 侧，方便您对多台主机日志进行采集，ICAgent 侧按照采集配置规则上报日志到 LTS。
 - [云主机 ECS 文本日志接入 LTS](#)
 - [裸金属服务 BMS 文本日志接入 LTS](#)
 - [ServiceStage 云主机日志接入 LTS](#)
4. **配置索引**：日志接入成功后，您可以前往索引配置页面，配置全文索引或字段索引，通过日志搜索与分析查看采集上来的日志数据。详细请参考[创建 LTS 日志索引](#)。

4.2.2 安装 ICAgent（区域内主机）

区域内主机是指用户登录云日志服务控制台所在 Region 区内的主机，例如北京四。如果您想使用云日志服务对区域内主机进行日志采集，例如节点日志、容器日志、标准输出日志等，那么您需要在区域内主机上需要安装 ICAgent。ICAgent 是云日志服务进行日志采集的工具，运行在需要采集日志的主机中。

前提条件

安装 ICAgent 前，请确保本地浏览器的时间、时区与主机的时间、时区一致。如果不一致，可能会导致日志上报出错。

安装 ICAgent 的限制说明

- Linux 环境：ICAgent 支持的 Linux 操作系统，请参考[Linux 操作系统](#)。
- Windows 环境：仅支持在如下 64 位系统的 Windows 环境中安装 ICAgent。
 - Windows Server 2016 R2 Datacenter
 - Windows Server 2016 R2 Standard
 - Windows Server 2016 Datacenter English
 - Windows Server 2016 R2 Standard English
 - Windows Server 2012 R2 Datacenter
 - Windows Server 2012 R2 Standard
 - Windows Server 2012 Datacenter English
 - Windows Server 2012 R2 Standard English

```
Windows Server 2008 R2 Enterprise
Windows Server 2008 R2 Standard
Windows Server 2008 Enterprise English
Windows Server 2008 R2 Standard English
```

- Windows环境不支持在云日志服务主机管理界面对ICAgent进行升级和卸载操作。如果需要使用新版本，请先在Windows环境中卸载旧版本ICAgent，再安装新版本ICAgent。

安装场景说明

ICAgent有两种安装场景，请按照您的需要进行选择。

表 4-2 安装场景

场景	说明
首次安装	该服务器未安装过ICAgent。 在服务器上执行如下命令，若没有ICAgent进程，则确认该服务器没有安装过ICAgent。 <pre>ps -aux grep icagent</pre> - 首次安装（Linux环境） - 首次安装（Windows环境）
继承安装（Linux环境支持）	您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，对于没有安装ICAgent的其他多个服务器，您可以采用该安装方式。 - 继承安装（Linux环境） - 继承批量安装（Linux环境）

首次安装

若服务器未安装过ICAgent，您可以参考如下操作在Linux环境或Windows环境首次安装ICAgent。

首次安装（Linux 环境）

- 步骤1 登录管理控制台，选择“管理与部署 > 云日志服务 LTS”，进入“日志管理”页面。
- 步骤2 登录[云日志服务控制台](#)。
- 步骤3 左侧导航栏选择“主机管理 > 主机”，进入“主机”页面。
- 步骤4 单击右上角“安装ICAgent”。

安装ICAgent前，请确保本地浏览器的时间、时区与主机的时间、时区一致。

图 4-5 安装 ICAgent



步骤5 主机类型选择“区域内主机”。

步骤6 安装系统选择“Linux”。

步骤7 安装方式选择“获取AK/SK凭证”时，需要设置该步骤。不需要再设置步骤**步骤8**。请提前获取访问密钥，安装 ICAgent 时需要输入 AK/SK。通过 AK 识别访问用户的身份，通过 SK 对请求数据进行签名验证，用于确保请求的机密性、完整性和请求者身份的正确性。请确保公共用户账号及其创建的 AK/SK 不会被删除或禁用。AK/SK 被删除，会导致安装的 ICAgent 无法正常上报数据到 LTS。

详细操作请参考[如何获取访问密钥AK/SK](#)。

终端节点用于在 VPC 和终端节点服务之间建立便捷、安全、私密的连接通道。如果您需要通过终端节点的方式打通 AOM/LTS 的网络，详细操作请参考[如何在VPCEP控制台创建终端节点？](#)

步骤8 安装方式选择“创建IAM委托”时，需要设置该步骤。该方式不需要获取访问密钥，通过创建委托 ICAgent 可以自动获取 AK/SK（访问密钥），安装 ICAgent 时不需要输入 AK/SK。

详细操作请参考[创建IAM委托](#)。

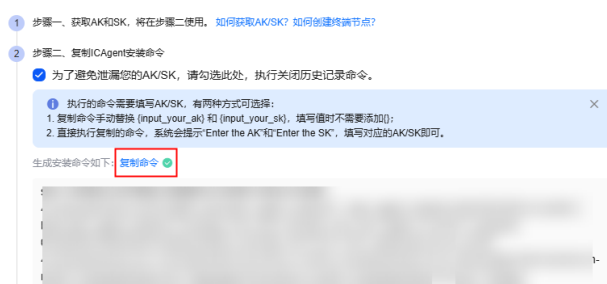
1. 登录统一身份认证服务控制台。
2. 在左侧导航栏单击“委托”，进入委托页面。
3. 单击右上角“创建委托”，具体操作请参考[委托其他云服务管理资源](#)，创建委托时参数设置要求参考如下。
 - a. 委托类型：选择“云服务”。
 - b. 云服务：选择“弹性云服务器 ECS 裸金属服务器 BMS”。
 - c. 持续时间：选择“永久”。
 - d. 授权权限：需要同时添加 LTS Administrator 和 APM Administrator，授权后需等待 15-30 分钟才可生效。
 - e. 权限的作用范围：指定区域项目资源。

4. 委托创建成功后，设置委托生效。
 - a. 进入弹性云服务器控制台。
 - b. 单击需要安装ICAgent的弹性云服务器名称，进入弹性云服务器参数配置页面。

您可以在购买ECS机器时设置委托，请在“购买弹性云服务器”页面，“高级配置”下方的“委托”中选择已创建的委托名称。待剩余参数配置完成后，单击“立刻购买”即可。
 - c. 在管理信息下方的“委托”后，单击 ，选择已创建的委托名称，单击  即可生效。更多关于委托的信息，请参见[委托其他账号管理资源](#)。

步骤9 在“安装ICAgent”页面，单击“复制命令”，复制ICAgent安装命令。

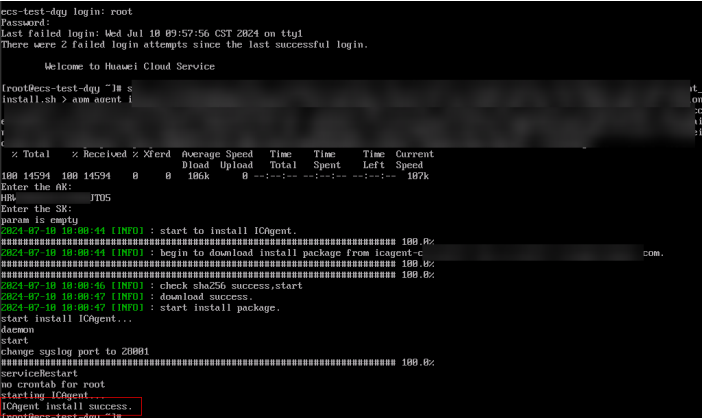
图 4-6 安装 ICAgent



步骤10 以登录华为云ECS主机为例，介绍如何登录到待安装主机执行ICAgent安装命令，请以实际采集主机为准。详细操作请参考[远程登录Linux弹性云服务器（VNC方式）](#)。

1. 进入弹性云服务器ECS控制台。
2. 找到需要安装ICAgent的ECS主机，单击目标主机操作列的“远程登录”。
3. 在弹出的“登录Linux云服务器”窗口中，选择“其他方式”下的VNC方式，单击“立即登录”。
4. 在新打开的页面中，根据界面提示，输入用户购买弹性云服务器设置的root用户名和密码。
5. ECS登录成功后，执行ICAgent安装命令：
 - “安装方式”选择“获取AK/SK凭证”，执行ICAgent安装命令进行安装，并根据提示输入已获取到的AK/SK。（若复制命令时手动替换了AK/SK，则系统不会再提示输入AK/SK）
 - “安装方式”选择“创建IAM委托”，直接执行ICAgent安装命令进行安装即可。（ICAgent可以自动获取AK/SK，不需要输入AK/SK）
6. 当显示“ICAgent install success”时，表示安装成功，ICAgent已安装在/opt/oss/servicemgr/目录。

图 4-7 安装回显



步骤11 安装成功后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看该主机的ICAgent状态显示“运行”。

当前网卡信息由节点上安装的ICAgent插件通过执行ifconfig命令获取。若网卡配置中没有inet信息，则不会上报IPv4地址，即在LTS主机页面的“主机IPv4”显示--；若未配置inet6信息，则不会上报IPv6地址，即在LTS主机页面的“主机IPv6”显示--。

图 4-8 主机

主机ID	主机名称	主机类型	主机状态	主机IPV4	主机IPV6	主机创建时间	主机更新时间
1-10000000000000000000	lts-10000000000000000000	区域内主机	运行	10.0.0.1	2000::1	2025-09-17 10:00:00	2025-09-17 10:00:00

----结束

首次安装（Windows 环境）

- 步骤1** 在“主机”页面，单击右上角“安装ICAgent”。
- 步骤2** 主机类型选择“区域内主机”。
- 步骤3** “安装系统”选择“Windows”。
- 步骤4** 您可以通过单击界面提供的ICAgent压缩包或者下载地址，下载ICAgent安装包到本地。
- 步骤5** 将ICAgent安装包存放至待安装ICAgent服务器的目录（如：C:\ICAgent）并解压。
- 步骤6** 获取AK/SK方法请参考：[如何获取访问密钥AK/SK](#)。
- 步骤7** 在“安装ICAgent”页面，单击“复制命令”，复制ICAgent安装命令到本地并手动替换AK/SK。
- 步骤8** 打开cmd窗口并进入ICAgent安装包的解压目录，执行ICAgent安装命令进行安装。

当显示“Service icagent installed successfully”时，表示安装成功。

- 如果安装了第三方杀毒软件，需要添加ICAgent为信任程序，否则可能导致ICAgent安装失败。
 - 如果需要卸载ICAgent，请在ICAgent安装包解压目录下，双击执行“ICAgent安装包解压目录\ICProbeAgent\bin>manual\win\uninstall.bat”脚本，当显示“icagent removed successfully”时，表示卸载成功。
- 卸载ICAgent不会删除对应目录的文件，请您根据实际情况自行删除。

- 查询ICAgent的状态，请在ICAgent安装包解压目录下，打开cmd窗口，执行如下命令，状态为RUNNING，表示ICAgent正在运行中；提示“The specified service does not exist as an installed service”或者“指定的服务未安装”，表示ICAgent已卸载。

```
sc query icagent
```

- 卸载后重新安装ICAgent，如果一直处于“pending”状态，可以在任务管理器中结束icagent.exe进程，然后再次重新安装ICAgent。

步骤9 安装成功后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看该主机的ICAgent状态显示“运行”。

当前网卡信息由节点上安装的ICAgent插件通过执行ifconfig命令获取。若网卡配置中没有inet信息，则不会上报IPv4地址，即在LTS主机页面的“主机IPv4”显示--；若未配置inet6信息，则不会上报IPv6地址，即在LTS主机页面的“主机IPv6”显示--。

----结束

继承安装

如果您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，对于没有安装ICAgent的其他多个服务器，您可以采用继承安装或继承批量安装的方式。

继承安装（Linux 环境）

如果您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent的安装包ICProbeAgent.tar.gz，对于没有安装ICAgent的服务器，可以通过该方式对服务器进行一键式继承安装。

继承安装的服务器需同属一个VPC下，并在同一个网段中。

步骤1 在已安装ICAgent的服务器上执行如下命令，其中x.x.x.x表示待安装ICAgent服务器的IP地址。

```
bash /opt/oss/servicemgr/ICAgent/bin/remotelInstall/remote_install.sh -ip x.x.x.x
```

步骤2 根据提示输入待安装ICAgent的服务器root用户密码。

- 如果已安装ICAgent的服务器安装过expect工具，执行上述命令后，即可完成安装。如果已安装ICAgent的服务器未安装expect工具，请根据提示输入密码，进行安装。
- 请确保已安装ICAgent的服务器可以使用root用户执行SSH、SCP命令，来与待安装ICAgent的服务器进行远端通信。
- 当显示“Install ICAgent success”时，表示安装成功，ICAgent已安装在了“/opt/oss/servicemgr/”目录。安装成功后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看该服务器ICAgent的状态。
- 如果没有显示“Install ICAgent success”代表安装失败，请卸载ICAgent后重新安装。详细请参考[卸载ICAgent](#)。

----结束

继承批量安装（Linux 环境）

如果您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent的安装包

ICProbeAgent.tar.gz，对于没有安装ICAgent的服务器，可以通过该方式对服务器进行一键式继承批量安装。

- 批量安装的服务器需同属一个VPC下，并在同一个网段中。
- 批量安装功能依赖python3.*版本，如果安装时提示找不到python请在服务器上安装python版本后重试。

前提条件

已收集需要安装ICAgent的所有服务器的IP地址、root密码，按照iplist.cfg格式整理好，并上传到已安装过ICAgent机器的/opt/ICAgent/目录下。iplist.cfg格式示例如下所示，IP地址与服务器root密码之间用空格隔开：

192.168.0.109密码（请根据实际情况填写）

192.168.0.39密码（请根据实际情况填写）

- iplist.cfg中包含您的敏感信息，建议您使用完之后进行清理。
- 如果所有服务器的密码一致，在iplist.cfg中只需列出IP，不需要列出密码。然后执行时需要输入此密码即可；如果某个IP密码与其他不一致，则需在此IP后填写其密码。

请参考如下方法：

步骤1 在已安装ICAgent的服务器上执行如下命令。

```
bash /opt/oss/servicemgr/ICAgent/bin/remotelyInstall/remote_install.sh -batchModeConfig /opt/ICAgent/iplist.cfg
```

根据脚本提示输入待安装服务器的root用户默认密码，如果所有IP的密码在iplist.cfg中已有配置，则直接输入回车键跳过即可，否则请输入默认密码。

请耐心等待，当提示“Install ICAgent success”时，则表示配置文件中的所有服务器安装操作已完成。

步骤2 安装完成后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看主机的[查看ICAgent状态](#)。

----结束

相关文档

如果在安装ICAgent时遇到问题，您可以参考[主机管理常见问题](#)获取解决方案。

4.2.3 安装 ICAgent（区域外主机）

区域外主机指的是华为云非当前Region或非华为云主机，例如自建IDC（Internet Data Center，互联网数据中心）、第三方云厂商或华为云跨Region主机等。如果您想使用云日志服务对区域外主机进行日志采集，那么您需要先保证区域外主机与华为云当前区域LTS网络互通，再在区域外主机上安装ICAgent。ICAgent是云日志服务进行日志采集的工具，运行在待采集日志的主机中。

按照区域外主机上报日志到LTS的网络通路来划分，可分为公网和专线：

- 公网：区域外主机与公网连通，那么区域外主机可以直接通过公网与华为云当前区域LTS进行通信，将日志上报的LTS。但是由于公网的安全性可能较低，因此在实际生产环境中通常选择专线接入方式。

区域外主机安装ICAgent选择公网方式时，该局点必须支持公网上报日志。

- 专线：区域外主机与华为云当前区域LTS通过跳板机或VPCEP的方式进行连通，安全性和稳定性都更高。在专线连通场景下，默认区域外主机与当前区域LTS网络不通，安装在区域外主机上的ICAgent无法直接访问华为云管理面上报日志的网段。因此需要配置网络打通方案，使用跳板机或VPCEP的方式去连通LTS后端将数据转发到LTS。
 - 跳板机：跳板机作为数据转发器，将区域外主机ICAgent采集到的数据转发给LTS。当您在进行测试，或者日志流量并不大的情况下，可以使用跳板机的方案。对于大流量日志场景推荐您使用VPCEP。
 - VPCEP：即VPC终端节点，能够提供便捷、安全的通道用于与华为云当前区域LTS进行连接，使VPC中的资源无需弹性公网IP就能够访问终端节点服务。这种方式能够减少数据在公网上传输的风险，提高数据传输的安全性和效率。

前提条件

安装ICAgent前，请确保本地浏览器的时间、时区与主机的时间、时区一致。如果不一致，可能会导致日志上报出错。

使用限制

- Linux环境：ICAgent安装支持的Linux操作系统。
- Windows环境：仅支持在如下64位系统的Windows环境中安装ICAgent。

Windows Server 2016 R2 Datacenter
Windows Server 2016 R2 Standard
Windows Server 2016 Datacenter English
Windows Server 2016 R2 Standard English

Windows Server 2012 R2 Datacenter
Windows Server 2012 R2 Standard
Windows Server 2012 Datacenter English
Windows Server 2012 R2 Standard English

Windows Server 2008 R2 Enterprise
Windows Server 2008 R2 Standard
Windows Server 2008 Enterprise English
Windows Server 2008 R2 Standard English

Windows环境不支持在云日志服务的主机页面对ICAgent进行升级和卸载操作，只支持日志采集功能。如果需要使用新版本，请先卸载旧版本ICAgent，再安装新版本ICAgent即可。

安装场景说明

ICAgent有两种安装场景，请按照您的需要进行选择。

表 4-3 安装场景

场景	说明
首次安装	该服务器上未安装过ICAgent。
继承安装（Linux环境支持）	您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，对于没有安装ICAgent的其他多个服务器，您可以采用该安装方式。

首次安装

若服务器未安装过ICAgent，您可以参考如下操作在Linux环境或Windows环境首次安装ICAgent。

首次安装（Linux 环境）

步骤1 登录[云日志服务控制台](#)。

步骤2 左侧导航栏选择“主机管理 > 主机”，进入“主机”页面。

步骤3 单击“安装ICAgent”。

图 4-9 安装 ICAgent



步骤4 “主机类型”选择“区域外主机”。

步骤5 “安装系统”选择“Linux”。

步骤6 设置网络连通性方式。区域外主机将日志上报到当前区域LTS的网络通路，建议您优先选择专线，更稳定可靠。

- 若选择“公网”，直接从**步骤8**开始执行。
- 若选择“专线”，需要从**步骤7**开始执行。

步骤7 选择“连通LTS后端方式”，在专线连通场景下，默认区域外主机与当前区域LTS网络不通，安装在区域外主机上的ICAgent无法直接访问华为云管理面上报日志的网段。因此需要配置网络打通方案，使用跳板机或VPCEP的方式去连通LTS。

- 选择“连通LTS后端方式”为“VPCEP”时：
配置VPCEP域名。请您在华为云网络工程师的协助下，在区域外云上配置VPCEP的DNS域名解析规则，将VPCEP的域名解析到指定IP。配置完成后，在“安装ICAgent”页面根据界面提示复制命令：

```
ping {VPCEP域名}
```


选择一台待采集日志的主机，执行该命令。如果能ping通，表示网络配置连通。
跳转到**步骤8**开始执行。
- 选择“连通LTS后端方式”为“跳板机”时：
 - 创建Linux操作系统的ECS弹性云服务器作为跳板机。

登录弹性云服务器控制台，创建一个Linux操作系统的ECS。创建ECS的具体步骤，请参考[购买弹性云服务器](#)。如果您已有符合条件ECS，可直接使用作为跳板机，不需要再次创建。

- 推荐CentOS 6.5 64bit及其以上版本的镜像，最低规格为1vCPUs | 1GB，推荐规格为2vCPUs | 4GB。
 - 如果跳板机使用公网与区域外主机互通，那么需要开通EIP；如果跳板机使用云专线VPC对等连接方式与区域外主机互通，那么不需要开通EIP。
 - 跳板机的区域需要与LTS当前Region一致。
- b. 添加跳板机ECS使用的安全组规则，开放入方向对应端口，保证区域外主机到跳板机数据连通。
- i. 登录弹性云服务器控制台，查看弹性云服务器列表，找到作为跳板机的弹性云服务器。
 - ii. 单击跳板机名称进入ECS详情，单击安全组名称，进入安全组详情页。
 - iii. 在该安全组详情页，选择“入方向规则”页签，单击“添加规则”。请按照[表4-4](#)设置端口，更多参数可根据业务网络需求填写。详情请参考[添加安全组规则](#)。

表 4-4 安全组规则

策略	协议	端口	说明
允许	TCP	8149,8102,8923,30200,30201,80	ICAgent发送数据到跳板机的端口列表，保证非本区域主机到跳板机ECS的数据连通性。

- c. 返回弹性云服务器列表，找到[步骤7.a](#)中作为跳板机的弹性云服务器，可以获取私有IP及弹性公网IP（如跳板机开通EIP，此处会显示弹性公网IP）。

图 4-10 获取私有 IP



- d. 返回云日志服务控制台，在“安装ICAgent”页面中输入获取到的跳板机私有IP，生成跳板机转发命令。跳板机私有IP是指VPC内网IP。
- e. 在“安装ICAgent”页面单击“复制命令”，复制SSH Tunnel转发命令。
- ```
ssh -f -N -L {跳板机私有IP}:8149:{LTS上报IP}:8149 -L {跳板机私有IP}:8102:{LTS上报IP}:8102 -L {跳板机私有IP}:8923:{LTS上报IP}:8923 -L {跳板机私有IP}:30200:{LTS上报IP}:30200 -L {跳板机私有IP}:30201:{LTS上报IP}:30201 -L {跳板机私有IP}:80:icagent-{region}.{obs_domain}:80 {跳板机私有IP}
```
- f. 以root用户名密码登录跳板机，执行复制的SSH Tunnel转发命令。
- g. 执行netstat -lnp | grep ssh命令查看对应端口是否被侦听，如果返回结果如[图4-11](#)所示，说明TCP端口已开通。

图 4-11 TCP 端口验证结果

```
root@ecs-debc-qff-tiaobanji ~]# netstat -lntp | grep ssh
tcp 0 0 192.168.0.79:8102 0.0.0.0:* LISTEN 1994/sshd
tcp 0 0 192.168.0.79:80 0.0.0.0:* LISTEN 1994/sshd
tcp 0 0 192.168.0.79:8149 0.0.0.0:* LISTEN 1994/sshd
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1772/sshd
tcp 0 0 192.168.0.79:30200 0.0.0.0:* LISTEN 1994/sshd
tcp 0 0 192.168.0.79:30201 0.0.0.0:* LISTEN 1994/sshd
tcp 0 0 192.168.0.79:8923 0.0.0.0:* LISTEN 1994/sshd
tcp6 0 0 :::22 :::* LISTEN 1772/sshd
```

如果跳板机ECS掉电重启，请重新执行netstat -lntp | grep ssh命令。

- h. 在“安装ICAgent”页面，填写DC和跳板机连接IP。
- DC：自定义主机节点所属数据中心名称，便于分类查看主机。由数字、大小写字母、中划线、下划线组成，并且最多不超过64个字符。
  - 跳板机连接IP：如果跳板机使用EIP方式连通区域外主机，此处填写跳板机弹性IP；如果跳板机与区域外主机使用云专线VPC对等连接方式，此处填写跳板机VPC内网IP，即私有IP。弹性IP和私有IP的查看可参考[步骤7.c](#)。

**步骤8** 获取AK/SK。详细请参考[如何获取访问密钥AK/SK](#)。在“安装ICAgent”页面，复制ICAgent安装命令时需要替换AK/SK。

**步骤9** 使用PuTTY等远程登录工具，以root用户登录待安装ICAgent的区域外主机，执行ICAgent安装命令进行安装。

当显示“ICAgent install success”时，表示安装成功，ICAgent已安装在了/opt/oss/servicemgr/目录。安装成功后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看该服务器中ICAgent的状态。

如果安装失败，请卸载ICAgent后重新安装，如果还未安装成功，请联系技术支持。

当前网卡信息由节点上安装的ICAgent插件通过执行ifconfig命令获取。若网卡配置中没有inet信息，则不会上报IPv4地址，即在LTS主机页面的“主机IPv4”显示--；若未配置inet6信息，则不会上报IPv6地址，即在LTS主机页面的“主机IPv6”显示--。

----结束

首次安装（Windows 环境）

- 步骤1** 左侧导航栏选择“主机管理 > 主机”，进入“主机”页面。
- 步骤2** 单击“安装ICAgent”。
- 步骤3** “主机类型”选择“区域外主机”。
- 步骤4** “安装系统”选择“Windows”。

图 4-12 安装 ICAgent (Windows)

**安装ICAgent** ⓘ

主机类型 区域内主机 区域外主机

安装系统 Linux Windows ⓘ

网络连通性方式 公网 专线

云外K8S集群将日志上报到LTS的网络通路（公网或专线），建议您优先选择专线，更稳定可靠  
区域外主机安装ICAgent选择公网方式时，该局点必须支持公网上报日志

- 步骤一、通过点击 [ICProbeAgent.zip](#)，或者通过 [https://icagent-cn-north-4.obs.cn-north-4.myhuaweicloud.com/ICAgent\\_windows/ICProbeAgent.zip](https://icagent-cn-north-4.obs.cn-north-4.myhuaweicloud.com/ICAgent_windows/ICProbeAgent.zip)，下载ICAgent安装包。
- 步骤二、将ICAgent安装包存放到目录（如：C:\ICAgent）并解压。
- 步骤三、获取AK和SK，将在后续步骤中使用。[如何获取AK/SK?](#)
- 步骤四、执行安装命令。

执行的命令需要填写AK/SK：  
复制命令手动替换 {input\_your\_ak} 和 {input\_your\_sk}，填写值时不需要添加{}；

生成安装命令如下：[复制命令](#)

```
install.bat -ak {input_your_ak} -sk {input_your_sk} -projectid 39f073c9f-
huaweicloud.com -isapig true;
```

打开cmd窗口并进入ICAgent安装包的解压目录，执行上一步中生成的命令。当显示“Service icagent installed successfully”时，表示安装成功。安装成功后，在左侧导航栏中选择“主机管理”，查看ICAgent状态。

**步骤5** 设置“网络连通性方式”。区域外主机将日志上报到LTS的网络通路（公网或专线），建议您优先选择专线，更稳定可靠。

- 若选择“公网”，直接从**步骤7**开始执行。
- 若选择“专线”，需要从**步骤6**开始执行。

**步骤6** 选择“连通LTS后端方式”，在专线连通场景下，默认区域外主机与当前区域LTS网络不通，安装在区域外主机上的ICAgent无法直接访问华为云管理面上报日志的网段。因此需要配置网络打通方案，使用跳板机或VPCEP的方式去连通LTS。

- 选择“连通LTS后端方式”为“VPCEP”时：  
配置VPCEP域名。请您在华为云网络工程师的协助下，在区域外云上配置VPCEP的DNS域名解析规则，将VPCEP的域名解析到指定IP。配置完成后，在“安装ICAgent”页面根据界面提示复制命令：  
`ping {VPCEP域名}`  
选择一台待采集日志的主机，执行该命令。如果能ping通，表示网络配置连通。跳转到**7**开始执行。
- 选择“连通LTS后端方式”为“跳板机”时：
  - 创建Linux操作系统的ECS弹性云服务器作为跳板机。  
登录弹性云服务器控制台，创建一个Linux操作系统的ECS。创建ECS的具体步骤，请参考[购买弹性云服务器](#)。如果您已有符合条件ECS，可直接使用作为跳板机，不需要再次创建。
    - 推荐CentOS 6.5 64bit及其以上版本的镜像，最低规格为1vCPUs | 1GB，推荐规格为2vCPUs | 4GB。
    - 如果跳板机使用公网与区域外主机互通，那么需要开通EIP；如果跳板机使用云专线VPC对等连接方式与区域外主机互通，那么不需要开通EIP。
    - 跳板机的区域需要与LTS当前Region一致。
  - 添加跳板机ECS使用的安全组规则，开放入方向对应端口，保证区域外主机到跳板机数据连通。

- i. 登录弹性云服务器控制台，查看弹性云服务器列表，找到作为跳板机的弹性云服务器。
- ii. 单击跳板机名称进入ECS详情，单击安全组名称，进入安全组详情页。
- iii. 在该安全组详情页，选择“入方向规则”页签，单击“添加规则”。请按照表4-5设置端口，更多参数可根据业务网络需求填写。详情请参考[添加安全组规则](#)。

表 4-5 安全组规则

| 策略 | 协议  | 端口                            | 说明                                          |
|----|-----|-------------------------------|---------------------------------------------|
| 允许 | TCP | 8149,8102,8923,30200,30201,80 | ICAgent发送数据到跳板机的端口列表，保证非本区域主机到跳板机ECS的数据连通性。 |

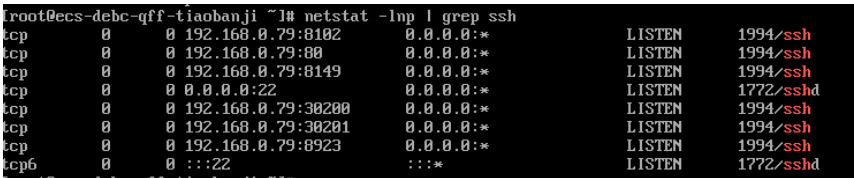
- c. 返回弹性云服务器列表，找到[步骤6.a](#)中作为跳板机的弹性云服务器，可以查看到私有IP及弹性公网IP（如跳板机开通EIP，此处会显示弹性公网IP）。

图 4-13 复制私有 IP



- d. 返回云日志服务控制台，在“安装ICAgent”页面中输入获取到的跳板机私有IP，生成跳板机转发命令。跳板机私有IP是指VPC内网IP。
- e. 在“安装ICAgent”页面单击“复制命令”，复制SSH Tunnel转发命令。  
ssh -f -N -L {跳板机私有IP}:8149:{LTS上报IP}:8149 -L {跳板机私有IP}:8102:{LTS上报IP}:8102 -L {跳板机私有IP}:8923:{LTS上报IP}:8923 -L {跳板机私有IP}:30200:{LTS上报IP}:30200 -L {跳板机私有IP}:30201:{LTS上报IP}:30201 -L {跳板机私有IP}:80:icagent-{region}-{obs\_domain}:80 {跳板机私有IP}
- f. 以root用户登录跳板机，执行复制的SSH Tunnel转发命令。
- g. 执行netstat -lnp | grep ssh命令查看对应端口是否被侦听，如果返回结果如[图4-14](#)所示，说明TCP端口已开通。

图 4-14 TCP 端口验证开通结果



如果跳板机ECS掉电重启，请重新执行netstat -lnp | grep ssh命令。

**步骤7** 通过“安装ICAgent”页面提示链接，下载ICAgent安装包。

**步骤8** 将ICAgent安装包存放到Windows主机目录（如：C:\ICAgent）并解压。

**步骤9** 获取AK/SK并保存好，复制ICAgent安装命令时需要替换AK/SK。详细请参考[如何获取访问密钥AK/SK](#)。



如果选择“连通LTS后端方式”为“跳板机”时，那么此处还要填写跳板机连接IP。

跳板机连接IP：如果跳板机使用EIP方式连通区域外主机，此处填写跳板机弹性IP；如果跳板机与区域外主机使用云专线VPC对等连接方式，此处填写跳板机VPC内网IP，即私有IP。弹性IP和私有IP的查看可参考[步骤6.c](#)。

**步骤10** 在“安装ICAgent”页面，单击“复制命令”，复制ICAgent安装命令。

**步骤11** 登录Windows主机，打开cmd窗口并进入ICAgent安装包的解压目录，执行ICAgent安装命令进行安装。

当显示“Service icagent installed successfully”时，表示安装成功。安装成功后，在云日志服务控制台左侧导航栏中选择“主机管理 > 主机”，查看ICAgent状态。

如果安装失败，请卸载ICAgent后重新安装，如果还未安装成功，请联系技术支持。

当前网卡信息由节点上安装的ICAgent插件通过执行ifconfig命令获取。若网卡配置中没有inet信息，则不会上报IPv4地址，即在LTS主机页面的“主机IPv4”显示--；若未配置inet6信息，则不会上报IPv6地址，即在LTS主机页面的“主机IPv6”显示--。

----结束

## 创建多台跳板机通过 ELB 进行负载均衡

单跳板机可能发生单点故障带来运维的不稳定性，此时可以创建多个跳板机并通过ELB负载均衡将流量分摊到不同的跳板机上，提高接入的可靠性。

**步骤1** 创建Linux操作系统的ECS弹性云服务器作为跳板机。

登录弹性云服务器控制台，创建一个Linux操作系统的ECS。创建ECS的具体步骤，请参考[购买弹性云服务器](#)。如果您已有符合条件ECS，可直接使用作为跳板机，不需要再次创建。

- 推荐CentOS 6.5 64bit及其以上版本的镜像，最低规格为1vCPUs | 1GB，推荐规格为2vCPUs | 4GB。
- 如果跳板机使用公网与区域外主机互通，那么需要开通EIP；如果跳板机使用云专线VPC对等连接方式与区域外主机互通，那么不需要开通EIP。
- 跳板机的区域需要与LTS当前Region一致。

**步骤2** 添加跳板机ECS使用的安全组规则，开放入方向对应端口，保证区域外主机到跳板机数据连通。

1. 登录弹性云服务器控制台，查看弹性云服务器列表，找到作为跳板机的弹性云服务器。
2. 单击跳板机名称进入ECS详情，单击安全组名称，进入安全组详情页。
3. 在该安全组详情页，选择“入方向规则”页签，单击“添加规则”。请按照[表 4-6](#)设置端口，更多参数可根据业务网络需求填写。详情请参考[添加安全组规则](#)。

表 4-6 安全组规则

| 策略 | 协议  | 端口                            | 说明                                          |
|----|-----|-------------------------------|---------------------------------------------|
| 允许 | TCP | 8149,8102,8923,30200,30201,80 | ICAgent发送数据到跳板机的端口列表，保证非本区域主机到跳板机ECS的数据连通性。 |

**步骤3** 返回弹性云服务器列表，找到**步骤1**中作为跳板机的弹性云服务器，可以查看到私有IP及弹性IP（如跳板机开通EIP，此处会显示弹性IP）。

图 4-15 复制私有 IP



**步骤4** 返回云日志服务控制台，在左侧导航栏选择“主机管理 > 主机”，单击“安装ICAgent”，在“安装ICAgent”页面中输入跳板机私有IP，生成跳板机转发命令。跳板机私有IP是指VPC内网IP。

**步骤5** 在“安装ICAgent”页面单击“复制命令”，复制SSH Tunnel转发命令。

```
ssh -f -N -L {跳板机私有IP}:8149:{LTS上报IP}:8149 -L {跳板机私有IP}:8102:{LTS上报IP}:8102 -L {跳板机私有IP}:8923:{LTS上报IP}:8923 -L {跳板机私有IP}:30200:{LTS上报IP}:30200 -L {跳板机私有IP}:30201:{LTS上报IP}:30201 -L {跳板机私有IP}:80:icagent-{region}-{obs_domain}:80 {跳板机私有IP}
```

**步骤6** 以root用户登录跳板机，执行复制的SSH Tunnel转发命令。

**步骤7** 重复以上步骤创建多个跳板机，并且将多个跳板机放入同一个VPC中。即在创建弹性云服务器，进行网络配置时，选择同一个虚拟私有云。

**步骤8** 登录弹性负载均衡控制台，创建弹性负载均衡ELB。具体请参见[创建弹性负载均衡](#)，在创建时需注意以下几点。

1. 创建ELB，在网络配置时，选择与跳板机ECS相同的VPC。
2. 新建弹性公网IP，ELB的弹性公网IP将作为跳板机连接IP。
3. 带宽根据业务量申请，并进行适配。

**步骤9** 分别为TCP的端口30200、30201、8149、8923、8102、80添加监听器。具体请参见[添加TCP监听器](#)。

**步骤10** 创建一个后端服务器组，并将所有的跳板机放置在同一个后端服务器组，具体请参见[添加后端云服务器](#)。

**步骤11** 返回云日志服务控制台，在“安装ICAgent”页面，将ELB的弹性公网IP填写到“跳板机连接IP”中，复制安装命令，在对应区域外主机上执行即可。

----结束

## 继承安装

如果您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，对于没有安装ICAgent的其他多个服务器，您可以采用继承安装或继承批量安装的方式。

## 继承安装（Linux 环境）

您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent的安装包ICProbeAgent.tar.gz，对于没有安装ICAgent的服务器，可以通过该方式对服务器进行一键式继承安装。



继承安装的服务器需同属一个VPC下，并在同一个网段中。

**步骤1** 在已安装ICAgent的服务器上执行如下命令，其中x.x.x.x表示待安装ICAgent服务器的IP地址。

```
bash /opt/oss/servicemgr/ICAgent/bin/remotelInstall/remote_install.sh -ip x.x.x.x
```

**步骤2** 根据提示输入待安装ICAgent的服务器root用户密码。

- 如果已安装ICAgent的服务器安装过expect工具，执行上述命令后，即可完成安装。如果已安装ICAgent的服务器未安装expect工具，请根据提示输入密码，进行安装。
- 请确保已安装ICAgent的服务器可以使用root用户执行SSH、SCP命令，来与待安装ICAgent的服务器进行远端通信。
- 如果安装失败，请卸载ICAgent后重新安装，如果还未安装成功，请联系技术支持。

**步骤3** 当显示“Install ICAgent success”时，表示安装成功，ICAgent已安装在了“/opt/oss/servicemgr/”目录。安装成功后，在云日志服务左侧导航栏中选择主机管理 > 主机”，查看该服务器ICAgent的状态。

----结束

## 继承批量安装（Linux 环境）

您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent的安装包 **ICProbeAgent.tar.gz**，对于没有安装ICAgent的服务器，可以通过该方式对服务器进行一键式继承批量安装。批量安装的服务器需同属一个VPC下，并在同一个网段中。

批量安装功能依赖python3.\*版本，如果安装时提示找不到python请在服务器上安装python版本后重试。

### 前提条件

已收集需要安装Agent的所有服务器的IP地址、root密码，按照iplist.cfg格式整理好，并上传到已安装过ICAgent机器的“/opt/ICAgent/”目录下。iplist.cfg格式示例如下所示，IP地址与root密码之间用空格隔开：

192.168.0.109 密码（请根据实际情况填写）

192.168.0.39 密码（请根据实际情况填写）

- iplist.cfg中包含您的敏感信息，建议您使用完之后进行清理。
- 如果所有服务器的密码一致，在iplist.cfg中只需列出IP，不用列出密码。然后执行时输入此密码即可；如果某个IP密码与其他不一致，则需在此IP后填写其密码。

### 操作步骤

**步骤1** 在已安装ICAgent的服务器上执行如下命令。

```
bash /opt/oss/servicemgr/ICAgent/bin/remotelInstall/remote_install.sh -batchModeConfig /opt/ICAgent/iplist.cfg
```

根据脚本提示输入待安装机器的root用户默认密码，如果所有IP的密码在iplist.cfg中已有配置，则直接输入回车键跳过即可，否则请输入默认密码。

请耐心等待，当提示Install ICAgent success时，则表示配置文件中的所有主机安装操作已完成。

**步骤2** 安装完成后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看主机的安装状态，详细请参考[查看ICAgent状态](#)。

----结束

## 相关文档

如果在安装ICAgent时遇到问题，您可以参考[主机管理常见问题](#)获取解决方案。

## 4.2.4 管理 ICAgent

LTS支持通过ICAgent采集方式上报日志。ICAgent是云日志服务进行日志采集的工具，运行在需要采集日志的主机中。本文介绍如何在目标服务器中安装ICAgent、升级ICAgent、卸载ICAgent、查看ICAgent状态、查看ICAgent版本说明。

根据目标服务器类型可分为以下两类，请根据您的服务器运行环境选择对应的方案，不同场景的使用流程及配置要求存在差异。

- **主机场景**：适用于物理服务器、云服务器ECS等传统计算环境。根据主机所在位置分为区域内主机和区域外主机。
- **容器场景**：基于Kubernetes容器化平台部署的业务场景。当前支持云容器引擎CCE、自建K8s、ServiceStage容器化部署。

## 安装 ICAgent 插件

LTS支持对主机场景或容器场景下的目标服务器安装ICAgent。

## 主机场景安装 ICAgent 插件

根据主机所在位置分为区域内主机和区域外主机，详细操作请参考[安装ICAgent（区域内主机）](#)和[安装ICAgent（区域外主机）](#)。

## 容器场景安装 ICAgent 插件

- CCE集群安装ICAgent  
当前只支持在CCE Standard集群和Turbo集群上安装ICAgent。不支持在CCE Autopilot集群和CCE敏捷版集群上安装ICAgent。
  - 在左侧导航栏选择“主机管理 > 主机”，进入“主机”页面。
  - 选择“CCE集群”，在搜索框中选择待升级ICAgent的集群，查看ICAgent状态显示未安装，单击“升级ICAgent”。

图 4-16 CCE 集群升级 ICAgent



- 如果在LTS主机页面看不到CCE集群，需要AOM2.0授权给LTS后，就可以在LTS页面看到CCE集群。进入AOM2.0控制台后，请参考[云服务授权](#)授予云日志服务（LTS）、云容器引擎（CCE）等云服务数据的访问权限。

- 未创建CCE集群时，采集容器标准输出到AOM的开关为置灰状态。
  - 当ICAgent版本号为5.12.133及以上时，支持关闭采集容器标准输出到AOM1.0的开关功能。
  - 首次创建的CCE集群，默认集群下的主机已安装了ICAgent且上报日志到LTS，采集容器标准输出到AOM的开关处于关闭状态；如需将日志上报至LTS则执行升级ICAgent操作时，关闭采集容器标准输出到AOM的开关。建议使用[云容器引擎CCE应用日志接入LTS](#)时直接采集容器标准输出到LTS，不推荐采集到AOM。
  - CCE集群ID（ClusterID）：每个集群为固定的ID。
  - 升级ICAgent时，LTS将为您CCE集群创建对应的日志组和主机组。且该日志组和主机组的名称为k8s-log-{ClusterID}。您可以创建接入配置时将当前CCE集群的日志接入到该日志组。详细请参考[云容器引擎CCE应用日志接入LTS](#)。
  - 当集群里的主机未安装ICAgent或ICAgent版本过低时，单击“升级ICAgent”操作，可对该集群里的所有主机安装ICAgent。
- c. 在“升级ICAgent”对话框中，单击“确定”开始升级ICAgent。
- 升级ICAgent预计需要1分钟左右，请耐心等待。待ICAgent的状态由“升级中”变为“运行”时，表示升级成功。

图 4-17 升级 ICAgent



- 自建kubernetes集群安装ICAgent的方法请参考[安装日志采集组件](#)。  
通过自建k8s方式安装ICAgent时，k8s集群节点数量不超过50个且集群中所有pod数量不超过400个。

## 升级 ICAgent 插件

为了更好的采集体验，LTS会不断更新ICAgent版本。当系统提示您有新的ICAgent版本时，您可以参考以下内容进行升级。

## 主机场景升级 ICAgent 插件

LTS会自动检测主机安装的ICAgent插件版本。如果检测到您的主机中后期不维护的ICAgent插件版本时，当您进入LTS控制台后，界面会弹出“ICAgent版本太低”的提示框。

- 您可以单击“**授权自动升级**”授权LTS为您在业务低峰期自动升级ICAgent插件。
- 如果您不想要自动升级ICAgent插件，也可以单击“**前往手动升级**”参考如下操作步骤，前往主机管理页面升级ICAgent。
- 如果您**暂不需要升级ICAgent插件**，可勾选“不再提示”前的复选框，取消提示信息。

如果需要升级Windows环境中的ICAgent，请先卸载旧版本ICAgent，再安装新版本ICAgent即可。

1. 登录管理控制台，选择“管理与部署 > 云日志服务 LTS”，进入“日志管理”页面。
2. 左侧导航栏选择“主机管理 > 主机”，进入“主机”页面。
3. 选择“区域内主机”或“区域外主机”，当系统提示您有新的ICAgent版本时，在主机列表中选中一个或多个待升级ICAgent前的复选框，单击“升级ICAgent”。
4. 在“升级ICAgent”对话框中，单击“确定”开始升级ICAgent。

升级ICAgent预计需要1分钟左右，请耐心等待。待ICAgent的状态由“升级中”变为“运行”时，表示升级成功。

如果升级后，界面显示ICAgent状态异常或者其它升级失败场景，请直接登录节点使用安装命令重新安装ICAgent即可（覆盖式安装，无需卸载操作）。详细操作请参考[安装ICAgent（区域内主机）](#)和[安装ICAgent（区域外主机）](#)

## 容器场景升级 ICAgent 插件

CCE集群升级ICAgent的操作方法与安装场景相同，请参考[CCE集群安装ICAgent](#)。

## 卸载 ICAgent



### 警告

服务器上的ICAgent被卸载后，会影响该服务器的日志采集能力，请谨慎操作！

卸载ICAgent不会删除对应的安装文件，请您根据实际情况自行删除。

### 主机场景

云日志服务主机管理界面，仅支持卸载安装在Linux环境中的ICAgent，如果需要卸载安装Windows环境中的ICAgent，请在ICAgent安装包解压目录下，双击执行“*ICAgent安装包解压目录*\ICProbeAgent\bin\manual\win\uninstall.bat”脚本，当显示“ICAgent uninstall success”时，表示卸载成功。

卸载方式，您可以按照需要进行选择：（以下提供的都是Linux系统的卸载方式）

- 通过界面卸载：此操作适用于正常安装ICAgent后需卸载的场景。
  - a. 左侧导航栏选择“主机管理 > 主机”，进入“主机”页面。
  - b. 勾选一个或多个待卸载ICAgent的服务器的复选框，单击“卸载ICAgent”。

- c. 在“卸载ICAgent”对话框中单击“确定”。  
ICAgent开始卸载，卸载ICAgent预计需要1分钟左右，请耐心等待。  
卸载完成后在主机列表中ICAgent状态显示“未安装”。  
通过界面卸载ICAgent后如果需要再次安装，请等待5分钟后执行安装操作，否则可能出现被再次自动卸载的情况。
- 登录服务器卸载：此操作适用于未成功安装ICAgent需卸载重装的场景。
  - a. 以root用户登录需卸载ICAgent的服务器。
  - b. 执行如下命令卸载ICAgent。

```
bash /opt/oss/servicemgr/ICAgent/bin/manual/uninstall.sh
```

  
当显示“ICAgent uninstall success”时，表示卸载成功。
- 远程卸载：此操作适用于正常安装ICAgent后需远程卸载的场景。  
除了上述登录服务器上执行uninstall.sh命令卸载ICAgent的方式，还可以对服务器进行远程卸载。远程卸载的服务器需同属一个VPC下，并在同一个网段中。
  - a. 在已安装ICAgent的服务器上执行如下命令，其中x.x.x.x表示待卸载ICAgent的服务器的IP地址。

```
bash /opt/oss/servicemgr/ICAgent/bin/remoteUninstall/remote_uninstall.sh -ip x.x.x.x
```
  - b. 根据提示输入待卸载ICAgent的服务器root用户密码。
    - 如果已安装ICAgent的服务器安装过expect工具，执行上述命令后，即可完成卸载。如果已安装ICAgent的服务器未安装expect工具，请根据提示输入密码，进行卸载。
    - 请确保已安装ICAgent的服务器可以使用root用户执行SSH、SCP命令，来与待卸载ICAgent的服务器进行远端通信。
    - 当显示“Uninstall ICAgent success”时，表示卸载成功。
- 批量卸载：此操作适用于正常安装ICAgent后需批量卸载的场景。  
当您已有服务器安装过ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent安装包ICProbeAgent.tar.gz，通过该方式可对多个服务器进行一键式继承批量卸载。

---

 **注意**

批量卸载的服务器需同属一个VPC下，并在同一个网段中。

---

### 前提条件

已收集需要卸载ICAgent的所有服务器的IP地址、密码，按照iplist.cfg格式整理好，并上传到已安装过ICAgent机器的/opt/ICAgent/目录下。iplist.cfg格式示例如下所示，IP地址与密码之间用空格隔开：

192.168.0.109 密码（请根据实际情况填写）

192.168.0.39 密码（请根据实际情况填写）

- iplist.cfg中包含您的敏感信息，建议您使用完之后进行清理。
- 如果所有服务器的密码一致，iplist.cfg中只需列出IP地址，无需填写密码，在执行时输入此密码即可；如果某个IP密码与其他不一致，则需在此IP地址后填写其密码。

### 操作步骤

- a. 在已安装ICAgent的服务器上执行如下命令。
- ```
bash /opt/oss/servicemgr/ICAgent/bin/remoteUninstall/remote_uninstall.sh - batchModeConfig /opt/ICAgent/iplist.cfg
```
- 根据脚本提示输入待卸载机器的root用户默认密码，如果所有IP地址的密码在iplist.cfg中已有配置，则直接输入回车键跳过即可，否则请输入默认密码。
- 请耐心等待，当提示Uninstall ICAgent success时，则表示配置文件中所有服务器的卸载操作已完成。
- 但是用来执行卸载操作的服务器上的ICAgent不会被卸载，您需要单独卸载该服务器的ICAgent。
- b. 卸载完成后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看该服务器的ICAgent状态。

容器场景

请参考[管理CCE集群的ICAgent插件](#)。

查看 ICAgent 状态

左侧导航栏选择“主机管理 > 主机”，查看目标主机的ICAgent状态。详细请参见[表 4-7](#)。

表 4-7 ICAgent 状态

状态	说明
运行	该服务器的ICAgent运行正常。
未安装	该服务器未安装ICAgent。
安装中	正在为该主机安装ICAgent。安装ICAgent预计需要1分钟左右，请耐心等待。
安装失败	该主机的ICAgent安装失败。
升级中	正在升级该服务器的ICAgent。升级ICAgent预计需要1分钟左右，请耐心等待。
升级失败	该服务器的ICAgent升级失败。
离线	输入的AK/SK错误导致该主机的ICAgent功能异常。请获取正确的AK/SK后重新安装。
异常	该主机ICAgent功能异常，请联系技术支持。
卸载中	正在卸载该主机。卸载ICAgent预计需要1分钟左右，请耐心等待。
鉴权错误	安装该主机时配置的参数问题导致无法正常鉴权。

查看 ICAgent 版本说明

左侧导航栏选择“主机管理 > 主机”，查看目标主机的ICAgent版本，建议您尽快升级到最新版本。详细请参见[表4-8](#)。

表 4-8 ICAgent 版本说明

版本号	说明	发布时间
7.3.6	• 优化标准输出采集逻辑。 • 优化ECS安装ICAgent的安装脚本。 • 优化ICAgent生成AgentId的逻辑。 • 解决标准输出采集任务配置多行时，文件最后一行日志不上报的问题。	2025-07-25
7.3.3	• 优化容器日志采集正则校验的逻辑。 • 解决容器日志采集路径配置为挂载目录上级时无法采集的问题。	2025-06-26
7.3.2	解决在自建k8s场景下无法在AOM1.0页面通过集群名称查找ICAgent上报系统日志的问题。	2025-06-09
7.3.1	• LTS配置ICAgent结构化日志过滤的白名单里支持逻辑关系选择。 • ICAgent采集上报支持base64编解码。 • ICAgent调用ECS openstack支持V2版本接口 • 自建k8s没有开启10255端口时，容器日志和标准输出日志也支持采集。	2025-05-26
7.2.6	ECS重装或卸载ICAgent时，优化清理AgentId文件的逻辑，解决私有镜像预置ICAgent后导致的AgentId重复问题。	2025-05-14
7.2.4	调整游标目录：游标文件名取消日志组日志流，同时将游标文件放入到日志组日志流组成的目录中。	2025-04-25
7.2.2	• 解决大量日志采集场景下，采集协程偶现阻塞导致ICAgent不采集标准输出日志的问题。 • 解决上报k8s事件时reason字段未上报的问题。	2025-03-13
7.2.1	• 结构化单行正则以及多行正则增加行号处理。 • 优化日志发送时label的获取逻辑。 • ntp指标支持配置多台ntp服务器。 • 适配容器npu指标采集，使用podIP进行指标获取。	2025-02-28
7.1.22	优化代码，减少获取pod信息时内存消耗。	2025-01-23
7.1.19	• 优化AgentId生成逻辑。 • 修改绕接文件过期时间为2小时。 • 优化日志打印。	2025-01-13
7.1.17	优化日志采集流程，调整发送块大小。	2025-01-03
7.1.14	• 优化日志采集流程。 • 日志采集时，持久化保存签名信息。	2024-12-25

版本号	说明	发布时间
7.1.12	解决在某些场景下，Go版本引起的定时器泄漏导致CPU升高的问题。	2024-12-15
7.1.6	- 支持日志上报镜像名称。 - 优化采集日志流程。 - 升级ICAgent基础镜像版本。	2024-11-21
7.1.5	解决用户project下所有配置删除或关闭后，仍然继续采集日志的问题。	2024-11-21
7.1.3	优化发送日志时httpClient组件性能。	2024-11-02
7.1.1	优化ICAgent架构，统一AOM1.0和AOM2.0的Agent。	2024-10-26
5.12.233	- 优化容器日志结构化性能。 - 解决CCE场景错误诊断偶现上报失败的问题。 - 解决虚机场景不活跃文件采集失效的问题。	2024-10-17
5.12.232	- 解析标准输出日志用到的json库替换为sonic，降低CPU使用率。 - LTS发送日志出现超时，解决超时问题。	2024-10-14
5.12.231	- 解决以下条件满足的情况下ICAgent重启的问题：标准输出采集到LTS、LTS不配置日志接入规则、CCE创建新容器并打印标准输出日志。 - 解决游标文件中hisfile变成目录的问题。 - 解决增量采集开关不生效的问题。	2024-10-11
5.12.230	- 解决游标定时刷新功能不生效的问题。 - 查找不到绕接文件的情况下，解决base文件绕接后无法重置游标的问题。 - 解决文件签名导致cpu高的问题。	2024-10-09
5.12.224	在ECS升级场景下，若“.bashrc文件中export HISTSIZE=0”大于1条，则清除“.bashrc文件中的export HISTSIZE=0”。	2024-09-27
5.12.218	- ICAgent上报日志支持GBK编码。 - ICAgent上报日志允许文件多次采集。	2024-09-26
5.12.185	- 解决虚机日志配置中黑名单路径不生效问题。 - 优化containerd标准输出日志采集的问题。	2024-05-20
5.12.184	- 解决容器日志采集功能中无法排除绕接文件的问题。 - 节点日志采集功能并发采集协程调整为32个。	2024-05-16

版本号	说明	发布时间
5.12.183	优化containerd节点采集容器标准输出绕接文件的问题。	2024-05-11
5.12.182	解决syslog开关问题。	2024-04-28
5.12.181	- 解决自建k8s icagent认证失败问题。 - 解决日志截断问题。 - 解决日志速率很大的情况下，查找不到绕接文件导致文件漏采的问题。	2024-04-25
5.12.177	解决绕接死循环问题。	2024-03-28
5.12.176	- zip流式解析优化：检查转储文件是否结束。 - 限制podlb每个主机最大连接数。	2024-03-18
5.12.175	解决了结构化日志采集性能瓶颈问题。	2024-03-13
5.12.172	优化支持的绕接方式。	2024-02-28
5.12.171	解决Docker节点标准输出日志Json解析问题（没有去掉转义字符）。	2024-01-31
5.12.170	- 主机日志，容器日志，标准输出日志支持增量采集。 - 解决主机gpu指标挂断问题。	2024-01-29
5.12.166	- 解决标准输出日志采集插件占用内存高问题。 - 解决虚机日志采集插件重复采集绕接文件问题。 - 游标文件中添加日志组和日志流信息。	2023-12-27
5.12.165	从配置文件获取初始agentID，如果不符合校验要求则使用随机生成的uuid。	2023-12-21
5.12.163	支持UniAgent插件化安装ICAgent。	2023-12-13
5.12.159	- 解决标准输出日志采集协程泄露问题。 - 解决标准输出日志采集到AOM后，不支持采集标准输出绕接日志的问题。	2023-11-27
5.12.158	解决关闭指标开关后容器指标内存泄露导致ICAgent重启的问题。	2023-11-08
5.12.157	- CCE接入LTS的容器日志采集：支持Docker驱动Devicemapper。 - 解决虚机日志量大（转储快）ICAgent内存暴涨导致重启问题。	2023-11-06
5.12.156	解决从OBS拉取安装包问题，将http协议改成https。	2023-11-01
5.12.154	支持结构化功能。	2023-10-31

版本号	说明	发布时间
5.12.150	- 解决集群name和集群id not-set问题。 - 支持CCE集群1.27版本。	2023-10-17
5.12.149	支持挂载绕接功能。	2023-10-12
5.12.148	修复gpu多卡场景，解决cpu高的问题。	2023-08-30
5.12.147	修复日志转储无法重启、主机gpu指标适配。	2023-08-17
5.12.142	支持CCE集群1.25及以上版本的容器gpu指标采集。	2023-06-13
5.12.139	解决上报LTS日志出现大量TIME_WAIT状态的TCP连接问题。	2023-04-25
5.12.135	- 解决CPU使用率为0的问题。 - 解决CCE1.23版本集群containerd节点容器网络指标缺失问题。 - 支持采集EulerOS 2.5系统的磁盘分区指标。	2023-02-08
5.12.133	容器的标准输出日志支持多行采集。	2022-12-17
5.12.130	支持将CCE日志直接接入LTS。	2022-11-04
5.12.120	- 增加进程的最大句柄数指标。 - 支持LTS的podlb域名的切换能力。	2022-08-28
5.12.111	新增线程指标、修复“获取lvs磁盘分区指标失败”问题。	2022-06-09
5.12.100	- 上报内存指标增加内存workingset使用量、内存workingset使用率。 - 容器采集支持通过标签区分stderr.log和stdout.log。 - 容器上报增加Pod_ip的tag。 **配置匹配当前目录文件。	2022-01-15
5.12.98	增加LTS日志黑名单功能，更改容器指标来源为working_set。	2021-09-29
5.12.96	新增云资源发现类型。	2021-09-22
5.12.90	更新gpu指标来源。	2021-07-15
5.12.87	新增磁盘支持类型。	2021-03-30
5.12.75	适配安全容器场景。	2021-03-09

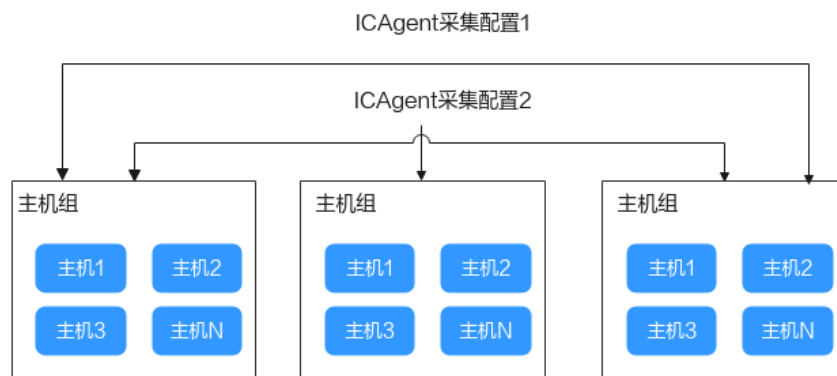
4.2.5 管理 LTS 主机组

主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。云日志服务支持通过一个接入配置来采集多台主机上的日志，您可以将这

些主机加入到同一个主机组，并将该主机组关联至对应的接入配置中，配置日志接入时以主机组为单位下发采集配置，方便您对多台主机日志进行采集。请参考图4-18。

- 当用户扩容主机时，只需在主机组中添加主机，该主机会自动继承关联的日志路径，无需为每台主机重复配置路径。
- 当用户修改多个主机采集路径时，只需修改对应的主机组关联的路径，无需为每台主机重复配置路径。

图 4-18 主机组



支持创建IP类型或自定义标识类型主机组。

- **创建主机组（IP地址）**：创建主机组时，直接勾选同类型主机加入该主机组即可，操作简单。
- **创建主机组（自定义标识）**：创建主机组时，需要在主机组和主机上分别创建标识，若标识存在交集，则会自动将该主机加入主机组内，操作方法复杂。

若有以下场景，可以选择使用自定义标识主机组：

- VPC等自定义网络环境中，可能出现不同服务器IP地址冲突的问题，导致日志服务无法管理ICAgent。使用自定义标识可以避免此类情况的发生。
- 多台服务器通过同一个自定义标识实现主机组弹性伸缩。您只需为新增的服务器配置相同的自定义标识，日志服务可自动识别，并将其添加至主机组中。

创建主机组

LTS支持创建IP类型或自定义标识类型的主机组。

创建主机组（IP 地址）

步骤1 登录管理控制台，选择“管理与部署 > 云日志服务 LTS”，进入“日志管理”页面。

步骤2 左侧导航栏选择“主机管理 > 主机组”，进入“主机组”页面。

步骤3 单击“新建主机组”。

步骤4 在弹出的新建主机组页面，输入“主机组名称”，主机组类型选择“IP”，主机类型选择“Linux主机”或“Windows主机”。自定义填写备注信息，字符长度0-1024个字符。

图 4-19 创建 IP 地址主机组

新建主机组

* 主机组名称

只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾

* 主机组类型

IP

自定义标识

* 主机类型

Linux主机

Windows主机

备注

0/1024

添加主机

主机列表

⚠ 请勾选已安装ICAgent的主机加入主机组，否则日志可能无法正常采集

➤ 安装ICAgent

🛑 卸载ICAgent

批量搜索主机IP

查看已选 (0)

🔍

🔍 点击此处添加筛选条件

☐ 主机名称

主机IPv4

主机IPv6

企业项目

ICAg...

ICAg...

更新...

暂无主机

主机接入需要先配置安装ICAgent，如果你的主机没有出现在这里，请前往安装ICAgent。

➤ 安装ICAgent

- 步骤5** 在列表中勾选需要加入该主机组的主机，单击“确定”，完成主机组的创建。
- 可以通过主机名称、企业项目、IPv4或IPv6对主机列表进行过滤，也可以单击“批量搜索主机IP”，并在弹出的搜索框中输入多个主机IP，进行批量搜索。
 - 当列表中没有所需主机时，单击“安装ICAgent”，在弹出的页面安装指引完成主机安装，具体操作可参见[安装ICAgent（区域内主机）](#)或[安装ICAgent（区域外主机）](#)。
- 结束

创建主机组（自定义标识）

选择自定义标识主机组类型时，需要提前规划好需要添加标识的主机，且主机已安装ICAgent。

- 步骤1** 单击“新建主机组”。
- 步骤2** 在弹出的新建主机组页面，输入“主机组名称”，主机组类型选择“自定义标识”，主机类型选择“Linux主机”或“Windows主机”。自定义填写备注信息，字符长度0-1024个字符。

图 4-20 创建自定义标识主机组

新建主机组

* 主机组名称

只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾

* 主机组类型

IP

自定义标识

* 主机类型

Linux主机

Windows主机

备注

0/1024

* 自定义标识

1. 仅ICAgent 5.12.117 及以上版本支持。升级指引

2. 自定义标识填写指引

值	操作
请输入 删除	

+ 添加

- 步骤3 输入自定义标识信息，若需要添加更多标识，单击“添加”。
最多可添加10个自定义标识。
- 步骤4 单击“确定”，主机组自定义标识创建完成后，请参考5将主机加入自定义标识主机组。
- 步骤5 执行以下操作创建custom_tag文件，用来保存主机标签。
 - 登录主机执行如下命令查看是否有/opt/cloud目录。

```
cd /opt/cloud
```

 - 若提示没有/opt/cloud目录，执行如下命令创建/opt/cloud目录。

```
mkdir /opt/cloud/
```
 - 若有/opt/cloud目录，在cloud目录下，执行如下命令创建lts目录。

```
mkdir lts
```
 - 继续执行如下命令，修改lts目录权限。

```
chmod 750 lts
```
 - 在lts目录下执行如下命令，创建custom_tag文件 。

```
cd lts  
touch custom_tag
```
 - 继续执行如下命令，修改custom_tag权限并打开该文件。

```
chmod 640 custom_tag;vi custom_tag
```
 - 按i进入insert模式，键入自定义标识后，按ESC键，“:wq!”保存退出即可。

```
:wq!
```
 - 支持以下两种方式将主机加入到自定义标识主机组：

表 4-9 添加方式

类型	方式1（推荐使用）	方式2
Linux主机	在主机里 <code>/opt/cloud/lts</code> 目录下的 <code>custom_tag</code> 文件中，查看该主机的标识，然后将该主机的标识，添加为主机组自定义标识，就可以将主机加入到该主机组下。例如：在主机里 <code>/opt/cloud/lts</code> 目录下的 <code>custom_tag</code> 文件中，查看该主机的标识为test1，创建主机组的自定义标识为test1，即将该主机加入到主机组下。	- 在主机里<code>/opt/cloud/lts</code>目录下的<code>custom_tag</code>文件中，添加主机组自定义标识，可以将主机加入到该主机组下。例如：主机组的自定义标识为test，则在<code>custom_tag</code>文件中填写test，就可以将主机加入到该主机组下。 - 当添加了多个自定义标识时，在主机里<code>/opt/cloud/lts</code>目录下的<code>custom_tag</code>文件中，任意填写一个自定义标识，就可以将主机加入到该主机组下。
Windows主机	在主机里 <code>C:\opt\cloud\lts</code> 目录下的 <code>custom_tag</code> 文件中，查看该主机的标识，然后将该主机的标识，添加为主机组自定义标识，就可以将主机加入到该主机组下。例如：在主机里 <code>C:\opt\cloud\lts</code> 目录下的 <code>custom_tag</code> 文件中，查看该主机的标识为test1，创建主机组的自定义标识为test1，即将该主机加入到主机组下。	- 在主机里<code>C:\opt\cloud\lts</code>目录下的<code>custom_tag</code>文件中，添加主机组自定义标识，可以将主机加入到该主机组下。例如：主机组的自定义标识为test，则在<code>custom_tag</code>文件中填写test，就可以将主机加入到该主机组下。 - 当添加了多个自定义标识时，在主机里<code>C:\opt\cloud\lts</code>目录下的<code>custom_tag</code>文件中，任意填写一个自定义标识，就可以将主机加入到该主机组下。

----结束

修改主机组

对于已创建的主机组可以对其名称进行修改，也可以对主机组进行添加主机、移除主机或者关联接入配置，具体操作参考如下[表4-10](#)。

表 4-10 操作列表

操作	具体步骤
修改主机组名称	1. 在“主机组”页面，单击待修改的主机组所在行的操作列“修改”。 2. 在弹出的修改主机组页面，修改主机组名称、自定义标识等信息。 3. 单击“确定”，完成主机名称修改。
添加主机	方式一： 1. 在主机组列表，找到主机组类型为IP的主机组，单击主机组名称前的。 2. 在主机页签，单击“添加主机”。 3. 在弹出的添加主机页面，主机列表中显示该主机组所选主机类型下所有未选主机，选择需要加入该主机组的主机。 • 可以通过主机名称、企业项目、IPv4或IPv6对主机列表进行过滤，也可以单击“批量搜索主机IP”，并在弹出的搜索框中输入多个主机IP，进行批量搜索。 • 当列表中没有所需主机时，单击“安装ICAgent”，在弹出的页面安装指引完成主机安装，具体操作可参见安装ICAgent。 4. 单击“确定”。 方式二： 1. 选择“主机管理 > 主机”页面。 2. 在主机列表中勾选需要添加的主机，单击“添加到主机组”。 3. 在弹出的添加到主机组页面，勾选目标主机组。 4. 单击“确定”，完成主机的添加。
移除主机	1. 在主机组列表，单击待修改的主机组所在行前的。 2. 在主机页签，单击待移除主机所在行操作列的“移除”。 3. 在弹出的移除主机页面，单击“确定”，将该主机移除。自定义标识主机组下的主机不支持该操作。
取消部署	1. 在主机组列表，单击待修改的主机组所在行前的。 2. 在主机页签，单击待移除主机所在行操作列的“取消部署”。 3. 在弹出的取消部署页面，单击“确定”，将该主机ICAgent卸载并移除。 • 自定义标识主机组下的主机不支持该操作。 • 主机取消部署后，其他主机组下的该主机也会被移除。
批量移除	1. 在主机组列表，单击待修改的主机组所在行前的。 2. 在主机页签，勾选待移除的主机，单击“批量移除”。 3. 单击“确定”。

操作	具体步骤
新增关联配置	1. 在主机组列表，单击待修改的主机组所在行前的 。 2. 默认显示主机页签，单击“相关日志接入规则”。 3. 单击“新增关联配置”。 4. 在弹出的新增关联配置页面，勾选需要关联的接入配置。 5. 单击“确定”，配置完成后会将所选的接入配置显示在列表中。
解除关联	1. 在“相关日志接入规则”页签，单击待解除配置所在行操作列的“解除关联”。 2. 单击“确定”，解除该主机组与该接入配置的关联。
批量解除关联	1. 在主机组列表，单击待修改的主机组所在行前的 。 2. 默认显示主机页签，在“相关日志接入规则”页签，勾选待解除的配置，单击“批量解除关联”。 3. 单击“确定”，解除该主机组与所勾选的接入配置的关联。
复制主机组ID	鼠标悬浮在主机组名称上，支持复制主机组ID。
导出主机信息	1. 在“主机”页面的区域内主机、CCE集群或区域外主机下方，勾选需要导出的主机。 2. 单击“导出”，选择导出全部数据到XLSX或导出已选中数据到XLSX，即可将主机信息导出到本地进行查看。

添加主机组标签

支持对已创建的主机组添加标签，每个标签由“标签键”和“标签值”组成，用于快速识别、检索和管理主机组。添加、修改、删除标签仅对当前主机组有效。

- 步骤1 左侧导航栏选择“主机管理 > 主机组”，进入“主机组”页面。
- 步骤2 单击目标主机组操作列的“标签管理”。
- 步骤3 在弹出的编辑标签页面，单击“添加”，填写标签键和标签值。如需添加多个标签可重复该步骤，最多支持添加20个标签。

图 4-21 主机组编辑标签



标签键key限制条件：

- 标签键可以包含任意语种的字母、数字和空格，以及_:=+~@字符，但首尾不能包含空格，且不能以_sys_开头。
- 标签键长度不能超过128个字符。
- 标签键名称不可重复。

标签值value限制条件：

- 标签值可以包含任意语种的字母、数字和空格，以及_:=+~@字符。
- 标签值长度不能超过255个字符。

标签策略：

若您的组织已经设定云日志服务的相关标签策略，则需按照标签策略规则为日志组、日志流、日志接入、主机组添加标签。标签如果不符合标签策略的规则，则可能会导致日志组、日志流、日志接入、主机组创建失败，请联系组织管理员了解标签策略详情。标签策略详细介绍请参考[标签策略概述](#)，标签管理详细介绍请参考[标签管理](#)。

删除标签：

如需删除标签，在“编辑标签”页面单击标签操作列的“删除”。

**警告**

如果配置转储时使用了该标签，删除标签后，请同步修改转储配置信息。

步骤4 单击“确定”，主机组标签添加完成。在主机组列表的“标签”列可以查看主机组已添加的标签。

----结束

删除主机组

如果主机组不再需要使用，可以删除主机组。

**警告**

删除主机组会导致用户日志接入相关业务异常，主机组删除后无法恢复，请谨慎操作。

步骤1 左侧导航栏选择“主机管理 > 主机组”，进入“主机组”页面。

步骤2 删除单个主机组。

1. 单击待删除的主机组所在行操作列的删除。
2. 在弹出的“删除主机组”页面，单击“确定”，删除该主机组。

步骤3 批量删除主机组。

1. 批量勾选待删除的主机组，单击列表左上方“批量删除”。
2. 在弹出的“删除主机组”页面，单击“确定”，删除所勾选的主机组。

----结束

相关文档

LTS支持通过调用API接口实现创建主机组、修改主机组、删除主机组等，详细请参考[主机组管理](#)。

4.2.6 云主机 ECS 文本日志接入 LTS

弹性云服务器（Elastic Cloud Server）作为核心计算资源，承载着大量业务应用的运行。应用运行会产生一些文本日志数据，例如，应用程序日志、系统日志等。这些日志数据是监控系统健康状态、优化性能和排查故障的基础支撑。为了更好地管理和分析日志，您可以将ECS文本日志接入LTS，实现日志的集中存储、以便于后续的日志查询、分析和告警。

您可以按照如下步骤完成接入配置。

步骤1：选择日志流：选择将日志存储到对应的日志流中，便于不同类型日志分类存储管理。

步骤2：选择主机组（可选）：选择日志接入的主机范围，主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。可以将需要采集日志的一台或多台主机加入到同一个主机组中，再将该主机组关联至同一个接入配置中。

步骤3：采集配置：配置日志的采集路径、采集策略等。

步骤4：索引配置：索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。

步骤5：完成接入配置：接入成功后，可以在接入列表进行任务的增删改查。

若需要采集多个场景的日志，您可以选择[批量设置多个接入配置](#)的方式，同时设置多个接入配置。

约束限制

- ICAgent采集插件的限制请参考[ICAgent限制](#)。
- 相同主机的同一个日志采集路径，如果在AOM进行了配置，则不能在LTS重复配置。详情请参考AOM的[配置虚拟机日志采集路径](#)。
- 配置日志接入时可以同时选择多个主机组，同一个主机组又可以添加多个主机。所以不同主机的路径可以通过主机组来控制采集配置接入。
- LTS下发采集配置后，ICAgent对文件的采集规则如下：
 - 新文件（首次采集）：若文件最后修改时间与当前时间相差超过6小时，则不采集。
 - 旧文件（非首次采集）：若文件最后修改时间与当前时间相差超过3分钟，则不采集。
- LTS暂不支持采集PostgreSQL（数据库）实例的日志。

前提条件

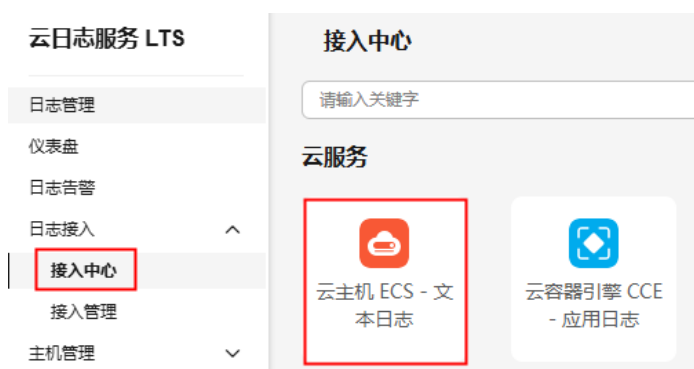
- 为用户添加云日志服务LTS的操作权限。详细请参考[授权IAM用户使用云日志服务LTS](#)。
- 已创建日志组和日志流。详细请参考[管理日志组](#)和[管理日志流](#)。
- 准备好需要采集日志的ECS主机。如果您已有可用的ECS主机，可重复使用，不需要再次创建。详细请参考[自定义购买ECS](#)。

- 主机已安装ICAgent，根据主机所在位置选择安装方式，详细请参考[安装ICAgent（区域内主机）](#)、[安装ICAgent（区域外主机）](#)。
- 已将安装ICAgent的主机加入主机组，详细请参考[管理LTS主机组](#)。
- 为了确保主机能正常上报日志数据和使用指标，主机的安全组开放入方向对应端口（8149、8102、8923、30200、30201、80）。详情请参考[添加安全组规则](#)。
- 日志接入前，需要确认开启ICAgent采集开关，请参考[设置LTS日志采集配额和使用量预警](#)。
 - 采集开关默认打开，当您不需要采集日志时，可通过关闭采集开关来停止日志采集，以减少资源占用。
 - 日志采集关闭后，ICAgent会停止采集日志，并且在应用运维管理AOM控制台的“日志采集开关”也会同步关闭。

步骤 1：选择日志流

1. 登录[云日志服务控制台](#)。
2. 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“云主机 ECS - 文本日志”进行ECS接入配置。

图 4-22 进入 ECS 接入配置



或在左侧导航栏中，选择“日志接入 > 接入管理”，单击“创建”，在弹出的页面中，选择“云主机 ECS - 文本日志”进行ECS接入配置。

图 4-23 进入 ECS 接入配置



3. 选择日志组。
 - **选择已创建的日志组：**单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组名称。
 - **若没有所需的日志组：**单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组，详情请参考[管理日志组](#)。

- 4. 选择日志流。
 - **选择已创建的日志流：**单击“所属日志流”后的目标框，在下拉列表中选择已创建的日志流。
 - **若没有所需的日志流：**单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。详情请参考[管理日志流](#)。
- 5. 单击“下一步：选择主机组”。

步骤 2：选择主机组（可选）

主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。请确保待采集日志的主机已[安装ICAgent](#)并[添加至主机组](#)。

- 1. 选择主机组。
 - **勾选已有主机组：**在主机组列表中勾选一个或多个需要采集日志的主机组。

注意

若此处不勾选主机组，单击“下一步：采集配置”，界面弹窗提示确认后，可以强制跳过。但是这会导致采集配置不生效，建议接入配置时选择主机组。

若此处未勾选主机组，可以通过以下两种方式对主机组进行设置。

- 在左侧导航栏选择“主机管理 > 主机组”，对主机组和接入配置进行关联。请参考[图4-24](#)。

图 4-24 主机组与接入配置关联



- 接入配置完成后，在左侧导航栏选择“日志接入 > 接入管理”，单击对应接入配置操作列的“修改”，进入编辑页面对主机组和接入配置进行关联。请参考[图4-25](#)。

图 4-25 修改接入配置



- **若没有所需的主机组：**单击列表左上方“新建”进入“新建主机组”页面。

图 4-26 新建主机组

新建主机组

* 主机组名称

只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或小数点结尾

* 主机组类型

* 主机类型

备注

0/1024

添加主机

主机列表 ⚠ 请勾选已安装ICAgent的主机加入主机组，否则日志可能无法正常采集

Q 点此处添加筛选条件

☐	主机名称	主机IPv4	主机IPv6	企业项目	ICAg...	ICAg...	更新...
暂无主机							
主机接入需要先配置安装ICAgent，如果你的主机没有出现在这里，请前往安装ICAgent。							
安装ICAgent							

- **若主机已安装ICAgent：**在“新建主机组”页面添加主机下方，勾选已安装ICAgent的主机。具体操作可参考[管理LTS主机组](#)。
- **若主机没有安装ICAgent：**在“新建主机组”页面添加主机下方，单击“安装ICAgent”为主机安装ICAgent。根据主机所在位置选择区域内主机安装或区域外主机安装。具体操作请参考[安装ICAgent（区域内主机）](#)和[安装ICAgent（区域外主机）](#)。在主机安装ICAgent完成后，在接入日志页面需要重新单击“新建”，进入“新建主机组”页面，在主机列表下方即可看到主机信息。

2. 单击“下一步：采集配置”。

步骤 3：采集配置

采集配置项涉及日志的采集范围、采集方式、格式处理等内容配置，请参考以下步骤完成采集配置。

图 4-27 采集基本配置

基本配置 ▾

* 采集配置名称

导入其他配置

只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾

* 路径配置 ①

绕接规则

+ 添加采集路径

删除 添加自定义绕接规则

担心路径填写不正确？使用路径验证

允许文件多次采集

开启后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见 [ICAgent版本说明](#)。
关闭后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。
暂不支持Windows场景

设置采集黑名单 ①

采集Windows事件日志 ②

警告

请注意您的隐私及敏感信息数据保护，不建议将隐私或敏感数据通过访问日志涉及的字段传输，必要时请加密保护。

操作步骤

1. 采集配置名称：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。
- 若需要复用其他已创建好的采集配置信息，单击输入框后面的“导入其他配置”，在导入其他配置页面勾选已创建好的配置，单击“确定”，即可直接复用该采集配置信息，后面的步骤可以不用设置了。

- 导入旧版配置：将旧版主机接入配置导入到新版日志接入中。

■ 若是新安装云日志服务的场景，页面没有显示“导入旧版配置”，则表示不需要导入旧版配置，直接新建配置即可。

■ 若是升级云日志服务的场景，页面显示“导入旧版配置”，若需要旧版配置里的主机日志路径，可以选择导入旧版配置，或者直接新建配置。

2. 路径配置：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。最多支持添加30个路径。采集路径设置规则参考如下：

- 采集路径支持递归路径，**表示递归5层目录。

示例：采集路径配置为 /var/logs/**/a.log，日志匹配如下：
- ```
/var/logs/a.log
/var/logs/1/a.log
/var/logs/1/2/a.log
/var/logs/1/2/3/a.log
/var/logs/1/2/3/4/a.log
/var/logs/1/2/3/4/5/a.log
```
- 文档版本 01 (2025-09-17)

版权所有 © 华为云计算技术有限公司

73



- 以上示例中的/1/2/3/4/5/, 表示/var/logs目录中, 由外向内递归的5个目录层级, 在这5个目录层级中只要存在a.log, 都能进行日志匹配。
  - 采集路径中只能出现一次\*\*, 不能出现两个及以上。正确示例: /var/logs/\*\*/a.log; 错误示例: /opt/test/\*\*/log/\*\*。
  - 采集路径中第一个层级不允许为\*\* (避免误采集系统文件), 错误示例: /\*\*/test。
- **采集路径支持模糊匹配, 匹配目录或文件名中的任何字符。**  
如果配置了C:\windows\system32类似的日志采集路径, 但无法采集日志, 请尝试打开WAF物理防火墙后重新配置。
- 示例1: 采集路径配置为/var/logs/\*/a.log, 表示/var/logs/目录下, 任何一个目录中存在a.log, 都能进行日志匹配, 例如:  

```
/var/logs/1/a.log
/var/logs/2/a.log
```
  - 示例2: 采集路径配置为/var/logs/service-\*/a.log, 日志匹配示例:  

```
/var/logs/service-1/a.log
/var/logs/service-2/a.log
```
  - 示例3: 采集路径配置为/var/logs/service/a\*.log, 日志匹配示例:  

```
/var/logs/service/a1.log
/var/logs/service/a2.log
```
- 如果配置的是文件名, 则直接采集对应文件, 只支持内容是文本格式的文件。
- 添加自定义绕接规则, ICAgent目前是通过文件名规则来判断是否为绕接文件, 如果您的绕接规则不符合内置类型时, 可以通过单击“添加自定义绕接规则”来进行匹配, 避免重复采集和绕接时的日志丢失。
- 内置类型**为{basename}{连接符}{绕接标识}.后缀, {basename}.{后缀}{连接符}{绕接标识}。其中连接符为-.\_绕接标识为非字母符号, 后缀为字母。
- 自定义绕接规则**为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为test.out.log, 绕接后的文件名为test.2024-01-01.0.out.log, test.2024-01-01.1.out.log, 因此在路径配置时, 采集路径为/opt/\*.log, 绕接规则为{basename}\.d{4}-d{2}-d{2}\.d{1}.out.log
- 支持验证采集路径的正确性, 确保正常采集日志。单击“使用路径验证”, 输入采集路径和日志文件绝对路径, 最多支持添加30个采集路径, 单击“确定”, 若路径正确, 界面提示验证成功。
- 支持验证绕接规则的正确性, 确保正常采集日志。单击“使用绕接规则验证”, 输入采集文件名、绕接后的文件名、绕接规则, 单击“确定”, 若绕接规则正确, 界面提示验证成功。
3. **允许文件多次采集。**(暂不支持Windows场景)  
开启“允许文件多次采集”后, 同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本, 详见[查看ICAgent版本说明](#)。  
关闭“允许文件多次采集”后, 采集路径不能重复配置, 即同一主机下的同一日志文件, 即使跨日志流, 也只能配置一次。
4. **设置采集黑名单:** LTS支持对日志进行过滤采集, 即通过设置黑名单, 在采集时过滤指定的目录或文件。  
目录和文件名支持完全匹配, 也支持模糊匹配, 具体可参考[路径配置内容](#)进行设置。

- 当设置的黑名单与配置的采集路径重复或者有重合时，优先过滤掉黑名单设置的文件。
  - 已经加了黑名单的日志，新建日志接入也无法采集黑名单里的日志，除非在设置采集黑名单下方删除采集路径，才能重新采集。
  - 过滤指定的目录时，可以过滤掉该目录下的所有文件，但是不能过滤该目录下文件夹里的日志文件。
  - Windows主机不支持设置采集黑名单。
5. **采集Windows事件日志：**当选择Windows主机采集日志时，需要开启“采集Windows事件日志”。Windows事件日志的采集速率最大支持500条/分钟。请参考[表4-11](#)配置参数：
- 相同主机下，就算跨日志组和日志流，Windows事件日志采集只能配置一次，不能重复配置。

表 4-11 采集 Windows 事件日志参数

| 名称        | 说明                                                                                                                                                                                                                                                     |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日志类型      | 日志类型有系统、应用程序、安全和启动。 <ul style="list-style-type: none"><li>● 系统日志：记录由Windows操作系统组件生成的事件，包括驱动程序、系统服务、硬件故障等。</li><li>● 应用程序日志：记录由用户应用程序或第三方软件生成的事件。</li><li>● 安全日志：记录与系统安全相关的事件，如登录尝试、权限变更、审计策略触发等。</li><li>● 启动日志：记录Windows操作系统启动过程中发生的各种事件信息。</li></ul> |
| 首次采集时间偏移量 | 如设置为7天，表示采集开始时间前7天内的日志（7天前的日志被忽略），该时间仅在首次配置采集生效，确保不会重复采集。最大支持设置为7天。                                                                                                                                                                                    |
| 事件等级      | 事件等级有information（信息）、warning（警告）、error（错误）、critical（严重）和verbose（详细）。根据Windows事件等级过滤采集。仅支持Windows Vista及以上的操作系统。                                                                                                                                        |

6. **结构化解析配置。**
- LTS支持**单行-全文日志、多行-全文日志、JSON、分隔符、单行-完全正则、多行-完全正则、组合解析**多种类型日志解析规则，您可以根据日志内容选择合适的解析规则。日志采集成功后，结构化处理好的日志会发送到指定的日志流上，您就可以对日志进行字段搜索。
- 开启结构化解析配置，详细操作请参考[配置ICAgent结构化解析](#)。

图 4-28 ICAgent 结构化解析配置



- 关闭结构化解析配置，则不会对日志数据进行结构化处理，原始日志格式会发送到指定日志流中，仅可以对日志进行一些常规关键词搜索。
7. 其他配置。设置采集路径后，您还可以设置日志拆分、采集二进制文件、自定义元数据等。

图 4-29 其他配置



表 4-12 其他配置

| 名称      | 说明                                                                                                                                                                                                                                                                                               | 示例 |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 最大目录深度  | 该参数用于控制日志采集路径使用**模糊匹配的文件目录层级，LTS支持设置最大目录层级为20层。<br>例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，最大目录深度设置为5。                                                                                                                                                                | 5  |
| 日志拆分    | 为了防止单条日志过大或被截断丢弃，您可以按文件大小拆分日志。 <ul style="list-style-type: none"><li>开启“日志拆分”，自定义设置日志拆分大小，设置范围为500KB-1024KB。日志拆分大小为500KB，即单条日志超过500KB会被拆分为多条采集。例如：日志大小为600KB，被拆分为2条日志采集，第一条500KB，第二条100KB。仅支持单行日志，不支持多行日志。</li><li>不开启“日志拆分”，单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。</li></ul>                            | 开启 |
| 采集二进制文件 | 支持采集以二进制格式存储的日志数据，您可以通过如下命令查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。<br>file -i 文件名 <ul style="list-style-type: none"><li>当日志的文件类型为二进制时，开启“采集二进制文件”，则对匹配接入规则的二进制日志文件进行采集，但仅支持UTF-8编码的字符串，非UTF-8编码的字符在LTS控制台页面会显示乱码。</li><li>当日志的文件类型为二进制时，未开启“采集二进制文件”，则对匹配接入规则的二进制日志文件停止采集，开启后即可进行采集。</li></ul> | 开启 |

| 名称     | 说明                                                                                                                                                                                                                            | 示例    |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| 日志文件编码 | 日志文件里字符内容的存储格式，支持UTF-8、GBK文件编码类型，GBK暂不支持Windows场景。请合理设置编码，确保日志内容能被正确读取和解析，避免乱码或数据损坏。 <ul style="list-style-type: none"><li>UTF-8编码是一种变长编码方式，用于表示Unicode字符集。</li><li>GBK全称《汉字内码扩展规范》，中文计算机编码的一种，是ASCII码和GB2312编码的扩展。</li></ul> | UTF-8 |
| 采集策略   | ICAgent采集新文件时，您可以设置从文件的末尾或开头开始读取文件。 <ul style="list-style-type: none"><li>增量采集：ICAgent采集新文件时，从文件的末尾开始读。</li><li>全量采集：ICAgent采集新文件时，从文件的开头开始读。</li></ul>                                                                       | 增量采集  |
| 自定义元数据 | <ul style="list-style-type: none"><li>关闭“自定义元数据”，使用ICAgent系统默认配置的字段上报到LTS，不需要用户配置且ICAgent系统不支持配置。</li><li>开启“自定义元数据”，根据用户选择的内置字段和自定义键值增加字段上报到LTS。<br/>系统内置字段：勾选需要设置的内置字段。<br/>自定义键值对：单击“添加”，输入键值key和键值Value。</li></ul>        | 开启    |

8. 参考[表4-13](#)配置日志格式、日志时间。不再推荐使用的配置下的功能，由于设计局限性不再推荐使用，LTS提供新ICAgent结构化解析的功能代替原有功能。若开启“结构化解析配置”按钮则不用设置该参数。

表 4-13 日志采集信息

| 名称   | 说明                                                                                                                                                                                                                          |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日志格式 | <ul style="list-style-type: none"><li>单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。</li><li>多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。</li></ul>                      |
| 日志时间 | <p>系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。</p> <ul style="list-style-type: none"><li>日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。</li><li>日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒。</li><li>采集日志时间限制：系统时间的前后24小时内。</li></ul> |

| 名称    | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | <p>时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。</p> <ul style="list-style-type: none"><li>如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。</li><li>如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。</li></ul> <p>如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。</p> <p>填写示例：</p> <p>YY - year (19)<br/>YYYY - year (2019)<br/>M - month (1)<br/>MM - month (01)<br/>D - day (1)<br/>DD - day (01)<br/>hh - hours (23)<br/>mm - minutes (59)<br/>ss - seconds (59)<br/>SSS - millisecond ( 999 )<br/>hpm - hours (03PM)<br/>h:mmpm - hours:minutes (03:04PM)<br/>h:mm:sspm - hours:minutes:seconds (03:04:05PM)<br/>hh:mm:ss ZZZZ (16:05:06 +0100)<br/>hh:mm:ss ZZZ (16:05:06 CET)<br/>hh:mm:ss ZZ (16:05:06 +01:00)</p> |
| 分行模式  | 日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 正则表达式 | <p>此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。</p> <p>时间通配和正则表达式均是从每行日志的开头进行严格匹配，如果匹配不上，则会默认使用系统时间上报，这样可能会和文件内容中的时间不一致。<b>如果没有特殊需求，建议使用单行日志-系统时间模式即可。</b></p> <p>ICAgent只支持RE2风格的正则表达式，详细请参考<a href="#">语法</a>。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

9. 单击“下一步：索引配置”。

## 步骤 4：索引配置

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

- 若您暂时无需通过某字段进行查询或分析，接入时可以直接跳过索引配置，不影响日志采集。待日志接入成功后，再根据业务需求优化索引配置（**仅对新写入的日志生效**），详细操作请参考[创建LTS日志索引](#)。在“索引配置”页面保持默认设置，直接单击“跳过并提交”，界面提示“日志接入成功”。
- 若您需要通过特定字段进行查询或分析，接入时请在“索引配置”页面参考[创建LTS日志索引](#)配置索引。

您可以单击“自动配置”，云日志服务LTS会根据近15分钟的第一条日志内容或常见内置保留字段（例如hostIP、hostName、pathFile）自动生成字段索引。或者手动添加字段，只能添加已结构化的字段。设置完成后，单击“提交”，界面提示“日志接入成功”。日志接入成功后，您仍可以根据业务需求随时调整索引配置（仅对新写入的日志生效）。

步骤 5：完成接入配置

接入成功后会生成一条接入配置信息。

图 4-30 接入配置

| 接入配置名称 | 接入配置类型 | 接入配置描述 | 接入配置索引 | 接入配置开关 | 操作 |
|--------|--------|--------|--------|--------|----|
| 接入配置名称 | 接入配置类型 | 接入配置描述 | 接入配置索引 | 接入配置开关 | 操作 |

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“修改”重新修改接入配置信息。
- 单击接入配置操作列的“更多 > 标签管理”即可添加标签。
- 单击接入配置操作列的“更多 > 复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。

 警告

删除接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，且删除后无法恢复，请谨慎操作。

- 若不需要采集日志，关闭“接入配置开关”列的开关。若需要重新采集日志，需要重新开启“接入配置开关”列的开关。

 警告

关闭接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，请谨慎操作。

- 单击接入配置操作列的“更多 > ICAgent采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。如果没有显示该功能，请参考[设置 ICAgent日志采集开关](#)开启“ICAgent诊断开关”。

批量设置多个接入配置

支持同时批量设置多个接入配置，操作简单，不用重复配置即可快速完成多个场景的接入配置。

- 步骤1 在“接入管理”页面，单击“批量创建”，进入配置详情页面。
1. 接入类型：选择云主机 ECS-文本日志。
  2. 规则列表：
    - 在输入框填写接入配置数量，单击“添加接入配置”。
    - 在右侧配置项下方自定义填写“规则名称”，配置项设置完成后，也可以双击左侧接入配置的名称自定义命名。规则名称只支持输入英文、数字、中



文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。长度为1-64个字符。

- 鼠标放在接入配置上面，单击即可直接复制接入配置。
- 鼠标放在接入配置上面，单击，在弹出的提示框单击“确定”删除接入配置。

### 3. 配置项：

- 左侧显示接入配置的信息，最多支持添加99个配置。
- 右侧显示配置接入的内容，详细请参考[步骤3：采集配置](#)进行设置。
- 一个接入配置设置完成后，单击“应用于其他接入配置”即可将该接入配置复制到其他接入配置。

**步骤2** 单击“参数检查”，检查成功后，单击“提交”，批量接入设置完成。

例如添加了4个接入配置，批量创建成功后，返回“接入管理”下方，就会显示4条接入配置任务。

**步骤3** （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个接入配置开关显示关闭的接入配置，单击“接入配置开启/关闭”，选择“开启”即可批量开启接入配置任务。
- 勾选多个接入配置开关显示开启的接入配置，单击“接入配置开启/关闭”，选择“关闭”。接入配置状态关闭后不会继续采集日志，请谨慎操作。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

## 添加接入配置标签

支持对已创建的接入配置任务添加标签，每个标签由“标签键”和“标签值”组成，用于快速识别、检索和管理接入配置任务。添加、修改、删除标签仅对当前接入配置有效。

**步骤1** 左侧导航栏选择“日志接入 > 接入管理”，进入“接入管理”页面。

**步骤2** 单击目标接入配置操作列的“更多 > 标签管理”。

**步骤3** 在弹出的编辑标签页面，单击“添加”，填写标签键和标签值。如需添加多个标签可重复该步骤，最多支持添加20个标签。

图 4-31 接入配置编辑标签



**标签键key限制条件：**

- 标签键可以包含任意语种的字母、数字和空格，以及\_:=+-@字符，但首尾不能包含空格，且不能以\_sys\_开头。
- 标签键长度不能超过128个字符。
- 标签键名称不可重复。

#### 标签值value限制条件：

- 标签值可以包含任意语种的字母、数字和空格，以及\_:=+-@字符。
- 标签值长度不能超过255个字符。

#### 标签策略：

若您的组织已经设定云日志服务的相关标签策略，则需按照标签策略规则为日志组、日志流、日志接入、主机组添加标签。标签如果不符合标签策略的规则，则可能会导致日志组、日志流、日志接入、主机组创建失败，请联系组织管理员了解标签策略详情。标签策略详细介绍请参考[标签策略概述](#)，标签管理详细介绍请参考[标签管理](#)。

#### 删除标签：

如需删除标签，在“编辑标签”页面单击标签操作列的“删除”。

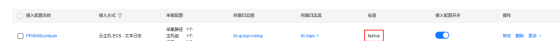


#### 警告

如果配置转储时使用了该标签，删除标签后，请同步修改转储配置信息。

**步骤4** 单击“确定”，接入配置的标签添加完成。在接入配置列表的“标签”列可以查看接入配置已添加的标签。

图 4-32 接入标签



----结束

## 相关文档

- 日志接入LTS成功后，您可通过云日志服务LTS的搜索与分析功能快速洞察日志数据，请参考[日志搜索与分析（即将下线SQL分析方式）](#)。
- 日志接入LTS成功后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- 如果在安装ICAgent时遇到问题，您可以参考[主机管理常见问题](#)获取解决方案。
- 如果在配置日志接入时遇到问题，您可以参考[日志接入常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志接入、查询日志接入、删除日志接入等，详细请参考[日志接入](#)。
- 为您推荐几种在真实使用场景下的数据采集实践教程。更多信息请参考[日志接入最佳实践](#)。
  - a. 如果是多云或者跨Region采集日志的场景，您可以参考[采集第三方云厂商、互联网数据中心、华为云其他Region云主机日志到LTS](#)。
  - b. 如果需要采集多渠道数据，您可以参考[采集多渠道日志数据到LTS](#)。



## 4.2.7 裸金属服务 BMS 文本日志接入 LTS

裸金属服务器（Bare Metal Server）是一款兼具虚拟机弹性和物理机性能的计算类服务，为您和您的企业提供专属的云上物理服务器，为核心数据库、关键应用系统、高性能计算、大数据等业务提供卓越的计算性能以及数据安全。

当您选择了裸金属服务BMS接入方式时，云日志服务可以将BMS待采集日志的路径配置到日志流中，ICAgent按照日志采集规则采集日志，以日志流为单位发往云日志服务，您可以在云日志服务控制台查看和分析日志，可以确保服务器的稳定运行和信息安全。

您可以按照如下步骤完成接入配置。

**步骤1：选择日志流：**选择将日志存储到对应的日志流中，便于不同类型日志分类存储管理。

**步骤2：选择主机组（可选）：**选择日志接入的主机范围，主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。可以将需要采集日志的一台或多台主机加入到同一个主机组中，再将该主机组关联至同一个接入配置中。

**步骤3：采集配置：**配置日志的采集路径、采集策略等。

**步骤4：索引配置：**索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。

**步骤5：完成接入配置：**接入成功后，可以在接入列表进行任务的增删改查。

若需要采集多个场景的日志，您可以选择**批量设置多个接入配置**的方式，同时设置多个接入配置。

### 约束限制

- ICAgent采集插件的限制请参考[ICAgent限制](#)。
- 相同主机的同一个日志采集路径，如果在AOM进行了配置，则不能在LTS重复配置。详情请参考AOM的[配置虚机日志采集路径](#)。
- 配置日志接入时可以同时选择多个主机组，同一个主机组又可以添加多个主机。所以不同主机的路径可以通过主机组来控制采集配置接入。
- LTS暂不支持采集PostgreSQL（数据库）实例的日志。

### 前提条件

- 为用户添加云日志服务LTS的操作权限。详细请参考[授权IAM用户使用云日志服务LTS](#)。
- 已创建日志组和日志流。详细请参考[管理日志组](#)和[管理日志流](#)。
- 准备好需要采集日志的ECS主机。如果您已有可用的ECS主机，可重复使用，不需要再次创建。详细请参考[自定义购买ECS](#)。
- 主机已安装ICAgent，根据主机所在位置选择安装方式，详细请参考[安装ICAgent（区域内主机）](#)、[安装ICAgent（区域外主机）](#)。
- 已将安装ICAgent的主机加入主机组，详细请参考[管理LTS主机组](#)。
- 日志接入前，需要确认开启ICAgent采集开关，请参考[设置LTS日志采集配额和使用量预警](#)。
  - 采集开关默认打开，当您不需要采集日志时，可通过关闭采集开关来停止日志采集，以减少资源占用。

- 日志采集关闭后，ICAgent会停止采集日志，并且在应用运维管理AOM控制台的“日志采集开关”也会同步关闭。

## 步骤 1：选择日志流

1. 登录[云日志服务控制台](#)。
2. 或在左侧导航栏中，选择“日志接入 > 接入管理”，单击“创建”，在弹出的页面中，选择“裸金属服务 BMS - 文本日志”。
3. 选择日志组。
  - **选择已创建的日志组：**单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组名称。
  - **若没有所需的日志组：**单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组，详情请参考[管理日志组](#)。
4. 选择日志流。
  - **选择已创建的日志流：**单击“所属日志流”后的目标框，在下拉列表中选择已创建的日志流。
  - **若没有所需的日志流：**单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。详情请参考[管理日志流](#)。
5. 单击“下一步：选择主机组（可选）”。

## 步骤 2：选择主机组（可选）

主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。请确保待采集日志的主机已[安装ICAgent](#)并[添加至主机组](#)。

1. 选择主机组。
  - **勾选已有主机组：**在主机组列表中勾选一个或多个需要采集日志的主机组。

### 注意

若此处不勾选主机组，单击“下一步：采集配置”，界面弹窗提示确认后，可以强制跳过。但是这会导致采集配置不生效，建议接入配置时选择主机组。

若此处未勾选主机组，可以通过以下两种方式对主机组进行设置。

- 在左侧导航栏选择“主机管理 > 主机组”，对主机组和接入配置进行关联。请参考[图4-33](#)。

图 4-33 主机组与接入配置关联



- 接入配置完成后，在左侧导航栏选择“日志接入 > 接入管理”，单击对应接入配置操作列的“修改”，进入编辑页面对主机组和接入配置进行关联。请参考[图4-34](#)。

图 4-34 修改接入配置



- 若没有所需的主机组：单击列表左上方“新建”进入“新建主机组”页面。

图 4-35 新建主机组



- 若主机已安装ICAgent：在“新建主机组”页面添加主机下方，勾选已安装ICAgent的主机。具体操作可参考[管理LTS主机组](#)。
- 若主机没有安装ICAgent：在“新建主机组”页面添加主机下方，单击“安装ICAgent”为主机安装ICAgent。根据主机所在位置选择区域内主机安装或区域外主机安装。具体操作请参考[安装ICAgent（区域内主机）](#)和[安装ICAgent（区域外主机）](#)。在主机安装ICAgent完成后，在接入日志页面需要重新单击“新建”，进入“新建主机组”页面，在主机列表下方即可看到主机信息。

2. 单击“下一步：采集配置”。

步骤 3：采集配置

采集配置项涉及日志的采集范围、采集方式、格式处理等内容配置，请参考以下步骤完成采集配置。

1. **采集配置名称**：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。  
若需要复用其他已创建好的采集配置信息，单击输入框后面的“导入其他配置”，在导入其他配置页面勾选已创建好的配置，单击“确定”，即可直接复用该采集配置信息，后面的步骤可以不用设置了。  
**导入旧版配置**：将旧版主机接入配置导入到新版日志接入中。

- 若是新安装云日志服务的场景，页面没有显示“导入旧版配置”，则表示不需要导入旧版配置，直接新建配置即可。
  - 若是升级云日志服务的场景，页面显示“导入旧版配置”，若需要旧版配置里的主机日志路径，可以选择导入旧版配置，或者直接新建配置。
2. **路径配置：**添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。最多支持添加30个路径。采集路径设置规则参考如下：
- **采集路径支持递归路径，\*\*表示递归5层目录。**  
示例：采集路径配置为 /var/logs/\*\*/a.log，日志匹配如下：  

```
/var/logs/a.log
/var/logs/1/a.log
/var/logs/1/2/a.log
/var/logs/1/2/3/a.log
/var/logs/1/2/3/4/a.log
/var/logs/1/2/3/4/5/a.log
```

    - 以上示例中的/1/2/3/4/5/，表示/var/logs目录中，由外向内递归的5个目录层级，在这5个目录层级中只要存在a.log，都能进行日志匹配。
    - 采集路径中只能出现一次\*\*，不能出现两个及以上。正确示例：/var/logs/\*\*/a.log；错误示例：/opt/test/\*\*/log/\*\*。
    - 采集路径中第一个层级不允许为\*\*（避免误采集系统文件），错误示例：/\*\*/test。
  - **采集路径支持模糊匹配，匹配目录或文件名中的任何字符。**  
如果配置了C:\windows\system32类似的日志采集路径，但无法采集日志，请尝试打开WAF物理防火墙后重新配置。
    - 示例1：采集路径配置为/var/logs/\*/a.log，表示/var/logs/目录下，任何一个目录中存在a.log，都能进行日志匹配，例如：  

```
/var/logs/1/a.log
/var/logs/2/a.log
```
    - 示例2：采集路径配置为/var/logs/service-\*/a.log，日志匹配示例：  

```
/var/logs/service-1/a.log
/var/logs/service-2/a.log
```
    - 示例3：采集路径配置为/var/logs/service/a\*.log，日志匹配示例：  

```
/var/logs/service/a1.log
/var/logs/service/a2.log
```
  - 如果配置的是文件名，则直接采集对应文件，只支持内容是文本格式的文件。
  - 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。  
**内置类型**为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-，绕接标识为非字母符号，后缀为字母。  
**自定义绕接规则**为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/\*.log，绕接规则为{basename}\\.d{4}-\\d{2}-\\d{2}\\.d{1}.out.log
  - 支持验证采集路径的正确性，确保正常采集日志。单击“使用路径验证”，输入采集路径和日志文件绝对路径，最多支持添加30个采集路径，单击“确定”，若路径正确，界面提示验证成功。

- 支持验证绕接规则的正确性，确保正常采集日志。单击“使用绕接规则验证”，输入采集文件名、绕接后的文件名、绕接规则，单击“确定”，若绕接规则正确，界面提示验证成功。
3. **允许文件多次采集。**（暂不支持Windows场景）
- 开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见[查看ICAgent版本说明](#)。
- 关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。
4. **设置采集黑名单：**LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。
- 目录和文件名支持完全匹配，也支持模糊匹配，具体可参考[路径配置内容](#)进行设置。
- 当设置的黑名单与配置的采集路径重复或者有重合时，优先过滤掉黑名单设置的文件。
  - 已经加了黑名单的日志，新建日志接入也无法采集黑名单里的日志，除非在设置采集黑名单下方删除采集路径，才能重新采集。
  - 过滤指定的目录时，可以过滤掉该目录下的所有文件，但是不能过滤该目录下文件夹里的日志文件。
  - Windows主机不支持设置采集黑名单。
5. **采集Windows事件日志：**当选择Windows主机采集日志时，需要开启“采集Windows事件日志”。Windows事件日志的采集速率最大支持500条/分钟。请参考[表4-14](#)配置参数：
- 相同主机下，就算跨日志组和日志流，Windows事件日志采集只能配置一次，不能重复配置。

**表 4-14** 采集 Windows 事件日志参数

| 名称        | 说明                                                                                                                                                                                                                                                     |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日志类型      | 日志类型有系统、应用程序、安全和启动。 <ul style="list-style-type: none"><li>● 系统日志：记录由Windows操作系统组件生成的事件，包括驱动程序、系统服务、硬件故障等。</li><li>● 应用程序日志：记录由用户应用程序或第三方软件生成的事件。</li><li>● 安全日志：记录与系统安全相关的事件，如登录尝试、权限变更、审计策略触发等。</li><li>● 启动日志：记录Windows操作系统启动过程中发生的各种事件信息。</li></ul> |
| 首次采集时间偏移量 | 如设置为7天，表示采集开始时间前7天内的日志（7天前的日志被忽略），该时间仅在首次配置采集生效，确保不会重复采集。最大支持设置为7天。                                                                                                                                                                                    |
| 事件等级      | 事件等级有information（信息）、warning（警告）、error（错误）、critical（严重）和verbose（详细）。根据Windows事件等级过滤采集。仅支持Windows Vista及以上的操作系统。                                                                                                                                        |

6. 结构化解析配置。
- LTS支持单行-全文日志、多行-全文日志、JSON、分隔符、单行-完全正则、多行-完全正则、组合解析多种类型日志解析规则，您可以根据日志内容选择合适的解析规则。日志采集成功后，结构化处理好的日志会发送到指定的日志流上，您就可以对日志进行字段搜索。
- 开启结构化解析配置，详细操作请参考[配置ICAgent结构化解析](#)。

图 4-36 ICAgent 结构化解析配置



- 关闭结构化解析配置，则不会对日志数据进行结构化处理，原始日志格式会发送到指定日志流中，仅可以对日志进行一些常规关键词搜索。
7. 其他配置。设置采集路径后，您还可以设置日志拆分、采集二进制文件、自定义元数据等。

表 4-15 其他配置

| 名称      | 说明                                                                                                                                                                                                                                                                                               | 示例 |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 最大目录深度  | 该参数用于控制日志采集路径使用**模糊匹配的文件目录层级，LTS支持设置最大目录层级为20层。<br>例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，最大目录深度设置为5。                                                                                                                                                                | 5  |
| 日志拆分    | 为了防止单条日志过大或被截断丢弃，您可以按文件大小拆分日志。 <ul style="list-style-type: none"><li>开启“日志拆分”，支持自定义设置日志拆分大小，设置范围为500KB-1024KB。日志拆分大小为500KB，即单条日志超过500KB会被拆分为多条采集。例如：日志大小为600KB，被拆分为2条日志采集，第一条500KB，第二条100KB。仅支持单行日志，不支持多行日志。</li><li>不开启“日志拆分”，单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。</li></ul>                          | 开启 |
| 采集二进制文件 | 支持采集以二进制格式存储的日志数据，您可以通过如下命令查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。<br>file -i 文件名 <ul style="list-style-type: none"><li>当日志的文件类型为二进制时，开启“采集二进制文件”，则对匹配接入规则的二进制日志文件进行采集，但仅支持UTF-8编码的字符串，非UTF-8编码的字符在LTS控制台页面会显示乱码。</li><li>当日志的文件类型为二进制时，未开启“采集二进制文件”，则对匹配接入规则的二进制日志文件停止采集，开启后即可进行采集。</li></ul> | 开启 |



| 名称     | 说明                                                                                                                                                                                                                            | 示例    |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| 日志文件编码 | 日志文件里字符内容的存储格式，支持UTF-8、GBK文件编码类型，GBK暂不支持Windows场景。请合理设置编码，确保日志内容能被正确读取和解析，避免乱码或数据损坏。 <ul style="list-style-type: none"><li>UTF-8编码是一种变长编码方式，用于表示Unicode字符集。</li><li>GBK全称《汉字内码扩展规范》，中文计算机编码的一种，是ASCII码和GB2312编码的扩展。</li></ul> | UTF-8 |
| 采集策略   | ICAgent采集新文件时，您可以设置从文件的末尾或开头开始读取文件。 <ul style="list-style-type: none"><li>增量采集：ICAgent采集新文件时，从文件的末尾开始读。</li><li>全量采集：ICAgent采集新文件时，从文件的开头开始读。</li></ul>                                                                       | 增量采集  |
| 自定义元数据 | <ul style="list-style-type: none"><li>关闭“自定义元数据”，使用ICAgent系统默认配置的字段上报到LTS，不需要用户配置且ICAgent系统不支持配置。</li><li>开启“自定义元数据”，根据用户选择的内置字段和自定义键值增加字段上报到LTS。<br/>系统内置字段：勾选需要设置的内置字段。<br/>自定义键值对：单击“添加”，输入键值key和键值Value。</li></ul>        | 开启    |

8. 参考[表4-16](#)配置日志格式、日志时间。若开启结构化解析配置则不用设置该参数。

表 4-16 日志采集信息

| 名称   | 说明                                                                                                                                                                                                                   |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日志格式 | <ul style="list-style-type: none"><li>单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。</li><li>多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。</li></ul>               |
| 日志时间 | 系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。 <ul style="list-style-type: none"><li>日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。</li><li>日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒。</li><li>采集日志时间限制：系统时间的前后24小时内。</li></ul> |



| 名称    | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | <p>时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。</p> <ul style="list-style-type: none"><li>如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。</li><li>如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。</li></ul> <p>如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。</p> <p>填写示例：</p> <p>YY - year (19)<br/>YYYY - year (2019)<br/>M - month (1)<br/>MM - month (01)<br/>D - day (1)<br/>DD - day (01)<br/>hh - hours (23)<br/>mm - minutes (59)<br/>ss - seconds (59)<br/>SSS - millisecond ( 999 )<br/>hpm - hours (03PM)<br/>h:mmpm - hours:minutes (03:04PM)<br/>h:mm:sspm - hours:minutes:seconds (03:04:05PM)<br/>hh:mm:ss ZZZZ (16:05:06 +0100)<br/>hh:mm:ss ZZZ (16:05:06 CET)<br/>hh:mm:ss ZZ (16:05:06 +01:00)</p> |
| 分行模式  | 日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 正则表达式 | <p>此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。</p> <p>时间通配和正则表达式均是从每行日志的开头进行严格匹配，如果匹配不上，则会默认使用系统时间上报，这样可能会和文件内容中的时间不一致。<b>如果没有特殊需求，建议使用单行日志-系统时间模式即可。</b></p> <p>ICAgent只支持RE2风格的正则表达式，详细请参考<a href="#">语法</a>。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

9. 单击“下一步：索引配置”。

## 步骤 4：索引配置

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

- 若您暂时无需通过某字段进行查询或分析，接入时可以直接跳过索引配置，不影响日志采集。待日志接入成功后，再根据业务需求优化索引配置（**仅对新写入的日志生效**），详细操作请参考[创建LTS日志索引](#)。在“索引配置”页面保持默认设置，直接单击“跳过并提交”，界面提示“日志接入成功”。
- 若您需要通过特定字段进行查询或分析，接入时请在“索引配置”页面参考[创建LTS日志索引](#)配置索引。

您可以单击“自动配置”，云日志服务LTS会根据近15分钟的第一条日志内容或常见内置保留字段（例如hostIP、hostName、pathFile）自动生成字段索引。或者手动添加字段，只能添加已结构化的字段。设置完成后，单击“提交”，界面提示“日志接入成功”。日志接入成功后，您仍可以根据业务需求随时调整索引配置（仅对新写入的日志生效）。

步骤 5：完成接入配置

接入成功后会生成一条接入配置信息。

图 4-37 接入配置

| 接入配置名称 | 接入配置类型 | 接入配置描述 | 接入配置索引 | 接入配置开关 | 操作 |
|--------|--------|--------|--------|--------|----|
| 接入配置名称 | 接入配置类型 | 接入配置描述 | 接入配置索引 | 接入配置开关 | 操作 |

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“修改”重新修改接入配置信息。
- 单击接入配置操作列的“更多 > 标签管理”即可添加标签。
- 单击接入配置操作列的“更多 > 复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。

 警告

删除接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，且删除后无法恢复，请谨慎操作。

- 若不需要采集日志，关闭“接入配置开关”列的开关。若需要重新采集日志，需要重新开启“接入配置开关”列的开关。

 警告

关闭接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，请谨慎操作。

- 单击接入配置操作列的“更多 > ICAgent采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。如果没有显示该功能，请参考[设置 ICAgent日志采集开关](#)开启“ICAgent诊断开关”。

批量设置多个接入配置

支持同时批量设置多个接入配置，操作简单，不用重复配置即可快速完成多个场景的接入配置。

步骤1 在“接入管理”页面，单击“批量创建”，进入配置详情页面。

- 接入类型：选择裸金属服务BMS-文本日志。
- 规则列表：
  - 在输入框填写接入配置数量，单击“添加接入配置”。
  - 在右侧配置项下方自定义填写“规则名称”，配置项设置完成后，也可以双击左侧接入配置的名称自定义命名。规则名称只支持输入英文、数字、中

文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。长度为1-64个字符。

- 鼠标放在接入配置上面，单击即可直接复制接入配置。
- 鼠标放在接入配置上面，单击，在弹出的提示框单击“确定”删除接入配置。

### 3. 配置项：

- 左侧显示接入配置的信息，最多支持添加99个配置。
- 右侧显示配置接入的内容，详细请参考[步骤3：采集配置](#)进行设置。
- 一个接入配置设置完成后，单击“应用于其他接入配置”即可将该接入配置复制到其他接入配置。

**步骤2** 单击“参数检查”，检查成功后，单击“提交”，批量接入设置完成。

例如添加了4个接入配置，批量创建成功后，在“接入管理”下方，就会显示4条接入配置任务。

**步骤3** （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个接入配置开关显示关闭的接入配置，单击“接入配置开启/关闭”，选择“开启”即可批量开启接入配置任务。
- 勾选多个接入配置开关显示开启的接入配置，单击“接入配置开启/关闭”，选择“关闭”。接入配置状态关闭后不会继续采集日志，请谨慎操作。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

## 添加接入配置标签

支持对已创建的接入配置任务添加标签，每个标签由“标签键”和“标签值”组成，用于快速识别、检索和管理接入配置任务。添加、修改、删除标签仅对当前接入配置有效。

**步骤1** 左侧导航栏选择“日志接入 > 接入管理”，进入“接入管理”页面。

**步骤2** 单击目标接入配置操作列的“更多 > 标签管理”。

**步骤3** 在弹出的编辑标签页面，单击“添加”，填写标签键和标签值。如需添加多个标签可重复该步骤，最多支持添加20个标签。

图 4-38 接入配置编辑标签



**标签键key限制条件：**

- 标签键可以包含任意语种的字母、数字和空格，以及\_!:=+-@字符，但首尾不能包含空格，且不能以\_sys\_开头。
- 标签键长度不能超过128个字符。
- 标签键名称不可重复。

标签值value限制条件：

- 标签值可以包含任意语种的字母、数字和空格，以及\_!:=+-@字符。
- 标签值长度不能超过255个字符。

标签策略：

若您的组织已经设定云日志服务的相关标签策略，则需按照标签策略规则为日志组、日志流、日志接入、主机组添加标签。标签如果不符合标签策略的规则，则可能会导致日志组、日志流、日志接入、主机组创建失败，请联系组织管理员了解标签策略详情。标签策略详细介绍请参考[标签策略概述](#)，标签管理详细介绍请参考[标签管理](#)。

删除标签：

如需删除标签，在“编辑标签”页面单击标签操作列的“删除”。

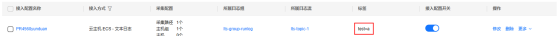


警告

如果配置转储时使用了该标签，删除标签后，请同步修改转储配置信息。

**步骤4** 单击“确定”，接入配置的标签添加完成。在接入配置列表的“标签”列可以查看接入配置已添加的标签。

图 4-39 接入标签



----结束

相关文档

- 日志接入LTS成功后，您可通过云日志服务LTS的搜索与分析功能快速洞察日志数据，请参考[日志搜索与分析（即将下线SQL分析方式）](#)。
- 日志接入LTS成功后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- 如果在安装ICAgent时遇到问题，您可以参考[主机管理常见问题](#)获取解决方案。
- 如果在配置日志接入时遇到问题，您可以参考[日志接入常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志接入、查询日志接入、删除日志接入等，详细请参考[日志接入](#)。
- 为您推荐几种在真实使用场景下的数据采集实践教程。更多信息请参考[日志接入最佳实践](#)。
  - a. 如果是多云或者跨Region采集日志的场景，您可以参考[采集第三方云厂商、互联网数据中心、华为云其他Region云主机日志到LTS](#)。
  - b. 如果需要采集多渠道数据，您可以参考[采集多渠道日志数据到LTS](#)。

## 4.2.8 ServiceStage 云主机日志接入 LTS

云日志服务（Log Tank Service，简称LTS）用于收集来自ServiceStage云主机的日志数据，通过海量日志数据的分析与处理，可以将云服务和应用程序的可用性和性能最大化，为您提供实时、高效、安全的日志处理能力，帮助您快速高效地进行实时决策分析、设备运维管理、用户业务趋势分析等。

### 说明

目前此功能仅支持白名单用户提交工单申请使用。详细操作请参考[提交工单](#)。

您可以按照如下步骤完成接入配置。

**步骤1：选择日志流：**选择将日志存储到对应的日志流中，便于不同类型日志分类存储管理。

**步骤2：选择主机组（可选）：**选择日志接入的主机范围，主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。可以将需要采集日志的一台或多台主机加入到同一个主机组中，再将该主机组关联至同一个接入配置中。

**步骤3：采集配置：**配置日志的采集路径、采集策略等。

**步骤4：索引配置：**索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。

**步骤5：完成接入配置：**接入成功后，可以在接入列表进行任务的增删改查。

若需要采集多个场景的日志，您可以选择[批量设置多个接入配置](#)的方式，同时设置多个接入配置。

### 前提条件

- 已创建ServiceStage应用。详细操作请参考[创建应用](#)。
- 已创建ServiceStage环境。详细操作请参考[创建环境](#)。
- 已创建ServiceStage组件。详细操作请参考[创建组件](#)。
- 已创建日志组和日志流。详细请参考[管理日志组](#)和[管理日志流](#)。
- 主机已安装ICAgent，根据主机所在位置选择安装方式，详细请参考[安装ICAgent（区域内主机）](#)、[安装ICAgent（区域外主机）](#)。
- 已将安装ICAgent的主机加入主机组，详细请参考[管理LTS主机组](#)。
- 日志接入前，需要确认开启ICAgent采集开关，请参考[设置LTS日志采集配额和使用量预警](#)。
  - 采集开关默认打开，当您不需要采集日志时，可通过关闭采集开关来停止日志采集，以减少资源占用。
  - 日志采集关闭后，ICAgent会停止采集日志，并且在应用运维管理AOM控制台的“日志采集开关”也会同步关闭

### 步骤 1：选择日志流

1. 登录[云日志服务控制台](#)。
2. 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“ServiceStage - 云主机日志”进行ServiceStage接入配置。

或在左侧导航栏中，选择“日志接入 > 接入管理”，单击“创建”，在弹出的页面中，单击“ServiceStage - 云主机日志”进行ServiceStage接入配置。

3. 选择应用。在下拉列表中选择具体的ServiceStage应用。
4. 选择环境。在下拉列表中选择具体的ServiceStage环境。
5. 选择日志组。
  - **选择已创建的日志组：**单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组名称。
  - **若没有所需的日志组：**单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组，详情请参考[管理日志组](#)。
6. 选择日志流。
  - **选择已创建的日志流：**单击“所属日志流”后的目标框，在下拉列表中选择已创建的日志流。
  - **若没有所需的日志流：**单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。详情请参考[管理日志流](#)。
7. 单击“下一步：选择主机组（可选）”。

## 步骤 2：选择主机组（可选）

主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。

1. 在主机组列表选择一个或多个需要采集日志的主机组。

**主机组可以为空，但是会导致采集配置不生效，建议第一次接入时选择主机组。**  
若不选择，可以在接入配置设置完成后对主机组进行设置。

  - 在“主机管理 > 主机组”页面对主机组和接入配置进行关联。
  - 在“接入 > 接入管理”页面，单击操作列的“修改”，进入接入配置页面对主机组和接入配置进行关联。
2. 若没有所需的主机组，单击列表左上方“新建”进入“新建主机组”页面。单击“添加标识”填写自定义标识，具体操作可参考[创建主机组（自定义标识）](#)。

图 4-40 新建主机组



新建主机组

\* 主机组名称   
只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以下划线结尾

\* 主机组类型 ☐ IP ☐ 自定义标识

\* 主机类型 ☒ Linux主机 ☐ Windows主机

备注   
0/1024

\* 自定义标识

1. 仅ICAgent 5.12.117 及以上版本支持，升级指引

2. 自定义标识填写指引

| 值                    | 操作                 |
|----------------------|--------------------|
| <input type="text"/> | <a href="#">删除</a> |

[+ 添加](#)

3. 单击“下一步：采集配置”。



### 步骤 3：采集配置

采集配置项涉及日志的采集范围、采集方式、格式处理等内容配置，请参考以下步骤完成采集配置。

1. **采集配置名称**：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。
2. **路径配置**：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。最多支持添加30个路径。采集路径设置规则参考如下：

- **采集路径支持递归路径，\*\*表示递归5层目录。**

示例：采集路径配置为 `/var/logs/**/a.log`，日志匹配如下：

```
/var/logs/a.log
/var/logs/1/a.log
/var/logs/1/2/a.log
/var/logs/1/2/3/a.log
/var/logs/1/2/3/4/a.log
/var/logs/1/2/3/4/5/a.log
```

- 以上示例中的`/1/2/3/4/5/`，表示`/var/logs`目录中，往里递归的5个目录层级，在这5个目录层级中只要存在`a.log`，都能进行日志匹配。
- 采集路径中只能出现一次`**`，不能出现两个及以上。正确示例：`/var/logs/**/a.log`；错误示例：`/opt/test/**/log/**`。
- 采集路径中第一个层级不允许为`**`（避免误采集系统文件），错误示例：`/**/test`。
- **采集路径支持模糊匹配，匹配目录或文件名中的任何字符。**如果配置了`C:\windows\system32`类似的日志采集路径，但无法采集日志，请尝试打开WAF物理防火墙后重新配置。
  - 示例1：采集路径配置为 `/var/logs/*/a.log`，表示`/var/logs/`目录下，任何一个目录中存在`a.log`，都能进行日志匹配，例如：  
`/var/logs/1/a.log`  
`/var/logs/2/a.log`
  - 示例2：采集路径配置为 `/var/logs/service-*/a.log`，日志匹配示例：  
`/var/logs/service-1/a.log`  
`/var/logs/service-2/a.log`
  - 示例3：采集路径配置为 `/var/logs/service/a*.log`，日志匹配示例：  
`/var/logs/service/a1.log`  
`/var/logs/service/a2.log`
- 如果配置的是文件名，则直接采集对应文件，只支持内容是文本格式的文件。
- 支持验证采集路径的正确性，确保正常采集日志。单击“使用路径验证”，输入采集路径和日志文件绝对路径，最多支持添加30个采集路径，单击“确定”，若路径正确，界面提示验证成功。

3. **允许文件多次采集。**（暂不支持Windows场景）

开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见[查看ICAgent版本说明](#)。



关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。

4. **设置采集黑名单：**LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。目录和文件名支持完全匹配，也支持模糊匹配，具体可参考[路径配置内容](#)进行设置。
- 当设置的黑名单与配置的采集路径重复或者有重合时，优先过滤掉黑名单设置的文件。
5. **采集Windows事件日志：**当选择Windows主机采集日志时，需要开启“采集Windows事件日志”，配置如下参数：

表 4-17 采集 Windows 事件日志参数

| 名称        | 说明                                                                                                                                                                                                                                             |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日志类型      | 日志类型有系统、应用程序、安全和启动。 <ul style="list-style-type: none"><li>系统日志：记录由Windows操作系统组件生成的事件，包括驱动程序、系统服务、硬件故障等。</li><li>应用程序日志：记录由用户应用程序或第三方软件生成的事件。</li><li>安全日志：记录与系统安全相关的事件，如登录尝试、权限变更、审计策略触发等。</li><li>启动日志：记录Windows操作系统启动过程中发生的各种事件信息。</li></ul> |
| 首次采集时间偏移量 | 如设置为7天，表示从采集开始时间前7天内的日志（7天前的日志被忽略），该时间仅在首次配置采集生效，确保不会重复采集。最大支持设置为7天。                                                                                                                                                                           |
| 事件等级      | 事件等级有information、warning、error、critical和verbose。根据Windows事件等级过滤采集。仅支持Windows Vista及以上的操作系统。                                                                                                                                                    |

6. **设置ServiceStage匹配规则，**选择对应组件。
7. **结构化解析配置。**
- LTS支持单行-全文日志、多行-全文日志、JSON、分隔符、单行-完全正则、多行-完全正则、组合解析多种类型日志解析规则，您可以根据日志内容选择合适的解析规则。日志采集成功后，结构化处理好的日志会发送到指定的日志流上，您就可以对日志进行字段搜索。
- 开启结构化解析配置，详细操作请参考[配置ICAgent结构化解析](#)。

图 4-41 ICAgent 结构化解析配置



- 关闭结构化解析配置，则不会对日志数据进行结构化处理，原始日志格式会发送到指定日志流中，仅可以对日志进行一些常规关键词搜索。

8. **其他配置。**设置采集路径后，您还可以设置日志拆分、采集二进制文件、自定义元数据等。

**表 4-18 其他配置**

| 名称      | 说明                                                                                                                                                                                                                                                                                                            | 示例    |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| 最大目录深度  | 该参数用于控制日志采集路径使用**模糊匹配的文件目录层级，LTS支持设置最大目录层级为20层。<br>例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，最大目录深度设置为5。                                                                                                                                                                             | 5     |
| 日志拆分    | 为了防止单条日志过大或被截断丢弃，您可以按文件大小拆分日志。 <ul style="list-style-type: none"><li>开启“日志拆分”，支持自定义设置日志拆分大小，设置范围为500KB-1024KB。日志拆分大小为500KB，即单条日志超过500KB会被拆分为多条采集。例如：日志大小为600KB，被拆分为2条日志采集，第一条500KB，第二条100KB。仅支持单行日志，不支持多行日志。</li><li>不开启“日志拆分”，单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。</li></ul>                                       | 开启    |
| 采集二进制文件 | 支持采集以二进制格式存储的日志数据，您可以通过如下命令查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。<br><code>file -i 文件名</code> <ul style="list-style-type: none"><li>当日志的文件类型为二进制时，开启“采集二进制文件”，则对匹配接入规则的二进制日志文件进行采集，但仅支持UTF-8编码的字符串，非UTF-8编码的字符在LTS控制台页面会显示乱码。</li><li>当日志的文件类型为二进制时，未开启“采集二进制文件”，则对匹配接入规则的二进制日志文件停止采集，开启后即可进行采集。</li></ul> | 开启    |
| 日志文件编码  | 日志文件里字符内容的存储格式，支持UTF-8、GBK文件编码类型，GBK暂不支持Windows场景。请合理设置编码，确保日志内容能被正确读取和解析，避免乱码或数据损坏。 <ul style="list-style-type: none"><li>UTF-8编码是一种变长编码方式，用于表示Unicode字符集。</li><li>GBK全称《汉字内码扩展规范》，中文计算机编码的一种，是ASCII码和GB2312编码的扩展。</li></ul>                                                                                 | UTF-8 |
| 采集策略    | ICAgent采集新文件时，您可以设置从文件的末尾或开头开始读取文件。 <ul style="list-style-type: none"><li>增量采集：ICAgent采集新文件时，从文件的末尾开始读。</li><li>全量采集：ICAgent采集新文件时，从文件的开头开始读。</li></ul>                                                                                                                                                       | 增量采集  |

| 名称     | 说明                                                                                                                                                                                                                     | 示例 |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 自定义元数据 | <ul style="list-style-type: none"><li>关闭“自定义元数据”，使用ICAgent系统默认配置的字段上报到LTS，不需要用户配置且ICAgent系统不支持配置。</li><li>开启“自定义元数据”，根据用户选择的内置字段和自定义键值增加字段上报到LTS。<br/>系统内置字段：勾选需要设置的内置字段。<br/>自定义键值对：单击“添加”，输入键值key和键值Value。</li></ul> | 开启 |

9. 参考表4-19设置日志格式、日志时间。若开启结构化解析配置则不用设置该参数。

表 4-19 日志采集信息

| 名称   | 说明                                                                                                                                                                                                                          |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日志格式 | <ul style="list-style-type: none"><li>单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。</li><li>多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。</li></ul>                      |
| 日志时间 | <p>系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。</p> <ul style="list-style-type: none"><li>日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。</li><li>日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒。</li><li>采集日志时间限制：系统时间的前后24小时内。</li></ul> |

| 名称    | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | <p>时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。</p> <ul style="list-style-type: none"><li>如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。</li><li>如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。</li></ul> <p>填写示例：</p> <p>YY - year (19)<br/>YYYY - year (2019)<br/>M - month (1)<br/>MM - month (01)<br/>D - day (1)<br/>DD - day (01)<br/>hh - hours (23)<br/>mm - minutes (59)<br/>ss - seconds (59)<br/>SSS - millisecond ( 999 )<br/>hpm - hours (03PM)<br/>h:mmpm - hours:minutes (03:04PM)<br/>h:mm:sspm - hours:minutes:seconds (03:04:05PM)<br/>hh:mm:ss ZZZZ (16:05:06 +0100)<br/>hh:mm:ss ZZZ (16:05:06 CET)<br/>hh:mm:ss ZZ (16:05:06 +01:00)</p> |
| 分行模式  | 日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 正则表达式 | 此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。ICAgent只支持RE2风格的正则表达式，详细请参考 <a href="#">语法</a> 。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

10. 单击“下一步：索引配置”。

## 步骤 4：索引配置

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

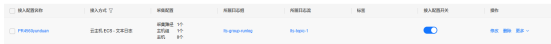
- 若您暂时无需通过某字段进行查询或分析，接入时可以直接跳过索引配置，不影响日志采集。待日志接入成功后，再根据业务需求优化索引配置（仅对新写入的日志生效），详细操作请参考[创建LTS日志索引](#)。在“索引配置”页面保持默认设置，直接单击“跳过并提交”，界面提示“日志接入成功”。
- 若您需要通过特定字段进行查询或分析，接入时请在“索引配置”页面参考[创建LTS日志索引](#)配置索引。

您可以单击“自动配置”，云日志服务LTS会根据近15分钟的第一条日志内容或常见内置保留字段（例如hostIP、hostName、pathFile）自动生成字段索引。或者手动添加字段，只能添加已结构化的字段。设置完成后，单击“提交”，界面提示“日志接入成功”。日志接入成功后，您仍可以根据业务需求随时调整索引配置（仅对新写入的日志生效）。

步骤 5：完成接入配置

接入成功后会生成一条接入配置信息。

图 4-42 接入配置



- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“修改”重新修改接入配置信息。
- 单击接入配置操作列的“更多 > 标签管理”即可添加标签。
- 单击接入配置操作列的“更多 > 复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。

 警告

删除接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，且删除后无法恢复，请谨慎操作。

- 若不需要采集日志，关闭“接入配置开关”列的开关。若需要重新采集日志，需要重新开启“接入配置开关”列的开关。

 警告

关闭接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，请谨慎操作。

- 单击接入配置操作列的“更多 > ICAgent采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。如果没有显示该功能，请参考[设置 ICAgent日志采集开关](#)开启“ICAgent诊断开关”。

批量设置多个接入配置

支持同时批量设置多个接入配置，操作简单，不用重复配置即可快速完成多个场景的接入配置。

**步骤1** 在“接入管理”页面，单击“批量创建”，进入配置详情页面。

1. 接入类型：选择ServiceStage-云主机日志。
2. 规则列表：
  - 在输入框填写接入配置数量，单击“添加接入配置”。
  - 在右侧配置项下方自定义填写“规则名称”，配置项设置完成后，也可以双击左侧接入配置的名称自定义命名。规则名称只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。长度为1-64个字符。
  - 鼠标放在接入配置上面，单击即可直接复制接入配置。
  - 鼠标放在接入配置上面，单击，在弹出的提示框单击“确定”删除接入配置。

### 3. 配置项：

- 左侧显示接入配置的信息，最多支持添加99个配置。
- 右侧显示配置接入的内容，详细请参考[步骤3：采集配置](#)进行设置。
- 一个接入配置设置完成后，单击“应用于其他接入配置”即可将该接入配置复制到其他接入配置。

**步骤2** 单击“参数检查”，检查成功后，单击“提交”，批量接入设置完成。

例如添加了4个接入配置，批量创建成功后，在“接入管理”下方，就会显示4条接入配置任务。

**步骤3** （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个接入配置开关显示关闭的接入配置，单击“接入配置开启/关闭”，选择“开启”即可批量开启接入配置任务。
- 勾选多个接入配置开关显示开启的接入配置，单击“接入配置开启/关闭”，选择“关闭”。接入配置状态关闭后不会继续采集日志，请谨慎操作。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

## 添加接入配置标签

支持对已创建的接入配置任务添加标签，每个标签由“标签键”和“标签值”组成，用于快速识别、检索和管理接入配置任务。添加、修改、删除标签仅对当前接入配置有效。

**步骤1** 左侧导航栏选择“日志接入 > 接入管理”，进入“接入管理”页面。

**步骤2** 单击目标接入配置操作列的“更多 > 标签管理”。

**步骤3** 在弹出的编辑标签页面，单击“添加”，填写标签键和标签值。如需添加多个标签可重复该步骤，最多支持添加20个标签。

图 4-43 接入配置编辑标签



### 标签键key限制条件：

- 标签键可以包含任意语种的字母、数字和空格，以及\_:=+~@字符，但首尾不能包含空格，且不能以\_sys\_开头。
- 标签键长度不能超过128个字符。
- 标签键名称不可重复。

### 标签值value限制条件：

- 标签值可以包含任意语种的字母、数字和空格，以及\_!:=+~@字符。
- 标签值长度不能超过255个字符。

#### 标签策略：

若您的组织已经设定云日志服务的相关标签策略，则需按照标签策略规则为日志组、日志流、日志接入、主机组添加标签。标签如果不符合标签策略的规则，则可能会导致日志组、日志流、日志接入、主机组创建失败，请联系组织管理员了解标签策略详情。标签策略详细介绍请参考[标签策略概述](#)，标签管理详细介绍请参考[标签管理](#)。

#### 删除标签：

如需删除标签，在“编辑标签”页面单击标签操作列的“删除”。

**警告**

如果配置转储时使用了该标签，删除标签后，请同步修改转储配置信息。

**步骤4** 单击“确定”，接入配置的标签添加完成。在接入配置列表的“标签”列可以查看接入配置已添加的标签。

图 4-44 接入标签



----结束

## 相关文档

- 日志接入LTS成功后，您可通过云日志服务LTS的搜索与分析功能快速洞察日志数据，请参考[日志搜索与分析（即将下线SQL分析方式）](#)。
- 日志接入LTS成功后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- 如果在安装ICAgent时遇到问题，您可以参考[主机管理常见问题](#)获取解决方案。
- 如果在配置日志接入时遇到问题，您可以参考[日志接入常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志接入、查询日志接入、删除日志接入等，详细请参考[日志接入](#)。
- 为您推荐几种在真实使用场景下的数据采集实践教程。更多信息请参考[日志接入最佳实践](#)。
  - a. 如果是多云或者跨Region采集日志的场景，您可以参考[采集第三方云厂商、互联网数据中心、华为云其他Region云主机日志到LTS](#)。
  - b. 如果需要采集多渠道数据，您可以参考[采集多渠道日志数据到LTS](#)。

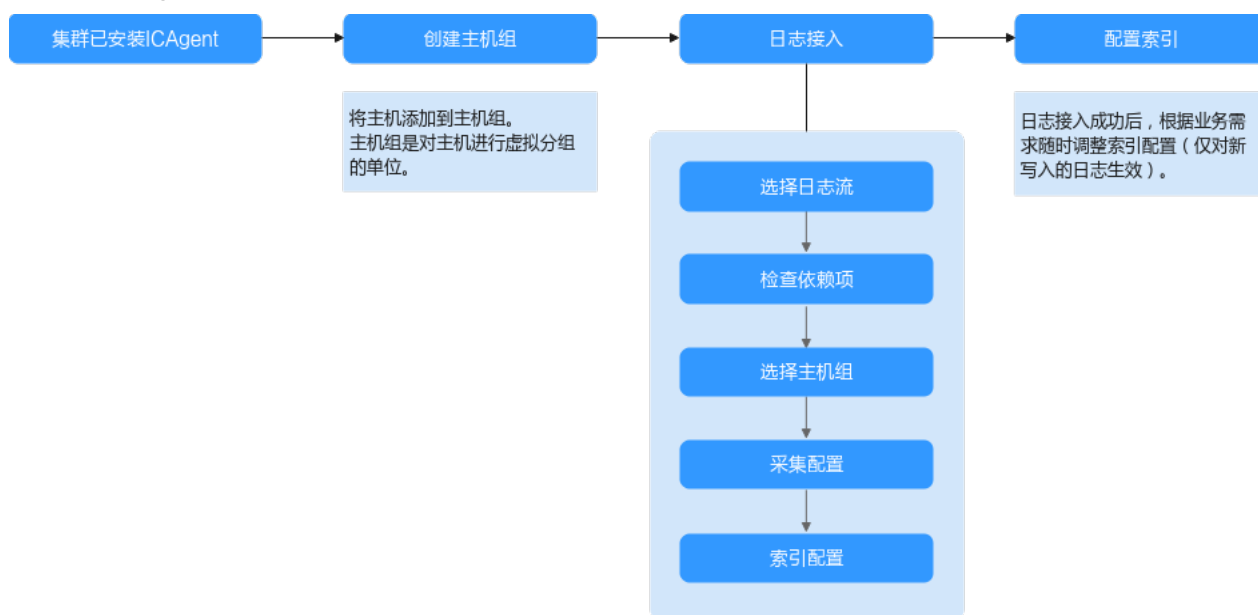
## 4.3 使用 ICAgent 插件采集容器日志

### 4.3.1 ICAgent 采集容器场景日志概述

云日志服务LTS支持通过ICAgent采集以下容器的日志：云容器引擎CCE应用日志、自建K8s（kubernetes）应用日志、ServiceStage容器应用日志。



图 4-45 ICAgent 使用流程



1. **集群已安装 ICAgent**：不同集群安装 ICAgent 的方式不同。
  - 云容器引擎 CCE 集群：首次创建的 CCE 集群，默认集群下的主机已安装了 ICAgent。若没有安装 ICAgent，请参考[容器场景安装 ICAgent 插件](#)。
  - 自建 Kubernetes 集群安装 ICAgent 的方法请参考[自建 K8s 应用日志接入 LTS](#)。
  - ServiceStage 容器化部署：将 ServiceStage 托管的容器应用日志采集到云日志服务，接入前请确保 ServiceStage 服务容器化部署的集群已安装 ICAgent。若没有安装 ICAgent，请参考[容器场景安装 ICAgent 插件](#)。
2. **创建主机组**：将已安装 ICAgent 的主机加入主机组。详细请参考[管理 LTS 主机组](#)。
3. **日志接入**：LTS 以主机组为单位下发采集配置 ICAgent 侧，方便您对多台主机日志进行采集，ICAgent 侧按照采集配置规则上报日志到 LTS。
  - [云容器引擎 CCE 应用日志接入 LTS](#)
  - [ServiceStage 容器应用日志接入 LTS](#)
  - [自建 K8s 应用日志接入 LTS](#)
4. **配置索引**：日志接入成功后，您可以前往索引配置页面，配置全文索引或字段索引，通过日志搜索与分析查看采集上来的日志数据。详细请参考[创建 LTS 日志索引](#)。

### 4.3.2 云容器引擎 CCE 应用日志接入 LTS

云容器引擎（Cloud Container Engine，简称 CCE）提供高度可扩展的、高性能的企业级 Kubernetes 集群。借助云容器引擎，您可以轻松部署、管理和扩展容器化应用程序。

云容器引擎 CCE 应用日志接入 LTS 后，在 LTS 控制台可以对收集到的日志进行统一管理和分析，帮助用户能够快速发现容器的问题，提高容器的性能和可靠性。还可以将日志报表进行可视化展示，从而更好地监控和管理容器化应用的运行状态。

您可以按照如下步骤完成接入配置。

**步骤1：选择日志流**：选择将日志存储到对应的日志流中，便于不同类型日志分类存储管理。

**步骤2：检查依赖项：**自动检查安装ICAgent、自定义标识符、日志组或日志流等检查项是否符合要求。

**步骤3：选择主机组（可选）：**选择日志接入的主机范围，主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。可以将需要采集日志的一台或多台主机加入到同一个主机组中，再将该主机组关联至同一个接入配置中。

**步骤4：采集配置：**配置日志的采集路径、采集策略等。

**步骤5：索引配置：**索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。

**步骤6：完成接入配置：**接入成功后，可以在接入列表进行任务的增删改查。

若需要采集多个场景的日志，您可以选择[批量设置多个接入配置](#)的方式，同时设置多个接入配置。

## 前提条件

- CCE集群已安装ICAgent，详细操作请参考[管理ICAgent](#)。  
当前只支持在CCE Standard集群和Turbo集群上安装ICAgent。不支持在CCE Autopilot集群和CCE敏捷版集群上安装ICAgent。  
(若没有提前安装ICAgent，配置CCE接入LTS时会自动检查修复安装ICAgent，因此推荐预先安装ICAgent，以避免CCE接入LTS时触发自动安装流程带来的等待时间。)
- 将已安装ICAgent的CCE集群加入自定义标识类型的主机组，详细操作请参考[创建主机组（自定义标识）](#)。
- 已关闭采集容器标准输出到AOM的开关。
- 已创建日志组和日志流。详细请参考[管理日志组](#)和[管理日志流](#)。
- 日志接入前，需要确认开启ICAgent采集开关，请参考[设置LTS日志采集配额和使用量预警](#)。
  - 采集开关默认打开，当您不需要采集日志时，可通过关闭采集开关来停止日志采集，以减少资源占用。
  - 日志采集关闭后，ICAgent会停止采集日志，并且在应用运维管理AOM控制台的“日志采集开关”也会同步关闭。
- 如果在LTS主机页面看不到CCE集群，需要AOM2.0授权给LTS后，就可以在LTS页面看到CCE集群。进入AOM2.0控制台后，请参考[云服务授权](#)授予云日志服务(LTS)、云容器引擎(CCE)等云服务数据的访问权限。

## 使用限制

- 支持容器引擎为Docker的CCE集群节点。详情请查看[节点概述](#)
- 支持使用Containerd作为容器引擎的CCE集群节点（ICAgent 5.12.130及以上版本）。
- 支持CCE Turbo集群（ICAgent 5.12.130及以上版本）。
- Docker存储驱动限制：容器文件日志采集目前仅支持overlay2存储驱动，不支持devicemapper作为存储驱动驱动节点。查看存储驱动类型，请使用如下命令：

```
docker info | grep "Storage Driver"
```
- 如果选择日志流时，采集方式为采集到集中日志流时，则必须已创建CCE集群。

- LTS下发采集配置后，ICAgent对文件的采集规则如下：
  - 新文件（首次采集）：若文件最后修改时间与当前时间相差超过12小时，则不采集。
  - 旧文件（非首次采集）：若文件最后修改时间与当前时间相差超过2小时，则不采集。

## 步骤 1：选择日志流

1. 登录[云日志服务控制台](#)。
2. 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“云容器引擎 CCE - 应用日志”进行CCE接入配置。

图 4-46 CCE 接入



或在左侧导航栏中，选择“日志接入 > 接入管理”，单击“创建”，在弹出的页面中，选择“云容器引擎 CCE - 应用日志”进行CCE接入配置。

图 4-47 创建 CCE 接入



3. 采集方式支持“采集到集中日志流”和“采集到自定义日志流”，您可以根据实际情况选择采集方式，推荐您使用采集到集中日志流。

表 4-20 采集方式

| 采集方式      | 说明                                                                                                                                                                                                                                                                     |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 采集到集中日志流  | <ul style="list-style-type: none"><li>• 优点：在CCE界面可以根据命名空间、工作负载、容器名称直接查看对应的日志，将所有日志采集到一个日志流。</li><li>• 缺点：<ul style="list-style-type: none"><li>- 不同的工作负载的日志结构不一样，采集到一个日志流后无法配置结构化解析，无法使用SQL可视化分析。</li><li>- 单个日志流的写入速率上限是100MB/S，集中采集对于大流量场景有性能瓶颈。</li></ul></li></ul> |
| 采集到自定义日志流 | <ul style="list-style-type: none"><li>• 优点：<ul style="list-style-type: none"><li>- 不同的工作负载的日志结构不一样，采集到不同日志流后可以配置结构化解析，使用SQL可视化分析。</li><li>- 多个日志流的写入速率累加起来可以线性扩增，自定义采集对于大流量场景没有性能瓶颈。</li></ul></li><li>• 缺点：在CCE界面无法根据命名空间、工作负载、容器名称直接查看对应的日志。</li></ul>               |

- 若选择“采集到集中日志流”，请执行如下操作步骤：

集中采集日志到一个固定的日志流。CCE集群默认的采集日志流分别为标准输出/错误stdout-{ClusterID}、节点文件hostfile-{ClusterID}、K8S事件event-{ClusterID}和容器文件containerfile-{ClusterID}。日志流名称会根据ClusterID自动命名，例如：集群ID为Cluster01，则标准输出/错误日志流为stdout-Cluster01。

在一个CCE集群下可以创建的采集日志流为标准输出/错误stdout-{ClusterID}、节点文件hostfile-{ClusterID}、K8s事件event-{ClusterID}和容器文件containerfile-{ClusterID}，如果某个日志组下，已创建某种采集日志流，则不会在其他日志组或当前日志组下再创建该日志流。

- 单击“CCE集群”后的目标框，在下拉列表中选择具体的集群。
  - 默认所属日志组为k8s-log-集群ID，例如集群ID为c7f3f4a5-bcb8-11ed-a4ec-0255ac100b07，默认所属日志组为k8s-log-c7f3f4a5-bcb8-11ed-a4ec-0255ac100b07。当无该日志组时，系统会提示：暂无该日志组，后续操作中，系统将会为您自动创建，创建完成后日志会集中采集到该日志组中。
  - 单击“下一步：检查依赖项”。
- 若选择“采集到自定义日志流”，请执行如下操作步骤：
    - 单击“CCE集群”后的目标框，在下拉列表中选择具体的集群。
    - 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
    - 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。

- iv. 单击“下一步：检查依赖项”。

## 步骤 2：检查依赖项

系统自动检查以下内容检查项是否符合要求：

1. 已安装ICAgent，且版本  $\geq 5.12.130$ 。
2. 存在自定义标识符为**k8s-log-集群ID**的主机组。
3. 存在名为**k8s-log-集群ID**的日志组。当选择日志流为**采集到集中日志流**时，会进行该项内容检查。
4. 存在系统推荐的集中采集的日志流。当选择日志流为**采集到集中日志流**时，会进行该项内容检查。

如果以上内容检查项中，有任意一项不符合要求，需单击“自动修复”按钮进行修复，否则将无法进行下一步操作。

- **自动修复**：一键帮您完成以上内容检查项配置。
- **重新检查**：重新检查依赖项。

## 步骤 3：选择主机组（可选）

主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。CCE集群已安装ICAgent，并且已创建自定义标识的主机组（如果不满足，配置CCE接入LTS时会自动检查修复）。

1. 在选择主机组页面，系统自动勾选检查依赖项创建的**k8s-log-集群ID**的主机组。
  - 您可以根据需要选择其他已创建的主机组。
  - 对于自定义CCE集群的主机组，自定义标识的格式需要是**k8s-log-集群ID**。
  - **主机组可以为空，但是会导致采集配置不生效，建议第一次接入时选择主机组。若不选择，可以在接入配置设置完成后对主机组进行设置。**
    - 在“主机管理 > 主机组”页面对主机组和接入配置进行关联。
    - 在“日志接入 > 接入管理”页面，单击操作列的“修改”，进入接入配置页面对主机组和接入配置进行关联。
2. 单击“下一步：采集配置”。

## 步骤 4：采集配置

采集配置项涉及日志的采集范围、采集方式、格式处理等内容配置，请参考以下步骤完成采集配置。

1. **采集配置名称**：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。
2. **数据源配置**：选择数据源类型，进行对应的数据源配置，详细请参考[表4-21](#)。对挂载的网盘或者共享存储，配置多台机器同时采集，会导致日志重复。

表 4-21 数据源配置参数

| 类型     | 参数配置                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 容器标准输出 | <p>采集集群内指定容器的标准输出日志和标准错误日志，仅支持Stderr和Stdout的日志。</p> <p>被匹配上的容器的标准输出会采集到指定的日志流，原先采集到的AOM的标准输出会停止采集。</p> <ul style="list-style-type: none"><li>采集容器标准输出到AOM1.0：默认集群下的主机已安装了ICAgent 且采集日志到AOM，采集容器标准输出到AOM的开关处于关闭状态。开启后标准输出只会采集到AOM，不会采集到LTS，建议您手动关闭该开关。</li><li>采集容器标准输出（stdout）和采集容器标准错误（stderr）两者必须得有一个是开启状态。</li><li>开启采集容器标准错误（stderr）后，选择采集目标路径：将标准输出和标准错误采集到不同的文件（stdout.log和stderr.log）、将标准输出和标准错误采集到同一个文件（stdout.log）。</li><li>允许文件多次采集。（暂不支持Windows场景）<br/>开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见<a href="#">查看ICAgent版本说明</a>。</li></ul> <p>关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。</p> |

| 类型     | 参数配置                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 容器文件路径 | <p>采集集群内指定容器内的文件日志。</p> <ul style="list-style-type: none"><li>● 添加采集路径：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集，更多示例请参考<a href="#">路径配置</a>。<br/>当CCE集群的工作负载中，已配置容器的挂载路径时，此时路径配置里添加的路径将无效。须将CCE集群页面中的挂载路径删除后，该配置才有效。<br/>采集容器日志，当前不支持采集软链接文件，如果采集软链接文件，需要您配置到软链接所指向的文件。</li><li>● 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。<br/><b>内置类型</b>为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-，绕接标识为非字母符号，后缀为字母。<br/><b>自定义绕接规则</b>为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\\.d{4}-\\d{2}-\\d{1}\\.out.log</li><li>● 支持验证采集路径的正确性，确保正常采集日志。单击“使用路径验证”，输入采集路径和日志文件绝对路径，最多支持添加30个采集路径，单击“确定”，若路径正确，界面提示验证成功。</li><li>● 支持验证绕接规则的正确性，确保正常采集日志。单击“使用绕接规则验证”，输入采集文件名、绕接后的文件名、绕接规则，单击“确定”，若绕接规则正确，界面提示验证成功。</li><li>● 允许文件多次采集。（暂不支持Windows场景）<br/>开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见<a href="#">查看ICAgent版本说明</a>。<br/>关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。</li><li>● 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。</li></ul> |



| 类型     | 参数配置                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 节点文件路径 | <p>采集集群内指定节点的文件。</p> <ul style="list-style-type: none"><li>添加采集路径：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集，更多示例请参考<a href="#">路径配置</a>。</li><li>添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。<br/><b>内置类型</b>为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-，绕接标识为非字母符号，后缀为字母。<br/><b>自定义绕接规则</b>为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\\.d{4}-\\.d{2}-\\.d{2}\\.d{1}.out.log</li><li>支持验证采集路径的正确性，确保正常采集日志。单击“使用路径验证”，输入采集路径和日志文件绝对路径，最多支持添加30个采集路径，单击“确定”，若路径正确，界面提示验证成功。</li><li>支持验证绕接规则的正确性，确保正常采集日志。单击“使用绕接规则验证”，输入采集文件名、绕接后的文件名、绕接规则，单击“确定”，若绕接规则正确，界面提示验证成功。</li><li>允许文件多次采集。（暂不支持Windows场景）<br/>开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见<a href="#">查看ICAgent版本说明</a>。<br/>关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。</li><li>设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。</li></ul> |
| K8S事件  | <p>采集K8S集群内的事件日志。</p> <p>K8S事件不能重复配置，即一个K8S集群的K8S事件，只能配置接入到一个日志流。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

3. （可选）**K8s匹配规则**：当数据源类型选择**容器标准输出**和**容器文件路径**时，支持设置K8s匹配规则。

填写正则匹配规则后，单击校验按钮，支持校验确保正则表达式的正确性。  
ICAgent只支持RE2风格的正则表达式，详细请参考[语法](#)。

表 4-22 K8s 匹配规则

| 参数名称               | 参数说明                                                                                                                                                                                                                                                       |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| K8s Namespace 正则匹配 | 通过Namespace名称指定采集的容器，支持正则匹配。<br>采集名称符合正则规则的Namespace的日志，为空时采集所有Namespace的日志。                                                                                                                                                                               |
| K8s Pod正则匹配        | 通过Pod名称指定待采集的容器，支持正则匹配。<br>采集名称符合正则规则的Pod的日志，为空时采集所有Pod的日志。                                                                                                                                                                                                |
| K8s容器名称正则匹配        | 通过容器名称指定待采集的容器（Kubernetes容器名称是定义在spec.containers中），支持正则匹配。<br>采集名称符合正则规则的容器的日志，为空时采集所有容器的日志。                                                                                                                                                               |
| K8s Label白名单       | 通过K8s Label白名单指定待采集的容器。如果您要设置K8s Label白名单，那么LabelKey必填，LabelValue可选填。<br>新增多条白名单时，支持选择And或or的关系，即全部满足或满足任意白名单就可以被匹配。<br>若LabelValue为空，则K8S Label中包含LabelKey的容器都匹配；若LabelValue不为空，则K8S Label中包含LabelKey=LabelValue的容器才匹配；LabelKey需要全匹配，LabelValue支持正则匹配。    |
| K8s Label黑名单       | 通过K8s Label黑名单排除不采集的容器。如果您要设置K8s Label黑名单，那么LabelKey必填，LabelValue可选填。<br>新增多条黑名单时，支持选择And或or的关系，即全部满足或满足任意黑名单就可以被排除。<br>若LabelValue为空，则K8S Label中包含LabelKey的容器都被排除；若LabelValue不为空，则K8S Label中包含LabelKey=LabelValue的容器才会被排除；LabelKey需要全匹配，LabelValue支持正则匹配。 |
| K8s Label日志标签      | 设置K8s Label日志标签后，日志服务将在日志中新增K8s Label相关字段。<br>设置K8S Label日志标签后，lts将在日志中新增相关字段。例如设置LabelKey为app，设置LabelValue为app_alias，当容器中包含app=lts时，将在日志中添加内容{app_alias: lts}。                                                                                            |
| 容器Label白名单         | 通过容器Label白名单指定待采集的容器。如果您要设置容器Label白名单，那么LabelKey必填，LabelValue可选填。<br>新增多条白名单时，支持选择And或or的关系，即全部满足或满足任意白名单就可以被匹配。<br>若LabelValue为空，则容器 Label中包含LabelKey的容器都匹配；若LabelValue不为空，则容器 Label中包含LabelKey=LabelValue的容器才匹配；LabelKey需要全匹配，LabelValue支持正则匹配。          |

| 参数名称        | 参数说明                                                                                                                                                                                                                                                               |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 容器Label黑名单  | <p>通过容器Label黑名单排除不采集的容器。如果您要设置容器Label黑名单，那么LabelKey必填，LabelValue可选填。</p> <p>新增多条黑名单时，支持选择And或or的关系，即全部满足或满足任意黑名单就可以被排除。</p> <p>若LabelValue为空，则容器 Label中包含LabelKey的容器都被排除；若LabelValue不为空，则容器Label中包含LabelKey=LabelValue的容器才会被排除；LabelKey需要全匹配，LabelValue支持正则匹配。</p> |
| 容器Label日志标签 | <p>设置容器Label日志标签后，日志服务将在日志中新增容器Label相关字段。</p> <p>设置容器 Label日志标签后，Its将在日志中新增相关字段。例如设置LabelKey为app，设置LabelValue为app_alias，当容器中包含app=Its时，将在日志中添加内容{app_alias: Its}。</p>                                                                                              |
| 环境变量白名单     | <p>用于指定待采集的容器。如果您要设置环境变量白名单，那么Label Key必填，Label Value可选填。</p> <p>新增多条白名单时，支持选择And或or的关系，即全部满足或满足任意白名单就可以被匹配。</p> <p>如果环境变量Value为空，则容器环境变量中包含环境变量Key的容器都匹配；如果环境变量Value不为空，则容器环境变量中包含环境变量Key=环境变量Value的容器才被匹配；LabelKey需要全匹配，LabelValue支持正则匹配。</p>                    |
| 环境变量黑名单     | <p>用于排除不采集的容器。如果您要设置环境变量黑名单，那么Label Key必填，Label Value可选填。</p> <p>新增多条黑名单时，支持选择And或or的关系，即全部满足或满足任意黑名单就可以被排除。</p> <p>如果环境变量Value为空，则容器环境变量中包含环境变量Key的容器都将被排除；如果环境变量Value不为空，则容器环境变量中包含环境变量Key=环境变量Value的容器才会被排除；LabelKey需要全匹配，LabelValue支持正则匹配。</p>                 |
| 环境变量日志标签    | <p>设置环境变量日志标签后，日志服务将在日志中新增环境变量相关字段。</p> <p>设置环境变量日志标签后，Its将在日志中新增相关字段，例如设置环境变量Key为app，设置环境变量Value为app_alias，当K8S中包含环境变量app=Its时，将在日志中添加的内容为{app_alias: Its}。</p>                                                                                                   |

#### 4. 结构化解析配置。

LTS支持单行-全文日志、多行-全文日志、JSON、分隔符、单行-完全正则、多行-完全正则、组合解析多种类型日志解析规则，您可以根据日志内容选择合适的解析规则。日志采集成功后，结构化处理好的日志会发送到指定的日志流上，您就可以对日志进行字段搜索。

- 开启结构化解析配置，详细操作请参考[配置ICAgent结构化解析](#)。

图 4-48 ICAgent 结构化解析配置



- 关闭结构化解析配置，则不会对日志数据进行结构化处理，原始日志格式会发送到指定日志流中，仅可以对日志进行一些常规关键词搜索。
5. **其他配置。**设置采集路径后，您还可以设置日志拆分、采集二进制文件、自定义元数据等。

表 4-23 其他配置

| 名称      | 说明                                                                                                                                                                                                                                                                                               | 示例 |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 最大目录深度  | 该参数用于控制日志采集路径使用**模糊匹配的文件目录层级，LTS支持设置最大目录层级为20层。<br>例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，最大目录深度设置为5。<br>如果需要采集容器文件路径，需要ICAgent升级到7.3.3版本以上才支持**模糊匹配。                                                                                                                | 5  |
| 日志拆分    | 为了防止单条日志过大或被截断丢弃，您可以按文件大小拆分日志。 <ul style="list-style-type: none"><li>开启“日志拆分”，支持自定义设置日志拆分大小，设置范围为500KB-1024KB。日志拆分大小为500KB，即单条日志超过500KB会被拆分为多条采集。例如：日志大小为600KB，被拆分为2条日志采集，第一条500KB，第二条100KB。仅支持单行日志，不支持多行日志。</li><li>不开启“日志拆分”，单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。</li></ul>                          | 开启 |
| 采集二进制文件 | 支持采集以二进制格式存储的日志数据，您可以通过如下命令查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。<br>file -i 文件名 <ul style="list-style-type: none"><li>当日志的文件类型为二进制时，开启“采集二进制文件”，则对匹配接入规则的二进制日志文件进行采集，但仅支持UTF-8编码的字符串，非UTF-8编码的字符在LTS控制台页面会显示乱码。</li><li>当日志的文件类型为二进制时，未开启“采集二进制文件”，则对匹配接入规则的二进制日志文件停止采集，开启后即可进行采集。</li></ul> | 开启 |

| 名称     | 说明                                                                                                                                                                                                                                       | 示例    |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| 日志文件编码 | <p>日志文件里字符内容的存储格式，支持UTF-8、GBK文件编码类型，GBK暂不支持Windows场景。请合理设置编码，确保日志内容能被正确读取和解析，避免乱码或数据损坏。</p> <ul style="list-style-type: none"><li>• UTF-8编码是一种变长编码方式，用于表示Unicode字符集。</li><li>• GBK全称《汉字内码扩展规范》，中文计算机编码的一种，是ASCII码和GB2312编码的扩展。</li></ul> | UTF-8 |
| 采集策略   | <p>ICAgent采集新文件时，您可以设置从文件的末尾或开头开始读取文件。</p> <ul style="list-style-type: none"><li>• 增量采集：ICAgent采集新文件时，从文件的末尾开始读。</li><li>• 全量采集：ICAgent采集新文件时，从文件的开头开始读。</li></ul>                                                                       | 增量采集  |
| 自定义元数据 | <ul style="list-style-type: none"><li>• 关闭“自定义元数据”，使用ICAgent系统默认配置的字段上报到LTS，不需要用户配置且ICAgent系统不支持配置。</li><li>• 开启“自定义元数据”，根据用户选择的内置字段和自定义键值增加字段上报到LTS。<br/>系统内置字段：勾选需要设置的内置字段。<br/>自定义键值对：单击“添加”，输入键值key和键值Value。</li></ul>               | 开启    |

6. 参考[表4-24](#)配置日志格式、日志时间。若开启结构化解析配置则不用设置该参数。

表 4-24 日志采集信息

| 名称   | 说明                                                                                                                                                                                                                                |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日志格式 | <ul style="list-style-type: none"><li>• 单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。</li><li>• 多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。</li></ul>                        |
| 日志时间 | <p>系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。</p> <ul style="list-style-type: none"><li>• 日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。</li><li>• 日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒。</li><li>• 采集日志时间限制：系统时间的前后24小时内。</li></ul> |

| 名称    | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | <p>时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。</p> <ul style="list-style-type: none"><li>如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。</li><li>如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。</li></ul> <p>填写示例：</p> <p>YY - year (19)<br/>YYYY - year (2019)<br/>M - month (1)<br/>MM - month (01)<br/>D - day (1)<br/>DD - day (01)<br/>hh - hours (23)<br/>mm - minutes (59)<br/>ss - seconds (59)<br/>SSS - millisecond ( 999 )<br/>hpm - hours (03PM)<br/>h:mmpm - hours:minutes (03:04PM)<br/>h:mm:sspm - hours:minutes:seconds (03:04:05PM)<br/>hh:mm:ss ZZZZ (16:05:06 +0100)<br/>hh:mm:ss ZZZ (16:05:06 CET)<br/>hh:mm:ss ZZ (16:05:06 +01:00)</p> |
| 分行模式  | 日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 正则表达式 | <p>此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。</p> <p>时间通配和正则表达式均是从每行日志的开头进行严格匹配，如果匹配不上，则会默认使用系统时间上报，这样可能会和文件内容中的时间不一致。<b>如果没有特殊需求，建议使用单行日志-系统时间模式即可。</b></p> <p>ICAgent只支持RE2风格的正则表达式，详细请参考<a href="#">语法</a>。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

#### 7. 单击“下一步：索引配置”。

## 步骤 5：索引配置

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

- 若您暂时无需通过某字段进行查询或分析，接入时可以直接跳过索引配置，不影响日志采集。待日志接入成功后，再根据业务需求优化索引配置（**仅对新写入的日志生效**），详细操作请参考[创建LTS日志索引](#)。在“索引配置”页面保持默认设置，直接单击“跳过并提交”，界面提示“日志接入成功”。
- 若您需要通过特定字段进行查询或分析，接入时请在“索引配置”页面参考[创建LTS日志索引](#)配置索引。



您可以单击“自动配置”，云日志服务LTS会根据近15分钟的第一条日志内容或常见内置保留字段（例如hostIP、hostName、pathFile）自动生成字段索引。或者手动添加字段，只能添加已结构化的字段。设置完成后，单击“提交”，界面提示“日志接入成功”。日志接入成功后，您仍可以根据业务需求随时调整索引配置（仅对新写入的日志生效）。

## 步骤 6：完成接入配置

接入成功后会生成一条接入配置信息。

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“修改”重新修改接入配置信息。
- 单击接入配置操作列的“更多 > 标签管理”即可添加标签。
- 单击接入配置操作列的“更多 > 复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。



### 警告

删除接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，且删除后无法恢复，请谨慎操作。

- 若不需要采集日志，关闭“接入配置开关”列的开关。若需要重新采集日志，需要重新开启“接入配置开关”列的开关。



### 警告

关闭接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，请谨慎操作。

- 单击接入配置操作列的“更多 > ICAgent采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。如果没有显示该功能，请参考[设置 ICAgent日志采集开关](#)开启“ICAgent诊断开关”。

## 批量设置多个接入配置

支持同时批量设置多个接入配置，操作简单，不用重复配置即可快速完成多个场景的接入配置

**步骤1** 在“接入管理”页面，单击“批量创建”，进入配置详情页面。

1. 接入类型：选择云容器引擎 CCE-应用日志。
2. 规则列表：
  - 在输入框填写接入配置数量，单击“添加接入配置”。
  - 在右侧配置项下方自定义填写“规则名称”，配置项设置完成后，也可以双击左侧接入配置的名称自定义命名。规则名称只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。长度为1-64个字符。
  - 鼠标放在接入配置上面，单击即可直接复制接入配置。



- 鼠标放在接入配置上面，单击，在弹出的提示框单击“确定”删除接入配置。
3. 配置项：
- 左侧显示接入配置的信息，最多支持添加99个配置。
  - 右侧显示配置接入的内容，详细请参考[步骤4：采集配置](#)进行设置。
  - 一个接入配置设置完成后，单击“应用于其他接入配置”即可将该接入配置复制到其他接入配置。

**步骤2** 单击“参数检查”，检查成功后，单击“提交”，批量接入设置完成。

例如添加了4个接入配置，批量创建成功后，在“接入管理”下方，就会显示4条接入配置任务。

**步骤3** （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个接入配置开关显示关闭的接入配置，单击“接入配置开启/关闭”，选择“开启”即可批量开启接入配置任务。
- 勾选多个接入配置开关显示开启的接入配置，单击“接入配置开启/关闭”，选择“关闭”。接入配置状态关闭后不会继续采集日志，请谨慎操作。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

## 相关文档

- 日志接入LTS成功后，您可通过云日志服务LTS的搜索与分析功能快速洞察日志数据，请参考[日志搜索与分析（即将下线SQL分析方式）](#)。
- 日志接入LTS成功后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- 如果在安装ICAgent时遇到问题，您可以参考[主机管理常见问题](#)获取解决方案。
- 如果在配置日志接入时遇到问题，您可以参考[日志接入常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志接入、查询日志接入、删除日志接入等，详细请参考[日志接入](#)。
- 为您推荐几种在真实使用场景下的数据采集实践教程。更多信息请参考[日志接入最佳实践](#)。
  - a. 如果是多云或者跨Region采集日志的场景，您可以参考[采集第三方云厂商、互联网数据中心、华为云其他Region云主机日志到LTS](#)。
  - b. 如果需要采集多渠道数据，您可以参考[采集多渠道日志数据到LTS](#)。

### 4.3.3 ServiceStage 容器应用日志接入 LTS

云日志服务（Log Tank Service，简称LTS）用于收集来自ServiceStage容器应用的日志数据，通过海量日志数据的分析与处理，可以将云服务和应用程序的可用性和性能最大化，为您提供实时、高效、安全的日志处理能力，帮助您快速高效地进行实时决策分析、设备运维管理、用户业务趋势分析等。

#### 说明

目前此功能仅支持白名单用户提交工单申请使用。详细操作请参考[提交工单](#)。

您可以按照如下步骤完成接入配置。

**步骤1：选择日志流：**选择将日志存储到对应的日志流中，便于不同类型日志分类存储管理。

**步骤2：检查依赖项：**自动检查依赖项是否符合要求。

**步骤3：选择主机组（可选）：**选择日志接入的主机范围，主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。可以将需要采集日志的一台或多台主机加入到同一个主机组中，再将该主机组关联至同一个接入配置中。

**步骤4：采集配置：**配置日志的采集路径、采集策略等。

**步骤5：索引配置：**索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。

**步骤6：完成接入配置：**接入成功后，可以在接入列表进行任务的增删改查。

若需要采集多个场景的日志，您可以选择**批量设置多个接入配置**的方式，同时设置多个接入配置。

## 前提条件

- 已创建ServiceStage应用。详细操作请参考[创建应用](#)。
- 已创建ServiceStage环境。详细操作请参考[创建环境](#)。
- 已创建ServiceStage组件。详细操作请参考[创建组件](#)。
- 已创建日志组和日志流。详细请参考[管理日志组](#)和[管理日志流](#)。
- ServiceStage托管的集群已安装ICAgent，详细操作请参考[管理ICAgent](#)。  
（若没有提前安装ICAgent，配置ServiceStage接入LTS时会自动检查修复安装ICAgent，因此推荐预先安装ICAgent，以避免ServiceStage接入LTS时触发自动安装流程带来的等待时间。）
- 将已安装ICAgent的CCE集群加入自定义标识类型的主机组，详细操作请参考[创建主机组（自定义标识）](#)。
- 日志接入前，需要确认开启ICAgent采集开关，请参考[设置LTS日志采集配额和使用量预警](#)。
  - 采集开关默认打开，当您不需要采集日志时，可通过关闭采集开关来停止日志采集，以减少资源占用。
  - 日志采集关闭后，ICAgent会停止采集日志，并且在应用运维管理AOM控制台的“日志采集开关”也会同步关闭

## 使用限制

- 支持容器引擎为Docker的CCE集群节点。
- 支持使用Containerd作为容器引擎的CCE集群节点（ICAgent 5.12.130及以上版本）。
- 容器内的日志目录如果已挂载到主机目录上，将无法通过配置容器文件路径方式采集到LTS，只能通过配置节点文件路径方式采集到LTS。
- Docker存储驱动限制：容器文件日志采集目前仅支持overlay2存储驱动，不支持devicemapper作为存储驱动。查看存储驱动类型，请使用如下命令：  
`docker info | grep "Storage Driver"`

## 步骤 1：选择日志流

1. 登录[云日志服务控制台](#)。
2. 在左侧导航栏中，选择“日志接入 > 接入中心”，单击自建软件下的“ServiceStage - 容器应用日志”进行ServiceStage接入配置。  
或在左侧导航栏中，选择“日志接入 > 接入管理”，单击“创建”，在弹出的页面中，单击“ServiceStage - 容器应用日志”进行ServiceStage接入配置。
3. 在选择日志流页面，设置如下参数。
  - a. 选择ServiceStage应用、ServiceStage环境。
  - b. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
  - c. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
4. 单击“下一步：检查依赖项”。

## 步骤 2：检查依赖项

1. 系统自动检查是否存在自定义标识符为**k8s-log-应用ID**的主机组。  
如果不符合要求，需单击“自动修复”按钮进行修复，否则将无法进行下一步操作。
  - **自动修复**：一键帮您完成内容检查项配置。
  - **重新检查**：重新检查依赖项。
2. 单击“下一步：选择主机组（可选）”。

## 步骤 3：选择主机组（可选）

1. 在主机组列表中，默认选择集群所在的主机组，您可以根据需要选择其他已创建的主机组。
2. 单击“下一步：采集配置”。

## 步骤 4：采集配置

采集配置项涉及日志的采集范围、采集方式、格式处理等内容配置，请参考以下步骤完成采集配置。

1. **采集配置名称**：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。
2. **数据源配置**：选择数据源类型，进行对应的数据源配置。支持的数据源类型有容器标准输出、容器文件路径、节点文件路径和K8S事件。

表 4-25 采集配置参数表

| 类型     | 参数配置                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 容器标准输出 | <p>采集集群内指定容器的标准输出日志和标准错误日志，仅支持Stderr和Stdout的日志。采集容器标准输出（stdout）和采集容器标准错误（stderr）两者必须得有一个是开启状态。</p> <ul style="list-style-type: none"><li>• 开启采集容器标准错误（stderr）后，选择采集目标路径：将标准输出和标准错误采集到不同的文件（stdout.log和stderr.log）、将标准输出和标准错误采集到同一个文件（stdout.log）。</li><li>• 被匹配上的容器的标准输出会采集到指定的日志流，原先采集到的AOM的标准输出会停止采集。</li><li>• 允许文件多次采集。（暂不支持Windows场景）<br/>开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见<a href="#">查看ICAgent版本说明</a>。</li></ul> <p>关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。</p>                                                                                                                     |
| 容器文件路径 | <p>采集集群内指定容器内的文件日志。</p> <ul style="list-style-type: none"><li>• 路径配置：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集，详细请参考<a href="#">2</a>。<br/>当CCE集群的工作负载中，已配置容器的挂载路径时，此时路径配置里添加的路径将无效。须将CCE集群页面中的挂载路径删除后，该配置才有效。</li><li>• 支持验证采集路径的正确性，确保正常采集日志。单击“使用路径验证”，输入采集路径和日志文件绝对路径，最多支持添加30个采集路径，单击“确定”，若路径正确，界面提示验证成功。</li><li>• 允许文件多次采集。（暂不支持Windows场景）<br/>开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见<a href="#">查看ICAgent版本说明</a>。</li></ul> <p>关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。</p> <ul style="list-style-type: none"><li>• 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。</li></ul> |

| 类型     | 参数配置                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 节点文件路径 | <p>采集集群内指定节点的文件。</p> <ul style="list-style-type: none"><li>● 路径配置：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。</li><li>● 允许文件多次采集。（暂不支持Windows场景）<br/>开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见<a href="#">查看ICAgent版本说明</a>。<br/>关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。</li><li>● 支持验证采集路径的正确性，确保正常采集日志。单击“使用路径验证”，输入采集路径和日志文件绝对路径，最多支持添加30个采集路径，单击“确定”，若路径正确，界面提示验证成功。</li><li>● 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。</li></ul> |
| K8S事件  | <p>采集K8S集群内的事件日志。无需设置参数，仅支持icagent 5.12.150及以上版本。</p> <p>K8S事件不能重复配置，即一个K8S集群的K8S事件，只能配置接入到一个日志流。</p>                                                                                                                                                                                                                                                                                                                                                                               |

3. 当数据源类型选择**容器标准输出**和**容器文件路径**时，需要设置ServiceStage匹配规则，在下拉框选择对应组件。
4. **结构化解析配置。**  
LTS支持**单行-全文日志**、**多行-全文日志**、**JSON**、**分隔符**、**单行-完全正则**、**多行-完全正则**、**组合解析**多种类型日志解析规则，您可以根据日志内容选择合适的解析规则。日志采集成功后，结构化处理好的日志会发送到指定的日志流上，您就可以对日志进行字段搜索。
- 开启结构化解析配置，详细操作请参考[配置ICAgent结构化解析](#)。

图 4-49 ICAgent 结构化解析配置



- 关闭结构化解析配置，则不会对日志数据进行结构化处理，原始日志格式会发送到指定日志流中，仅可以对日志进行一些常规关键词搜索。
5. **其他配置。**设置采集路径后，您还可以设置日志拆分、采集二进制文件、自定义元数据等。

表 4-26 其他配置

| 名称      | 说明                                                                                                                                                                                                                                                                                              | 示例    |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| 最大目录深度  | 该参数用于控制日志采集路径使用**模糊匹配的文件目录层级，LTS支持设置最大目录层级为20层。<br>例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，最大目录深度设置为5。                                                                                                                                                               | 5     |
| 日志拆分    | 为了防止单条日志过大或被截断丢弃，您可以按文件大小拆分日志。 <ul style="list-style-type: none"><li>开启“日志拆分”，支持自定义设置日志拆分大小，设置范围为500KB-1024KB。日志拆分大小为500KB，即单条日志超过500KB会被拆分为多条采集。例如：日志大小为600KB，被拆分为2条日志采集，第一条500KB，第二条100KB。仅支持单行日志，不支持多行日志。</li><li>不开启“日志拆分”，单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。</li></ul>                         | 开启    |
| 采集二进制文件 | 支持采集以二进制格式存储的日志数据，您可以通过如下命令查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。<br>file -i 文件名 <ul style="list-style-type: none"><li>当日志的文件类型为二进制时，开启“采集二进制文件”，则对匹配接入规则的二进制日志文件进行采集，但仅支持UTF8编码的字符串，非UTF-8编码的字符在LTS控制台页面会显示乱码。</li><li>当日志的文件类型为二进制时，未开启“采集二进制文件”，则对匹配接入规则的二进制日志文件停止采集，开启后即可进行采集。</li></ul> | 开启    |
| 日志文件编码  | 日志文件里字符内容的存储格式，支持UTF-8、GBK文件编码类型，GBK暂不支持Windows场景。请合理设置编码，确保日志内容能被正确读取和解析，避免乱码或数据损坏。 <ul style="list-style-type: none"><li>UTF-8编码是一种变长编码方式，用于表示Unicode字符集。</li><li>GBK全称《汉字内码扩展规范》，中文计算机编码的一种，是ASCII码和GB2312编码的扩展。</li></ul>                                                                   | UTF-8 |
| 采集策略    | ICAgent采集新文件时，您可以设置从文件的末尾或开头开始读取文件。 <ul style="list-style-type: none"><li>增量采集：ICAgent采集新文件时，从文件的末尾开始读。</li><li>全量采集：ICAgent采集新文件时，从文件的开头开始读。</li></ul>                                                                                                                                         | 增量采集  |

| 名称     | 说明                                                                                                                                                                                                                     | 示例 |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 自定义元数据 | <ul style="list-style-type: none"><li>关闭“自定义元数据”，使用ICAgent系统默认配置的字段上报到LTS，不需要用户配置且ICAgent系统不支持配置。</li><li>开启“自定义元数据”，根据用户选择的内置字段和自定义键值增加字段上报到LTS。<br/>系统内置字段：勾选需要设置的内置字段。<br/>自定义键值对：单击“添加”，输入键值key和键值Value。</li></ul> | 开启 |

6. 参考表4-27设置日志格式、日志时间。若开启结构化解析配置则不用设置该参数。

表 4-27 日志采集信息

| 名称   | 说明                                                                                                                                                                                                                          |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日志格式 | <ul style="list-style-type: none"><li>单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。</li><li>多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。</li></ul>                      |
| 日志时间 | <p>系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。</p> <ul style="list-style-type: none"><li>日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。</li><li>日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒。</li><li>采集日志时间限制：系统时间的前后24小时内。</li></ul> |



| 名称    | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | <p>时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。</p> <ul style="list-style-type: none"><li>如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。</li><li>如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。</li></ul> <p>填写示例：</p> <p>YY - year (19)<br/>YYYY - year (2019)<br/>M - month (1)<br/>MM - month (01)<br/>D - day (1)<br/>DD - day (01)<br/>hh - hours (23)<br/>mm - minutes (59)<br/>ss - seconds (59)<br/>SSS - millisecond (999)<br/>hpm - hours (03PM)<br/>h:mmpm - hours:minutes (03:04PM)<br/>h:mm:sspm - hours:minutes:seconds (03:04:05PM)<br/>hh:mm:ss ZZZZ (16:05:06 +0100)<br/>hh:mm:ss ZZZ (16:05:06 CET)<br/>hh:mm:ss ZZ (16:05:06 +01:00)</p> |
| 分行模式  | 日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 正则表达式 | <p>此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。</p> <p>ICAgent只支持RE2风格的正则表达式，详细请参考<a href="#">语法</a>。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

7. 单击“下一步：索引配置”。

## 步骤 5：索引配置

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

- 若您暂时无需通过某字段进行查询或分析，接入时可以直接跳过索引配置，不影响日志采集。待日志接入成功后，再根据业务需求优化索引配置（**仅对新写入的日志生效**），详细操作请参考[创建LTS日志索引](#)。在“索引配置”页面保持默认设置，直接单击“跳过并提交”，界面提示“日志接入成功”。
- 若您需要通过特定字段进行查询或分析，接入时请在“索引配置”页面参考[创建LTS日志索引](#)配置索引。

您可以单击“自动配置”，云日志服务LTS会根据近15分钟的第一条日志内容或常见内置保留字段（例如hostIP、hostName、pathFile）自动生成字段索引。或者手动添加字段，只能添加已结构化的字段。设置完成后，单击“提交”，界面提

示“日志接入成功”。日志接入成功后，您仍可以根据业务需求随时调整索引配置（仅对新写入的日志生效）。

## 步骤 6：完成接入配置

接入成功后会生成一条接入配置信息。

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“修改”重新修改接入配置信息。
- 单击接入配置操作列的“更多 > 标签管理”即可添加标签。
- 单击接入配置操作列的“更多 > 复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。

### 警告

删除接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，且删除后无法恢复，请谨慎操作。

- 若不需要采集日志，关闭“接入配置开关”列的开关。若需要重新采集日志，需要重新开启“接入配置开关”列的开关。

### 警告

关闭接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，请谨慎操作。

- 单击接入配置操作列的“更多 > ICAgent采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。如果没有显示该功能，请参考[设置 ICAgent日志采集开关](#)开启“ICAgent诊断开关”。

## 批量设置多个接入配置

支持同时批量设置多个接入配置，操作简单，不用重复配置即可快速完成多个场景的接入配置。

**步骤1** 在“接入管理”页面，单击“批量创建”，进入配置详情页面。

1. 接入类型：选择ServiceStage-容器应用日志。
2. 规则列表：
  - 在输入框填写接入配置数量，单击“添加接入配置”。
  - 在右侧配置项下方自定义填写“规则名称”，配置项设置完成后，也可以双击左侧接入配置的名称自定义命名。规则名称只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。长度为1-64个字符。
  - 鼠标放在接入配置上面，单击即可直接复制接入配置。
  - 鼠标放在接入配置上面，单击，在弹出的提示框单击“确定”删除接入配置。
3. 配置项：

- 左侧显示接入配置的信息，最多支持添加99个配置。
- 右侧显示配置接入的内容，详细请参考[步骤4：采集配置](#)进行设置。
- 一个接入配置设置完成后，单击“应用于其他接入配置”即可将该接入配置复制到其他接入配置。

**步骤2** 单击“参数检查”，检查成功后，单击“提交”，批量接入设置完成。

例如添加了4个接入配置，批量创建成功后，在“接入管理”下方，就会显示4条接入配置任务。

**步骤3** （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个接入配置开关显示关闭的接入配置，单击“接入配置开启/关闭”，选择“开启”即可批量开启接入配置任务。
- 勾选多个接入配置开关显示开启的接入配置，单击“接入配置开启/关闭”，选择“关闭”。接入配置状态关闭后不会继续采集日志，请谨慎操作。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

## 相关文档

- 日志接入LTS成功后，您可通过云日志服务LTS的搜索与分析功能快速洞察日志数据，请参考[日志搜索与分析（即将下线SQL分析方式）](#)。
- 日志接入LTS成功后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- 如果在安装ICAgent时遇到问题，您可以参考[主机管理常见问题](#)获取解决方案。
- 如果在配置日志接入时遇到问题，您可以参考[日志接入常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志接入、查询日志接入、删除日志接入等，详细请参考[日志接入](#)。
- 为您推荐几种在真实使用场景下的数据采集实践教程。更多信息请参考[日志接入最佳实践](#)。
  - a. 如果是多云或者跨Region采集日志的场景，您可以参考[采集第三方云厂商、互联网数据中心、华为云其他Region云主机日志到LTS](#)。
  - b. 如果需要采集多渠道数据，您可以参考[采集多渠道日志数据到LTS](#)。

### 4.3.4 自建 K8s 应用日志接入 LTS

自建K8s（kubernetes）是开源的一个容器编排引擎，它支持自动化部署、大规模可伸缩、应用容器化管理。用户将自建K8s集群内服务或集群节点特定路径文件的应用日志上报至LTS后，可以对Kubernetes集群内服务日志进行存储和分析。

您可以按照如下步骤完成接入配置。

**步骤1：选择日志流：**选择将日志存储到对应的日志流中，便于不同类型日志分类存储管理。

**步骤2：检查依赖项：**自动检查依赖项是否符合要求。

**步骤3：安装日志采集组件：**在主机上安装ICAgent。

**步骤4：选择主机组（可选）：**选择日志接入的主机范围，主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。可以将需要采集日志的一台或多台主机加入到同一个主机组中，再将该主机组关联至同一个接入配置中。

**步骤5：采集配置：**配置日志的采集路径、采集策略等。

**步骤6：索引配置：**索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。

**步骤7：完成接入配置：**接入成功后，可以在接入列表进行任务的增删改查。

若需要采集多个场景的日志，您可以选择**批量设置多个接入配置**的方式，同时设置多个接入配置。

前提条件

- 请确保已在Kubernetes集群中执行安装Helm v3的命令。
- 请确保Kubernetes集群已配置kubectl。
- 请确保节点的kubelet进程监听10255端口（只读端口，用于获取pod信息）。
- 已创建日志组和日志流。详细请参考[管理日志组](#)和[管理日志流](#)。
- 日志接入前，需要确认开启ICAgent采集开关，请参考[设置LTS日志采集配额和使用量预警](#)。
  - 采集开关默认打开，当您不需要采集日志时，可通过关闭采集开关来停止日志采集，以减少资源占用。
  - 日志采集关闭后，ICAgent会停止采集日志，并且在应用运维管理AOM控制台的“日志采集开关”也会同步关闭

步骤 1：选择日志流

1. 登录[云日志服务控制台](#)。
2. 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“自建k8s - 应用日志”进行自建K8s接入配置。  
或在左侧导航栏中，选择“日志接入 > 接入管理”，单击“创建”，在弹出的页面中，单击“自建k8s - 应用日志”进行自建K8s接入配置。
3. 采集方式：采集到集中日志流和采集到自定义日志流，您可以根据实际情况选择采集方式，推荐您使用采集到集中日志流。

表 4-28 采集方式

| 采集方式     | 说明                                                                                                                                                                                        |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 采集到集中日志流 | <ul style="list-style-type: none"><li>• 优点：相同类型的日志在一个日志流中，方便集中搜索。</li><li>• 缺点：<br/>不同的工作负载的日志结构不一样，采集到一个日志流后无法配置结构化解析，无法使用SQL可视化分析。<br/>单个日志流的写入速率上限是100MB/S，集中采集对于大流量场景有性能瓶颈。</li></ul> |

| 采集方式      | 说明                                                                                                                                                                                   |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 采集到自定义日志流 | <ul style="list-style-type: none"><li>• 优点：<br/>不同的工作负载的日志结构不一样，采集到不同日志流后可以配置结构化解析，使用SQL可视化分析。<br/>多个日志流的写入速率累加起来可以线性扩增，自定义采集对于大流量场景没有性能瓶颈。</li><li>• 缺点：日志流数量较多，管理起来相对繁琐。</li></ul> |

- 若选择“采集到集中日志流”，请执行如下操作步骤：

集中采集日志到一个固定的日志流。Kubernetes集群默认的采集日志流分别为标准输出/错误stdout-{ClusterID}、节点文件hostfile-{ClusterID}、容器文件containerfile-{ClusterID}、K8S事件: event-{ClusterID}。日志流名称会根据ClusterID自动命名，例如：集群ID为Cluster01，则标准输出/错误日志流为stdout-Cluster01。

在一个kubernetes集群下可以创建的采集日志流为标准输出/错误stdout-{ClusterID}、节点文件hostfile-{ClusterID}、容器文件containerfile-{ClusterID})和K8s事件event-{ClusterID}，如果某个日志组下，已创建某种采集日志流，则不会在其他日志组或当前日志组下再创建该日志流。

- 选择采集方式“采集到集中日志流”。
- 输入“集群名称”和“集群ID”。
- 选择“所属日志组”。

当无该日志组时，系统会提示：暂无该日志组，后续操作中，系统将会为您自动创建，创建完成后日志会集中采集到该日志组中。

- 若选择“采集到自定义日志流”，请执行如下操作步骤：

- 选择采集方式“采集到自定义日志流”。
- 输入“集群名称”和“集群ID”。
- 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
- 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。

图 4-50 采集到自定义日志流



当前接入方式

自建k8s - 应用日志

重新选择

将自建k8s的容器标准输出日志、容器文件日志、节点文件日志采集到云日志服务。[了解更多](#)

采集方式

采集到集中日志流

采集到自定义日志流

采集到自定义日志流:

优点:

不同的工作负载的日志结构不一样。采集到不同日志流后可以分别配置结构化解析  
多个日志流的写入速率累加起来可以线性扩增。自定义采集对于大流量场景没有性能瓶颈

缺点:

日志流数量较多，管理起来相对繁琐

集群名称

集群ID

所属日志组

DCS-log-group-1679885600016

C 没有所需日志组? 请点击 [新建](#)

所属日志流

--请选择--

C 没有所需日志流? 请点击 [新建](#)

4. 单击“下一步：检查依赖项”。

步骤 2：检查依赖项

1. 系统自动检查以下三项是否符合要求：
- 存在自定义标识符为**k8s-log-集群ID**的主机组。

- 存在名为**k8s-log-集群ID**的日志组。支持修改日志组的日志存储时间和备注。当选择日志流为**采集到集中日志流**时，会进行该项内容检查。

- 存在系统推荐的集中采集的日志流。支持修改日志流的日志存储时间和备注。当选择日志流为**采集到集中日志流**时，会进行该项内容检查。

如果以上三项中，有任意一项不符合要求，需单击“自动修复”按钮进行修复，否则将无法进行下一步操作。

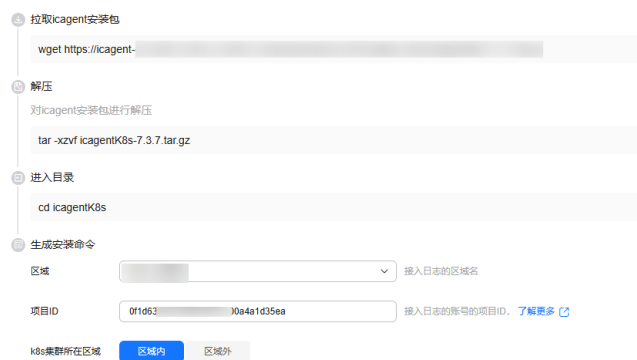
- **自动修复**：一键帮您完成以上三项配置。

- **重新检查**：重新检查依赖项。
2. 单击“下一步：安装日志采集组件”。

步骤 3：安装日志采集组件

在Kubernetes集群中，选择任意一台主机，按照LTS页面的提示执行如下操作步骤：

图 4-51 安装日志采集组件



1. 在LTS的安装日志采集组件页面，获取ICAgent安装包。
2. 在LTS页面上复制解压命令，解压ICAgent安装包。
3. 执行如下命令，进入icagentK8s目录。  
`cd icagentK8s`
4. 在生成安装命令下方，选择接入日志的区域名、接入日志的账号的项目ID。
5. k8s集群所在区域选择“区域内”。
- a. 在安装步骤下方，复制ICAgent安装命令。执行的命令需要填写AK/SK，有两种方式可选择：复制命令时手动替换AK/SK或者直接执行命令时再根据提示替换AK/SK。

图 4-52 安装 IC Agent



- b. 使用PuTTY等远程登录工具，以root用户登录待安装主机，执行复制到的ICAgent安装命令。  
当显示“ICAgent install success”时，表示安装成功，安装成功后，在主机列表下方查看ICAgent状态显示“运行”。  
安装ICAgent成功后，单击“确认安装完毕”。
6. k8s集群所在区域选择“区域外”。
- a. 网络连通性方式支持公网或专线。区域外集群将日志上报到当前区域LTS的网络通路，建议您优先选择专线，更稳定可靠。
  - 若选择“公网”，直接从6.c开始执行。
  - 若选择“专线”，需要从6.b开始执行。
- b. 连通LTS后端方式支持VPCEP或跳板机。在专线连通场景下，默认区域外集群与当前区域LTS网络不通，安装在区域外主机上的ICAgent无法直接访问管理面上报日志的网段。因此需要配置网络打通方案，使用VPCEP或跳板机的方式去连通LTS。
  - 选择“连通LTS后端方式”为“VPCEP”时：



配置VPCEP域名。请您在网络工程师的协助下，在区域外云上配置VPCEP的DNS域名解析规则，将VPCEP的域名解析到指定IP。配置完成后，根据界面提示复制命令，选择一台待采集日志的主机，执行该命令。如果能ping通，表示网络配置连通。

■ 选择“连通LTS后端方式”为“跳板机”时：

- 1) 创建Linux操作系统的ECS弹性云服务器作为跳板机，输入跳板机私有IP。
- 2) 获取AK/SK。填写DC和跳板机连接IP。

DC：自定义主机节点所属数据中心名称，便于分类查看主机。由数字、大小写字母、中划线、下划线组成，并且最多不超过64个字符。

跳板机连接IP：如果跳板机使用EIP方式连通区域外主机，此处填写跳板机弹性IP；如果跳板机与区域外主机使用云专线VPC对等连接方式，此处填写跳板机VPC内网IP，即私有IP。

- c. 在安装步骤下方，复制ICAgent安装命令。执行的命令需要填写AK/SK，有两种方式可选择：复制命令时手动替换AK/SK或者直接执行命令时再根据提示替换AK/SK。
- d. 使用PuTTY等远程登录工具，以root用户登录待安装主机，执行复制到的ICAgent安装命令。

当显示“Install ICAgent success”时，表示安装成功，安装成功后，在主机列表下方查看ICAgent状态显示“运行”。

安装ICAgent成功后，单击“确认安装完毕”。

## 步骤 4：选择主机组（可选）

1. 在主机组列表中，默认选择集群所在的主机组，您可以根据需要选择其他已创建的主机组。
2. 单击“下一步：采集配置”。

## 步骤 5：采集配置

采集配置项涉及日志的采集范围、采集方式、格式处理等内容配置，请参考以下步骤完成采集配置。

1. **采集配置名称**：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。
2. **数据源配置**：选择数据源类型，进行对应的数据源配置。

表 4-29 采集配置参数表

| 类型     | 参数配置                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 容器标准输出 | <p>采集集群内指定容器的标准输出日志和标准错误日志，仅支持Stderr和Stdout的日志。</p> <p>采集容器标准输出（stdout）和采集容器标准错误（stderr）。两者必须得有一个是开启状态。</p> <ul style="list-style-type: none"><li>• 开启后采集容器标准错误（stderr），可以选择采集目标路径：将标准输出和标准错误采集到不同的文件（stdout.log和stderr.log）、将标准输出和标准错误采集到同一个文件（stdout.log）。</li><li>• 被匹配上的容器的标准输出会采集到指定的日志流，原先采集到的AOM的标准输出会停止采集。</li><li>• 允许文件多次采集。（暂不支持Windows场景）<br/>开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见<a href="#">查看ICAgent版本说明</a>。</li></ul> <p>关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。</p> |

| 类型     | 参数配置                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 容器文件路径 | <p>采集集群内指定容器内的文件路径日志。</p> <ul style="list-style-type: none"><li>● <b>路径配置</b>：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。<br/>当CCE集群的工作负载中，已配置容器的挂载路径时，此时路径配置里添加的路径将无效。须将CCE集群页面中的挂载路径删除后，该配置才有效。</li><li>● 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。<br/><b>内置类型</b>为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-，绕接标识为非字母符号，后缀为字母。<br/><b>自定义绕接规则</b>为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\\.d{4}-\\.d{2}-\\.d{2}\\.d{1}.out.log</li><li>● 支持验证采集路径的正确性，确保正常采集日志。单击“使用路径验证”，输入采集路径和日志文件绝对路径，最多支持添加30个采集路径，单击“确定”，若路径正确，界面提示验证成功。</li><li>● 支持验证绕接规则的正确性，确保正常采集日志。单击“使用绕接规则验证”，输入采集文件名、绕接后的文件名、绕接规则，单击“确定”，若绕接规则正确，界面提示验证成功。</li><li>● 允许文件多次采集。（暂不支持Windows场景）<br/>开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见<a href="#">查看ICAgent版本说明</a>。<br/>关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。</li><li>● 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。</li></ul> |

| 类型     | 参数配置                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 节点文件路径 | <p>采集集群内指定节点路径的文件。</p> <ul style="list-style-type: none"><li>● <b>路径配置</b>：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。</li><li>● 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。<br/><b>内置类型</b>为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-，绕接标识为非字母符号，后缀为字母。<br/><b>自定义绕接规则</b>为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\\.d{4}-\\.d{2}-\\.d{2}\\.d{1}.out.log</li><li>● 支持验证采集路径的正确性，确保正常采集日志。单击“使用路径验证”，输入采集路径和日志文件绝对路径，最多支持添加30个采集路径，单击“确定”，若路径正确，界面提示验证成功。</li><li>● 支持验证绕接规则的正确性，确保正常采集日志。单击“使用绕接规则验证”，输入采集文件名、绕接后的文件名、绕接规则，单击“确定”，若绕接规则正确，界面提示验证成功。</li><li>● 允许文件多次采集。（暂不支持Windows场景）<br/>开启“允许文件多次采集”后，同一主机下的同一日志文件支持被采集到多个日志流。该功能依赖ICAgent版本，详见<a href="#">查看ICAgent版本说明</a>。<br/>关闭“允许文件多次采集”后，采集路径不能重复配置，即同一主机下的同一日志文件，即使跨日志流，也只能配置一次。</li><li>● 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。</li></ul> |
| K8S事件  | <p>采集K8S集群内的事件日志。无需设置参数。仅支持icagent 5.12.150及以上版本。</p> <p>K8S事件不能重复配置，即一个K8S集群的K8S事件，只能配置接入到一个日志流。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

3. 当数据源类型选择容器标准输出和容器文件路径时，设置K8s匹配规则，非必选项。  
填写正则匹配规则后，单击校验按钮，支持校验确保正则表达式的正确性。
4. **结构化解析配置。**  
LTS支持**单行-全文日志、多行-全文日志、JSON、分隔符、单行-完全正则、多行-完全正则、组合解析**多种类型日志解析规则，您可以根据日志内容选择合适的解析规则。日志采集成功后，结构化处理好的日志会发送到指定的日志流上，您就可以对日志进行字段搜索。
- 开启结构化解析配置，详细操作请参考[配置ICAgent结构化解析](#)。

图 4-53 ICAgent 结构化解析配置



- 关闭结构化解析配置，则不会对日志数据进行结构化处理，原始日志格式会发送到指定日志流中，仅可以对日志进行一些常规关键词搜索。
5. **其他配置。**设置采集路径后，您还可以设置日志拆分、采集二进制文件、自定义元数据等。

表 4-30 其他配置

| 名称      | 说明                                                                                                                                                                                                                                                                                               | 示例 |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 最大目录深度  | 该参数用于控制日志采集路径使用**模糊匹配的文件目录层级，LTS支持设置最大目录层级为20层。<br>例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，最大目录深度设置为5。<br>如果需要采集容器文件路径，需要ICAgent升级到7.3.3版本以上才支持**模糊匹配。                                                                                                                | 5  |
| 日志拆分    | 为了防止单条日志过大或被截断丢弃，您可以按文件大小拆分日志。 <ul style="list-style-type: none"><li>开启“日志拆分”，支持自定义设置日志拆分大小，设置范围为500KB-1024KB。日志拆分大小为500KB，即单条日志超过500KB会被拆分为多条采集。例如：日志大小为600KB，被拆分为2条日志采集，第一条500KB，第二条100KB。仅支持单行日志，不支持多行日志。</li><li>不开启“日志拆分”，单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。</li></ul>                          | 开启 |
| 采集二进制文件 | 支持采集以二进制格式存储的日志数据，您可以通过如下命令查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。<br>file -i 文件名 <ul style="list-style-type: none"><li>当日志的文件类型为二进制时，开启“采集二进制文件”，则对匹配接入规则的二进制日志文件进行采集，但仅支持UTF-8编码的字符串，非UTF-8编码的字符在LTS控制台页面会显示乱码。</li><li>当日志的文件类型为二进制时，未开启“采集二进制文件”，则对匹配接入规则的二进制日志文件停止采集，开启后即可进行采集。</li></ul> | 开启 |

| 名称     | 说明                                                                                                                                                                                                                                   | 示例    |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| 日志文件编码 | <p>日志文件里字符内容的存储格式，支持UTF-8、GBK文件编码类型，GBK暂不支持Windows场景。请合理设置编码，确保日志内容能被正确读取和解析，避免乱码或数据损坏。</p> <ul style="list-style-type: none"><li>UTF-8编码是一种变长编码方式，用于表示Unicode字符集。</li><li>GBK全称《汉字内码扩展规范》，中文计算机编码的一种，是ASCII码和GB2312编码的扩展。</li></ul> | UTF-8 |
| 采集策略   | <p>ICAgent采集新文件时，您可以设置从文件的末尾或开头开始读取文件。</p> <ul style="list-style-type: none"><li>增量采集：ICAgent采集新文件时，从文件的末尾开始读。</li><li>全量采集：ICAgent采集新文件时，从文件的开头开始读。</li></ul>                                                                       | 增量采集  |
| 自定义元数据 | <ul style="list-style-type: none"><li>关闭“自定义元数据”，使用ICAgent系统默认配置的字段上报到LTS，不需要用户配置且ICAgent系统不支持配置。</li><li>开启“自定义元数据”，根据用户选择的内置字段和自定义键值增加字段上报到LTS。<br/>系统内置字段：勾选需要设置的内置字段。<br/>自定义键值对：单击“添加”，输入键值key和键值Value。</li></ul>               | 开启    |

6. 参考表4-31配置日志格式、日志时间。若开启结构化解析配置则不用设置该参数。

表 4-31 日志采集信息

| 名称   | 说明                                                                                                                                                                                                                          |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日志格式 | <ul style="list-style-type: none"><li>单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。</li><li>多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。</li></ul>                      |
| 日志时间 | <p>系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。</p> <ul style="list-style-type: none"><li>日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。</li><li>日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒。</li><li>采集日志时间限制：系统时间的前后24小时内。</li></ul> |

| 名称    | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | <p>时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。</p> <ul style="list-style-type: none"><li>如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。</li><li>如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。</li></ul> <p>如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。</p> <p>填写示例：</p> <p>YY - year (19)<br/>YYYY - year (2019)<br/>M - month (1)<br/>MM - month (01)<br/>D - day (1)<br/>DD - day (01)<br/>hh - hours (23)<br/>mm - minutes (59)<br/>ss - seconds (59)<br/>SSS - millisecond ( 999 )<br/>hpm - hours (03PM)<br/>h:mmpm - hours:minutes (03:04PM)<br/>h:mm:sspm - hours:minutes:seconds (03:04:05PM)<br/>hh:mm:ss ZZZZ (16:05:06 +0100)<br/>hh:mm:ss ZZZ (16:05:06 CET)<br/>hh:mm:ss ZZ (16:05:06 +01:00)</p> |
| 分行模式  | 日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 正则表达式 | <p>此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。</p> <p>时间通配和正则表达式均是从每行日志的开头进行严格匹配，如果匹配不上，则会默认使用系统时间上报，这样可能会和文件内容中的时间不一致。<b>如果没有特殊需求，建议使用单行日志-系统时间模式即可。</b></p> <p>ICAgent只支持RE2风格的正则表达式，详细请参考<a href="#">语法</a>。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

#### 7. 单击“下一步：索引配置”。

### 步骤 6：索引配置

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

- 若您暂时无需通过某字段进行查询或分析，接入时可以直接跳过索引配置，不影响日志采集。待日志接入成功后，再根据业务需求优化索引配置（**仅对新写入的日志生效**），详细操作请参考[创建LTS日志索引](#)。在“索引配置”页面保持默认设置，直接单击“跳过并提交”，界面提示“日志接入成功”。
- 若您需要通过特定字段进行查询或分析，接入时请在“索引配置”页面参考[创建LTS日志索引](#)配置索引。



您可以单击“自动配置”，云日志服务LTS会根据近15分钟的第一条日志内容或常见内置保留字段（例如hostIP、hostName、pathFile）自动生成字段索引。或者手动添加字段，只能添加已结构化的字段。设置完成后，单击“提交”，界面提示“日志接入成功”。日志接入成功后，您仍可以根据业务需求随时调整索引配置（仅对新写入的日志生效）。

## 步骤 7：完成接入配置

接入成功后会生成一条接入配置信息。

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“修改”重新修改接入配置信息。
- 单击接入配置操作列的“更多 > 标签管理”即可添加标签。
- 单击接入配置操作列的“更多 > 复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。



### 警告

删除接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，且删除后无法恢复，请谨慎操作。

- 若不需要采集日志，关闭“接入配置开关”列的开关。若需要重新采集日志，需要重新开启“接入配置开关”列的开关。



### 警告

关闭接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，请谨慎操作。

- 单击接入配置操作列的“更多 > ICAgent采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。如果没有显示该功能，请参考[设置 ICAgent日志采集开关](#)开启“ICAgent诊断开关”。

## 批量设置多个接入配置

支持同时批量设置多个接入配置，操作简单，不用重复配置即可快速完成多个场景的接入配置。

**步骤1** 在“接入管理”页面，单击“批量创建”，进入配置详情页面。

1. 接入类型：选择自建k8s - 应用日志。
2. 规则列表：
  - 在输入框填写接入配置数量，单击“添加接入配置”。
  - 在右侧配置项下方自定义填写“规则名称”，配置项设置完成后，也可以双击左侧接入配置的名称自定义命名。规则名称只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。长度为1-64个字符。
  - 鼠标放在接入配置上面，单击即可直接复制接入配置。

- 鼠标放在接入配置上面，单击，在弹出的提示框单击“确定”删除接入配置。

### 3. 配置项：

- 左侧显示接入配置的信息，最多支持添加99个配置。
- 右侧显示配置接入的内容，详细请参考[步骤5：采集配置](#)进行设置。
- 一个接入配置设置完成后，单击“应用于其他接入配置”即可将该接入配置复制到其他接入配置。

**步骤2** 单击“参数检查”，检查成功后，单击“提交”，批量接入设置完成。

例如添加了4个接入配置，批量创建成功后，在“接入管理”下方，就会显示4条接入配置任务。

**步骤3** （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个接入配置开关显示关闭的接入配置，单击“接入配置开启/关闭”，选择“开启”即可批量开启接入配置任务。
- 勾选多个接入配置开关显示开启的接入配置，单击“接入配置开启/关闭”，选择“关闭”。接入配置状态关闭后不会继续采集日志，请谨慎操作。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

## 相关文档

- 日志接入LTS成功后，您可通过云日志服务LTS的搜索与分析功能快速洞察日志数据，请参考[日志搜索与分析（即将下线SQL分析方式）](#)。
- 日志接入LTS成功后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- 如果在安装ICAgent时遇到问题，您可以参考[主机管理常见问题](#)获取解决方案。
- 如果在配置日志接入时遇到问题，您可以参考[日志接入常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志接入、查询日志接入、删除日志接入等，详细请参考[日志接入](#)。
- 为您推荐几种在真实使用场景下的数据采集实践教程。更多信息请参考[日志接入最佳实践](#)。
  - a. 如果是多云或者跨Region采集日志的场景，您可以参考[采集第三方云厂商、互联网数据中心、华为云其他Region云主机日志到LTS](#)。
  - b. 如果需要采集多渠道数据，您可以参考[采集多渠道日志数据到LTS](#)。

## 4.4 配置 ICAgent 结构化解析

云日志服务LTS支持通过ICAgent结构化解析对日志进行结构化处理。在创建日志接入时设置ICAgent结构化解析规则，支持单行-全文日志、多行-全文日志、JSON、分隔符、单行-完全正则、多行-完全正则、组合解析，您可以根据日志内容选择合适的解析规则。日志采集成功后，将结构化解析后日志发送到指定的日志流上，您就可以对日志进行搜索与分析。

ICAgent只支持RE2风格的正则表达式，详细请参考[语法](#)。

配置结构化解析

支持以下日志结构化解析规则：

表 4-32 解析规则类型

| 类型      | 说明                                                                                                                                              |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| 单行-全文日志 | 使用换行符\n作为一条日志的结束符，并将每一条日志保存在一个默认的键值content中。                                                                                                    |
| 多行-全文日志 | 适用于一条完整的日志数据跨占多行（例如Java程序日志）的原始日志。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。                          |
| JSON    | 适用于日志文本中每行内容为一个原始日志，且每条日志可按JSON解析规则提取为多个key-value键值对的日志解析模式。                                                                                    |
| 分隔符     | 适用于日志文本中每行内容为一个原始日志，且每条日志可根据指定的分隔符提取为多个key-value键值对的日志解析模式。                                                                                     |
| 单行-完全正则 | 适用于日志文本中每行内容为一个原始日志，且每条日志可按正则表达式提取为多个key-value键值对的日志解析模式。                                                                                       |
| 多行-完全正则 | 适用于日志文本中一条完整的日志数据跨占多行（例如Java程序日志），且可按正则表达式提取为多个key-value键值对的日志解析模式。                                                                             |
| 组合解析    | 当您的日志结构太过复杂，涉及多种解析模式，单种解析模式（如完整正则模式、JSON模式等）无法满足日志解析需求时，您可以使用组合解析规则解析日志，在插件配置下方输入插件语法（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插件配置，ICAgent会根据处理配置顺序逐一执行。 |

单行-全文日志

单行-全文日志解析规则是使用换行符\n作为一条日志的结束符，并将每一条日志保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。

设置操作：

1. 选择单行-全文日志。
2. 日志过滤默认关闭。如果需要从海量的日志数据中筛选出符合特定条件的记录，开启日志过滤开关，参考如下操作添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。

图 4-54 日志过滤规则



- 设置白名单规则。添加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。
  - i. 在白名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式）。新增多条白名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意白名单就可以被采集。单行/多行全文模式下，默认使用content作为全文的键{key}名。  
例如只采集日志中包含hello的日志行，可配置采集规则为`.*hello.*`
  - ii. 单击操作列的“校验”，在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。

图 4-55 校验

- iii. 校验成功后，单击“确定”或“关闭”退出校验页面。
- 设置黑名单规则。添加的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。
  - i. 在黑名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式）。新增多条黑名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意黑名单就可以被排除。单行/多行全文模式下，默认使用content作为全文的键{key}名。  
例如不采集日志中包含hello的日志行，可配置采集规则为`.*hello.*`
  - ii. 单击操作列的“校验”，在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。
  - iii. 校验成功后，单击“确定”或“关闭”退出校验页面。

### 样例展示：

在结构化解析配置中选择“单行-全文日志”模式，日志上报到LTS后，原始日志样例如下所示，经结构化处理后的日志数据上报到LTS的展示效果如[图4-56](#)所示。

- 原始日志样例：

```
[2025-06-05_09:53:34] [WARNING] [MODULE_Mf0s2] - iZ4:f=t.+CU)dSPGRU<1r^IGt!,8SD
ibP!wd.ey5S19V.yZRVD_3-GCX%c(h
[2025-06-05_09:53:34] [WARNING] [MODULE_yPRzp] - ,+iaHEYx+(H-w7g.19s13XSK%ksNe4
[2025-06-05_09:53:34] [INFO] [MODULE_TfNKq] -):21:1^V)G%)X8A8zPE!iC$_m<z3H^
KDhA7(W_!yeXk1)g3tm%c,m%!2!OeH
!WwY4dIMBAz-U.ph@r!RxPvFNYKCyh
[2025-06-05_09:53:34] [ERROR] [MODULE_naV0b] - fR5PxhHm(A5s5co-c)!PD%mWer37ww
zZ3@(^vC#CELo;(BR=f_AhzJADlpR@
V;ZLg3c)N,q-K5t,*ke7jPnY)s#5i2
[2025-06-05_09:53:34] [INFO] [MODULE_5fMUb] - 6i%B0qSl,yDP3.o)C&U+dE%-Sabony
```

```
G.7Pfr7xcx/NrX3SV%*ZVWgW+CGD;(
-UHB4ckm(=OQ,rx8zeYFGNtcK/Z)x
```

- 结果展示：

图 4-56 单行日志上报结果



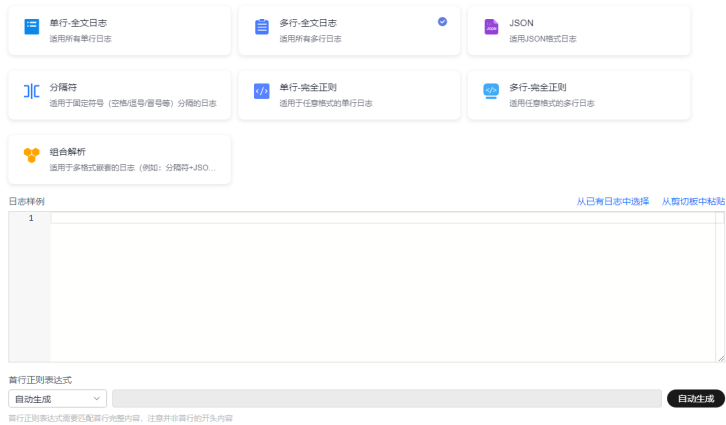
## 多行-全文日志

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-全文日志解析规则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。

### 设置操作：

1. 选择多行-全文日志。
2. 从“从已有日志中选择”或“从剪切板中粘贴”选择日志样例。
  - 从已有日志中选择：单击“从已有日志中选择”，在弹出框中通过选择不同时间段筛选日志，根据业务需求选择待操作的日志，单击“确定”。
  - 从剪切板中粘贴：单击“从剪切板中粘贴”，可将您剪切的日志内容复制到示例日志框中。



3. 在“首行正则表达式”下方支持自动生成或手动输入正则表达式。首行正则表达式需要匹配首行完整内容，注意并非首行的开头内容。

图 4-57 首行正则表达式



4. 日志过滤默认关闭。如果需要从海量的日志数据中筛选出符合特定条件的记录，开启日志过滤开关，参考如下操作添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。

图 4-58 日志过滤规则



- 设置白名单规则。添加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。
- i. 在白名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式）。新增多条白名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意白名单就可以被采集。单行/多行全文模式下，默认使用content作为全文的键{key}名。  
例如只采集日志中包含hello的日志行，可配置采集规则为.\*hello.\*
- ii. 单击操作列的“校验”，在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。

图 4-59 校验



- iii. 校验成功后，单击“确定”或“关闭”退出校验页面。
- 设置黑名单规则。添加的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。
- i. 在黑名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式）。新增多条黑名单过滤规则时，支持选择And或Or的关系，即全部



满足或满足任意黑名单就可以被排除。单行/多行全文模式下，默认使用content作为全文的键{key}名。

例如不采集日志中包含hello的日志行，可配置采集规则为.\*hello.\*

- ii. 单击操作列的“校验”，在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。
- iii. 校验成功后，单击“确定”或“关闭”退出校验页面。

样例展示：

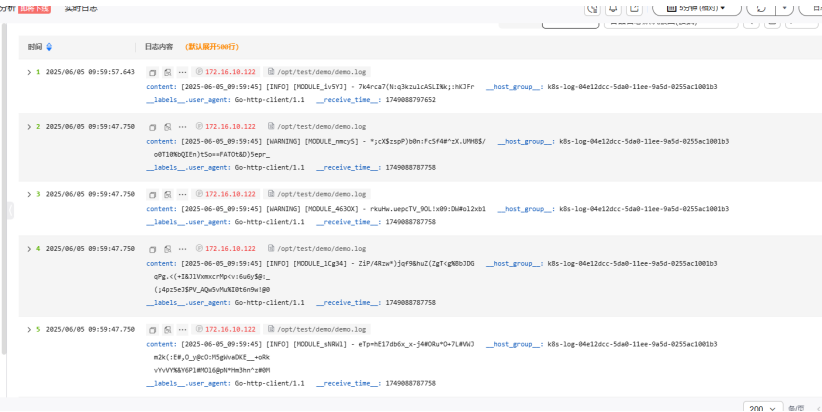
在结构化解析配置中选择“多行-全文日志”模式，日志上报到LTS后，原始日志样例如下所示，经结构化处理后的日志数据上报到LTS的展示效果如图4-60所示。

- 原始日志样例：

```
[2025-06-05_09:59:45] [INFO] [MODULE_sNRWl] - eTp=hE17db6x_x-j4#ORu*O+7L#VWJm2k(:E#,O_y@cO:M5gWvaDKE__+oRk
vYvVY%&Y6Pl#MOl6@pN*Hm3hn^z#0M
[2025-06-05_09:59:45] [INFO] [MODULE_lCg34] - ZiP/4Rzw*)jqf9&huZ(ZgT<g%BbJDG
qPg.<(+l&JlVxmxcRmp<v:6u6y$@:_
(;4pz5eJ$PV_AQw5vMu%l0t6n9w!@0
[2025-06-05_09:59:45] [WARNING] [MODULE_463OX] - rkuHw.uepcTV_9OL!x09:DW#ol2xb1
[2025-06-05_09:59:45] [WARNING] [MODULE_nmcyS] - *;cX$zspP)b0n:FcSf4#^zX.UMH8$/
o0T10%bQlEn)tSo==FATOt&D)5epr_
[2025-06-05_09:59:45] [INFO] [MODULE_iv5YJ] - 7k4rca7(N;q3kzulcASLI%k;hKJFr
```

首行正则表达式：  
^\[d{4}-d{2}-d{2}\.d{2}:d{2}:d{2}\]
- 结果展示

图 4-60 多行日志上报结果



JSON

JSON解析规则适用于日志文本中每行内容为一个原始日志，且每条日志可按JSON解析规则提取为多个key-value键值对的日志解析模式。若不需要提取key-value，可以参考多行-全文日志配置。

设置操作：

1. 选择JSON格式。
2. 日志过滤默认关闭。如果需要从海量的日志数据中筛选出符合特定条件的记录，开启日志过滤开关，参考如下操作添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。



图 4-61 日志过滤规则



- 设置白名单规则。添加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。
  - i. 在白名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式），Key值为日志字段名称。  
新增多条白名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意白名单就可以被采集。  
例如只采集日志中包含hello的日志行，字段值填写hello，过滤规则为`.*hello.*`
  - ii. 单击操作列的“校验”，在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。

图 4-62 校验



- iii. 校验成功后，单击“确定”或“关闭”退出校验页面。
- 设置黑名单规则。添加的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。
  - i. 在黑名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式），Key值为日志字段名称。  
新增多条黑名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意黑名单就可以被排除。  
例如不采集日志中包含hello的日志行，字段值填写hello，过滤规则为`.*hello.*`
  - ii. 单击操作列的“校验”在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。
  - iii. 校验成功后，单击“确定”或“关闭”退出校验页面。
- 3. 上传原始日志。  
打开上传原始日志开关后，原始日志将作为content字段的值上传到日志服务。
- 4. 上传解析失败日志。  
打开上传解析失败日志开关后，原始日志将作为\_content\_parse\_fail\_字段的值上传到日志服务。

5. 关于开启或关闭“上传原始日志”和“上传解析失败日志”时，日志上报参考如下。

图 4-63 结构化解析



表 4-33 日志上报说明

| 参数                                                                              | 日志上报说明                                                                                                                           |
|---------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>开启“上传原始日志”</li><li>开启“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>解析成功：上报解析后日志和原始日志content字段。</li><li>解析失败：上报原始日志content字段和_content_parse_fail_字段。</li></ul> |
| <ul style="list-style-type: none"><li>开启“上传原始日志”</li><li>关闭“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>解析成功：上报解析后日志和原始日志content字段。</li><li>解析失败：上报原始日志content字段。</li></ul>                        |
| <ul style="list-style-type: none"><li>关闭“上传原始日志”</li><li>开启“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>解析成功：上报解析后日志。</li><li>解析失败：上报_content_parse_fail_字段。</li></ul>                             |
| <ul style="list-style-type: none"><li>关闭“上传原始日志”</li><li>关闭“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>解析成功：上报解析后日志。</li><li>解析失败：仅上报内置字段和label字段。</li></ul>                                      |

6. 自定义日志时间。  
开启后可指定某一字段作为日志时间，或关闭此项使用日志被采集时间作为日志时间。
7. json解析层数。增加json解析层数配置，取值范围为1~4，只能整数，默认值为1。  
将json格式日志的字段展开，例如原始日志为{"key1":{"key2":"value"}}，解析1层日志为：{"key1":{"key2":"value"}}，解析2层日志为：{"key1.key2":"value"}。
8. json字符串解析：默认关闭。开启后支持解析转义后的json字符串为json对象，例如{"key1":{"key2":"value"}}解析为key1.key2:value。

样例展示：

在结构化解析配置中选择“JSON”模式，日志上报到LTS后，原始日志样例如下所示，经结构化处理后的日志数据上报到LTS的展示效果如图4-64所示。

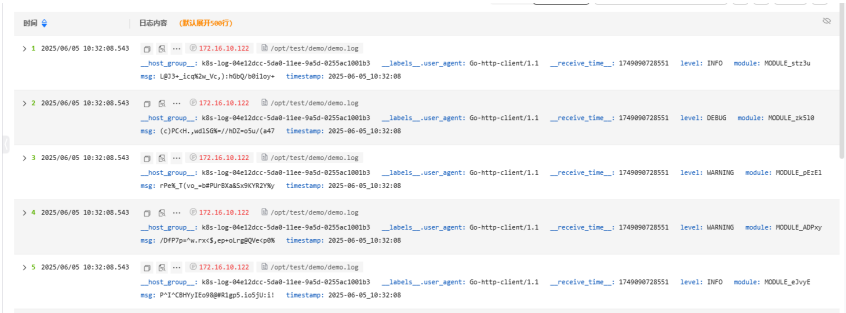
- 原始日志样例：

```
{
 "timestamp": "2025-06-05_10:32:08",
 "level": "INFO",
 "module": "MODULE_eJvyE",
 "msg": "P^I^CBHYlEo98@#R1gp5.io5jU:!!"
}
{"timestamp": "2025-06-05_10:32:08",
"level": "WARNING",
"module": "MODULE_ADpXy",
"msg": "/DfP7p=^w.rx<$,ep+oLrg@QVe<p0%"}
{"timestamp": "2025-06-05_10:32:08",
"level": "WARNING",
"module": "MODULE_pEzEl",
"msg": "rPe
```

```
%_T(vo_=b#PUrBXa&Sx9KYR2Y%y"}
{"timestamp":"2025-06-05_10:32:08","level":"DEBUG","module":"MODULE_zk5l0","msg":"(c)PC<H,wld
SG%=/hDZ=o5u/(a47"}
{"timestamp":"2025-06-05_10:32:08","level":"INFO","module":"MODULE_stz3u","msg":"L@J3+_icq
%2w_Vc_)hGbQ/b0i1oy+"}
```

- 结果展示：

图 4-64 JSON 日志上报结果



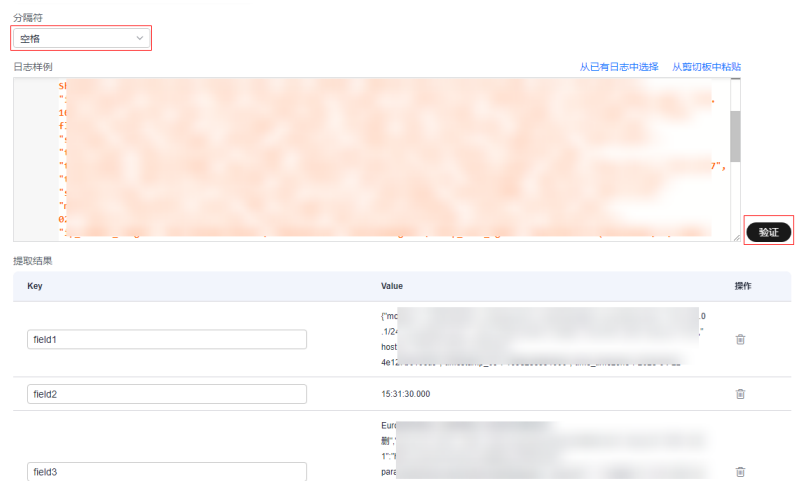
## 分隔符

分隔符解析规则适用于日志文本中每行内容为一个原始日志，且每条日志可根据指定的分隔符提取为多个key-value键值对的日志解析模式。若不需要提取key-value，可以参考[多行-全文日志](#)配置。

### 设置操作：

1. 选择分隔符。
2. 根据原始日志内容选择分隔符，或自定义其他需要的特殊字符作为分隔符。
3. 从“从已有日志中选择”或“从剪切板中粘贴”选择日志样例，单击“验证”，在提取结果下方查看结果。
  - 从已有日志中选择：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
  - 从剪切板中粘贴：单击“从剪切板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。

图 4-65 分隔符



4. 日志过滤默认关闭。如果需要从海量的日志数据中筛选出符合特定条件的记录，开启日志过滤开关，参考如下操作添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。

图 4-66 日志过滤规则



- 设置白名单规则。添加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。

i. 在白名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式），Key值为日志字段名称。

新增多条白名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意白名单就可以被采集。

例如只采集日志中包含hello的日志行，字段值填写hello，过滤规则为.\*hello.\*

ii. 单击操作列的“校验”，在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。

图 4-67 校验



- iii. 校验成功后，单击“确定”或“关闭”退出校验页面。

- 设置黑名单规则。添加的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。

i. 在黑名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式），Key值为日志字段名称。

新增多条黑名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意黑名单就可以被排除。

例如不采集日志中包含hello的日志行，字段值填写hello，过滤规则为.\*hello.\*

ii. 单击操作列的“校验”在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。

iii. 校验成功后，单击“确定”或“关闭”退出校验页面。

5. 上传原始日志。  
打开上传原始日志开关后，原始日志将作为content字段的值上传到日志服务。
6. 上传解析失败日志。  
打开上传解析失败日志开关后，原始日志将作为\_content\_parse\_fail\_字段的值上传到日志服务。
7. 关于开启或关闭“上传原始日志”和“上传解析失败日志”时，日志上报参考如下。

图 4-68 结构化解析



表 4-34 日志上报说明

| 参数                                                                              | 日志上报说明                                                                                                                           |
|---------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>开启“上传原始日志”</li><li>开启“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>解析成功：上报解析后日志和原始日志content字段。</li><li>解析失败：上报原始日志content字段和_content_parse_fail_字段。</li></ul> |
| <ul style="list-style-type: none"><li>开启“上传原始日志”</li><li>关闭“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>解析成功：上报解析后日志和原始日志content字段。</li><li>解析失败：上报原始日志content字段。</li></ul>                        |
| <ul style="list-style-type: none"><li>关闭“上传原始日志”</li><li>开启“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>解析成功：上报解析后日志。</li><li>解析失败：上报_content_parse_fail_字段。</li></ul>                             |
| <ul style="list-style-type: none"><li>关闭“上传原始日志”</li><li>关闭“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>解析成功：上报解析后日志。</li><li>解析失败：仅上报内置字段和label字段。</li></ul>                                      |

8. 自定义日志时间。  
开启后可指定某一字段作为日志时间，或关闭此项使用日志被采集时间作为日志时间。

样例展示：

在结构化解析配置中选择“分隔符”模式，日志上报到LTS后，原始日志样例如下所示，经结构化处理后的日志数据上报到LTS的展示效果如图4-69所示。

- 原始日志样例：
- ```
[2025-06-05_10:04:26] [WARNING] [MODULE_vlQR1] - _4xi&#x2020;M.VHv1_L#CQzas-nC
[2025-06-05_10:04:26] [WARNING] [MODULE_RzMii] - Bdp3_*Eryun:pluL-BdU#y=MgN++7
[2025-06-05_10:04:26] [DEBUG] [MODULE_q9161] - oxUZ#q#xn!oc5WS-<98IB*aH,:FLoj
[2025-06-05_10:04:26] [DEBUG] [MODULE_Ze3vY] - L-g;ZQ%*n/ORiFSYM:#T)z;XxKjpVv
[2025-06-05_10:04:26] [DEBUG] [MODULE_5ZCwd] - e)!6:CZk/MufokE)A5Kr<rBf9o=9x;
```

在日志样例下方输入日志，选择空格分隔符，单击“验证”，提取key-value键值对，根据需要修改key值。



- 结果展示：

图 4-69 分隔符日志上报结果



单行-完全正则

单行-完全正则解析规则适用于日志文本中每行内容为一个原始日志，且每条日志可按正则表达式提取为多个key-value键值对的日志解析模式。若不需要提取key-value，可以参考[多行-全文日志](#)配置。

设置操作：

1. 选择单行-完全正则。
2. 从“从已有日志中选择”或“从剪贴板中粘贴”选择日志样例。
 - 从已有日志中选择：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
 - 从剪贴板中粘贴：单击“从剪贴板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。
3. 在“提取正则表达式”下方输入要提取日志的正则表达式，单击“验证”，在提取结果下方查看结果。

或者单击“正则表达式自动生成”，在新打开的页面根据日志样例提取字段，输入key值，单击“确定”即可自动生成正则表达式，设置完成后，单击“确定”。

图 4-70 提取正则表达式

4. 日志过滤默认关闭。如果需要从海量的日志数据中筛选出符合特定条件的记录，开启日志过滤开关，参考如下操作添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。

图 4-71 日志过滤规则

白名单规则	
通过白名单功能可以限制或禁止指定IP地址的访问。白名单规则分为两种模式，由系统提供默认规则，并支持通过配置规则来定制规则。默认禁止上互联网访问的IP为全公网地址，如： 禁止公网地址访问：禁止公网地址访问公网地址和公网地址，禁止公网地址访问公网地址和公网地址，配置规则名称为："deny"	
Key	访问策略（访问策略ID）
策略	删除
黑名单规则	
通过黑名单功能可以限制或禁止指定IP地址的访问。黑名单规则分为两种模式，由系统提供默认规则，并支持通过配置规则来定制规则。默认禁止上互联网访问的IP为全公网地址，如： 禁止公网地址访问：禁止公网地址访问公网地址和公网地址，禁止公网地址访问公网地址和公网地址，配置规则名称为："deny"	
Key	访问策略（访问策略ID）
策略	删除

- 设置白名单规则。添加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。
 - i. 在白名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式），Key值为日志字段名称。

新增多条白名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意白名单就可以被采集。

例如只采集日志中包含hello的日志行，字段值填写hello，过滤规则为.*hello.*
 - ii. 单击操作列的“校验”，在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。

图 4-72 校验

校验

字段值

hello

过滤规则 (正则表达式)

.*hello.*

校验成功

校验

确定

关闭

- iii. 校验成功后，单击“确定”或“关闭”退出校验页面。
- 设置黑名单规则。添加的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。

- i.

在黑名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式），Key值为日志字段名称。

新增多条黑名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意黑名单就可以被排除。

例如不采集日志中包含hello的日志行，字段值填写hello，过滤规则为.*hello.*

ii.

单击操作列的“校验”在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。

iii.

校验成功后，单击“确定”或“关闭”退出校验页面。
5. 上传原始日志。
打开上传原始日志开关后，原始日志将作为content字段的值上传到日志服务。
6. 上传解析失败日志。
打开上传解析失败日志开关后，原始日志将作为_content_parse_fail_字段的值上传到日志服务。
7. 关于开启或关闭“上传原始日志”和“上传解析失败日志”时，日志上报参考如下。

图 4-73 结构化解析



表 4-35 日志上报说明

参数	日志上报说明
• 开启“上传原始日志” • 开启“上传解析失败日志”	• 解析成功：上报解析后日志和原始日志content字段。 • 解析失败：上报原始日志content字段和_content_parse_fail_字段。
• 开启“上传原始日志” • 关闭“上传解析失败日志”	• 解析成功：上报解析后日志和原始日志content字段。 • 解析失败：上报原始日志content字段。
• 关闭“上传原始日志” • 开启“上传解析失败日志”	• 解析成功：上报解析后日志。 • 解析失败：上报_content_parse_fail_字段。
• 关闭“上传原始日志” • 关闭“上传解析失败日志”	• 解析成功：上报解析后日志。 • 解析失败：仅上报内置字段和label字段。

8. 自定义日志时间。
开启后可指定某一字段作为日志时间，或关闭此项使用日志被采集时间作为日志时间。

样例展示：

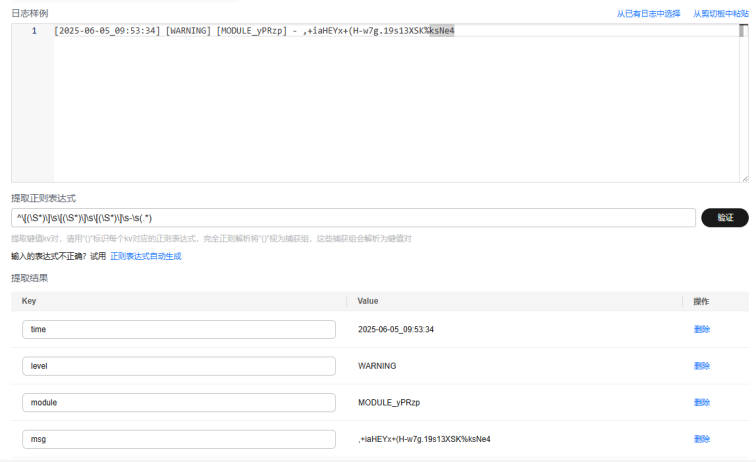
在结构化解析配置中选择“单行-完全正则”模式，日志上报到LTS后，原始日志样例如下所示，经结构化处理后的日志数据上报到LTS的展示效果如图4-74所示。

- 原始日志样例：

```
[2025-06-05_10:10:45] [DEBUG] [MODULE_qqQeu] - +XPLj#eMaBGoL.kF*e@E7OcWzihN=F
[2025-06-05_10:10:45] [WARNING] [MODULE_6Fjpv] - 60QkLDc-53A^<!e_FJ#FW9)x0C00%7
[2025-06-05_10:10:45] [ERROR] [MODULE_mK3lb] - J$7DqL.dR&7^CG(qPSjY2vprb+2nfm
[2025-06-05_10:10:45] [INFO] [MODULE_73x94] - rs3yPQ!(U9^$zZQW8U1-3YQRu,3;@
[2025-06-05_10:10:45] [WARNING] [MODULE_jczuV] - Wsc9Q@L%7itQF3SUallKED3vQS1&E)
```

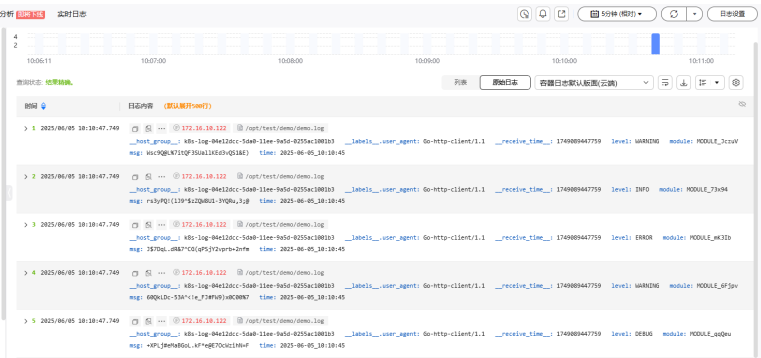
输入提取正则表达式，单击“验证”，提取key-value键值对，根据日志内容修改key值方便识别。

^(\\(S*)\\)\\s\\(\\(S*)\\)\\s\\(\\(S*)\\)\\s\\(S*)(.*)



- 结果展示：

图 4-74 单行-完全正则日志上报结果



多行-完全正则

多行-完全正则解析规则适用于日志文本中一条完整的日志数据跨占多行（例如Java程序日志），且可按正则表达式提取为多个key-value键值对的日志解析模式。若不需要提取key-value，可以参考多行-全文日志配置。配置多行-完全正则模式时，您需要先输入日志样例，使用首行正则表达式匹配首行完整内容，自定义提取正则表达式。

设置操作：

- 选择多行-完全正则。

2. 从“从已有日志中选择”或“从剪切板中粘贴”选择日志样例。
 - 从已有日志中选择：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
 - 从剪切板中粘贴：单击“从剪切板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。
3. 在“首行正则表达式”下方支持自动生成或手动输入正则表达式。首行正则表达式需要匹配首行完整内容，注意并非首行的开头内容。

“首行正则表达式”用于识别多行日志的行首，例如以下日志示例：

2024-10-11 10:59:07.000 a.log:1 level:warn
no.1 log
2024-10-11 10:59:17.000 a.log:2 level:warn
no.2 log

完整的多行是：

2024-10-11 10:59:07.000 a.log:1 level:warn
no.1 log

首行则是：

2024-10-11 10:59:07.000 a.log:1 level:warn

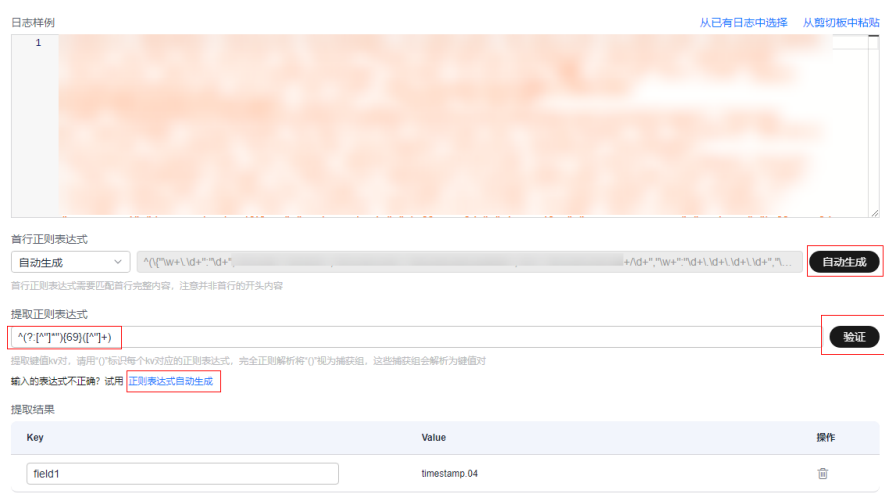
首行正则示例：`^\d{4}-\d{2}-\d{2}\d{3}:\d{2}:\d{2}\.\d{3}$`，由于每个首行的日期是唯一的，因此可以根据日期来生成首行正则。
4. 在“提取正则表达式”下方输入要提取日志的正则表达式，单击“验证”，在提取结果下方查看结果。

或者单击“正则表达式自动生成”，在新打开的页面根据日志样例提取字段，输入key值，单击“确定”即可自动生成正则表达式，设置完成后，单击“确定”。

提取结果显示的是“提取正则表达式”执行的结果，而不是“首行正则表达式”执行的结果，“首行正则表达式”执行的结果需要到目标日志流查看。

如果“首行正则表达式”手动输入的正则表达式有误，则会导致无法查看上报的日志流数据。

图 4-75 设置正则表达式



5. 日志过滤默认关闭。如果需要从海量的日志数据中筛选出符合特定条件的记录，开启日志过滤开关，参考如下操作添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。

图 4-76 日志过滤规则



- 设置白名单规则。添加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。
 - i. 在白名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式），Key值为日志字段名称。
新增多条白名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意白名单就可以被采集。
例如只采集日志中包含hello的日志行，字段值填写hello，过滤规则为.*hello.*
 - ii. 单击操作列的“校验”，在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。

图 4-77 校验



- iii. 校验成功后，单击“确定”或“关闭”退出校验页面。
- 设置黑名单规则。添加的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。
 - i. 在黑名单规则下方，单击“添加”，填写Key值和过滤规则（正则表达式），Key值为日志字段名称。
新增多条黑名单过滤规则时，支持选择And或Or的关系，即全部满足或满足任意黑名单就可以被排除。
例如不采集日志中包含hello的日志行，字段值填写hello，过滤规则为.*hello.*
 - ii. 单击操作列的“校验”在弹出的页面中校验规则的正确性，输入字段值，单击“校验”，提示校验成功。
 - iii. 校验成功后，单击“确定”或“关闭”退出校验页面。
- 6. 上传原始日志。
打开上传原始日志开关后，原始日志将作为content字段的值上传到日志服务。
- 7. 上传解析失败日志。
打开上传解析失败日志开关后，原始日志将作为_content_parse_fail_字段的值上传到日志服务。

8. 关于开启或关闭“上传原始日志”和“上传解析失败日志”时，日志上报参考如下。

图 4-78 结构化解析



表 4-36 日志上报说明

参数	日志上报说明
- 开启“上传原始日志” - 开启“上传解析失败日志”	- 解析成功：上报解析后日志和原始日志content字段。 - 解析失败：上报原始日志content字段和_content_parse_fail_字段。
- 开启“上传原始日志” - 关闭“上传解析失败日志”	- 解析成功：上报解析后日志和原始日志content字段。 - 解析失败：上报原始日志content字段。
- 关闭“上传原始日志” - 开启“上传解析失败日志”	- 解析成功：上报解析后日志。 - 解析失败：上报_content_parse_fail_字段。
- 关闭“上传原始日志” - 关闭“上传解析失败日志”	- 解析成功：上报解析后日志。 - 解析失败：仅上报内置字段和label字段。

- ## 9. 自定义日志时间。

开启后可指定某一字段作为日志时间，或关闭此项使用日志被采集时间作为日志时间。

样例展示:

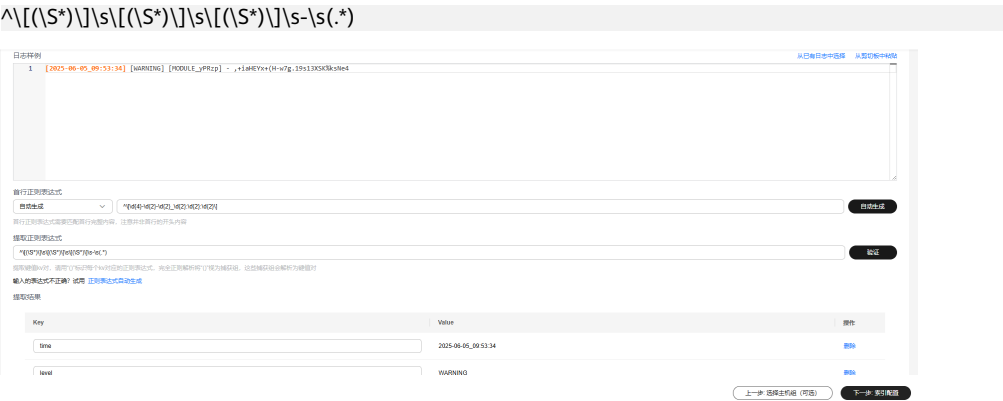
在结构化解析配置中选择“多行-完全正则”模式，日志上报到LTS后，原始日志样例如下所示，经结构化处理后的日志数据上报到LTS的展示效果如图4-79所示。

- 原始日志:
[2025-06-05_10:14:19] [ERROR] [MODULE_JI1bU] - PFs^7BLiq:E^5e\$ja5=450.e1E7p4Q
o^\$(Xbyjhdjb#N6On\$;q(x0\$#IO@go
[2025-06-05_10:14:19] [WARNING] [MODULE_rg02h] - vvAUN29a)ZYcz/fWm\$%/v_h&!;:oY%
[2025-06-05_10:14:19] [ERROR] [MODULE_Cbwrf] - P=wLj;fNamE/u(TN!7uvF+ObDI8/2F
;X\$uqRR!7^7a^mE!ul.R)9F-km1.AI
UA8(!*oB9kgp@zoC<HAY<@wgN#7#SH
[2025-06-05_10:14:19] [INFO] [MODULE_FmHmo] - a+OnTZbUPcv&KNl\$=<orJrgvZ^^n02
[2025-06-05_10:14:19] [DEBUG] [MODULE_G027Q] - o!7pv-znpU\$R=vs4Dhk;=v2Rh17k%)
O^fKR\$0d-rPqQmhZd=_4nQ_wvV+tu
X<C#dm\$Ud\$P\$KCrFWV(0nMdT<c_2/y

首行正则表达式支持自动生成或手动输入。

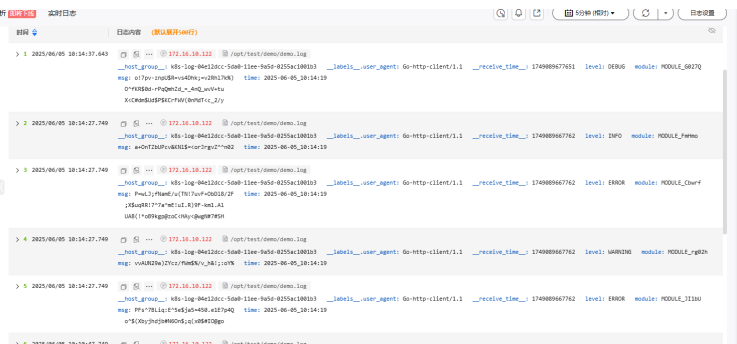
$$\wedge[\backslash d\{4\}-\backslash d\{2\}-\backslash d\{2\} \backslash d\{2\}:\backslash d\{2\}:\backslash d\{2\}\backslash]$$

输入提取正则表达式，单击“验证”，提取key-value键值对，根据日志内容修改key值方便识别。



- 结果展示：

图 4-79 多行-完全正则日志上报结果



组合解析

当您的日志结构太过复杂，涉及多种解析模式，单种解析模式（如Nginx模式、完整正则模式、JSON模式等）无法满足日志解析需求时，您可以使用组合解析规则解析日志，在插件配置下方输入插件语法（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插件配置，ICAgent会根据处理配置顺序逐一执行。

设置操作：

1. 选择组合解析。
 2. 从“从已有日志中选择”或“从剪切板中粘贴”选择日志样例，在插件配置下方输入配置内容。
 3. 您可以根据日志内容参考以下插件语法自定义设置。输入插件配置后，您可以单击“验证”，即可在提取结果下方显示解析结果。您可以通过4提供的日志示例，了解插件配置的用法。
- processor_regex

表 4-37 正则提取

参数	类型	说明
source_key	string	原始字段名。

参数	类型	说明
multi_line_regex	string	首行正则表达式。
regex	string	正则表达式()中为提取字段。
keys	string数组	为提取的内容指定字段名。
keep_source	boolean	是否保留原始字段。
keep_source_if_parse_error	boolean	解析错误是否保留原始字段。

- processor_split_string

表 4-38 分隔符解析

参数	类型	说明
source_key	string	原始字段名。
split_sep	string	分隔符，根据split_type填写。
keys	string数组	为提取的内容指定字段名。
keep_source	boolean	被解析后的日志中是否保留原始字段。
split_type	char/special_char/string	分隔类型，支持char单字符、special_char不可见字符、string字符串。
keep_source_if_parse_error	boolean	解析错误是否保留原始字段。

- processor_split_key_value

表 4-39 键值对分割

参数	类型	说明
source_key	string	原始字段名。
split_sep	string	键值对之间的分隔符，默认值为制表符\t。
split_type	char/special_char/string	分隔类型，支持char单字符、special_char不可见字符、string字符串。

参数	类型	说明
expand_connector	string	单个键值对中键与值之间的分隔符，默认值为半角冒号（:）。
keep_source	boolean	被解析后的日志中是否保留原始字段。

- processor_add_fields

表 4-40 添加字段

参数	类型	说明
fields	json/object	待添加的字段名和字段值。键值对格式，支持添加多个。

- processor_drop

表 4-41 丢弃字段

参数	类型	说明
drop_keys	string数组	丢弃的字段列表。

- processor_rename

表 4-42 重命名字段

参数	类型	说明
source_keys	string数组	待重命名的原始字段。
dest_keys	string数组	重命名后的字段。

- processor_json

表 4-43 json 展开提取

参数	类型	说明
source_key	string	原始字段名。
keep_source	boolean	被解析后的日志中是否保留原始字段。
expand_depth	int	json展开的深度。默认值为0，表示不限制。1表示当前层级，以此类推。
expand_connector	string	json展开时的连接符，默认值为英文符号点（.）
prefix	string	json展开时，对字段名附加的前缀。

参数	类型	说明
keep_source_if_parse_error	boolean	解析错误是否保留原始字段。
parse_escaped_json	boolean	支持解析转义后的json字符串为json对象。

- processor_filter_regex

表 4-44 过滤器

参数	类型	说明
include	json/object	key为日志字段，value为匹配的正则表达式。
exclude	json/object	key为日志字段，value为匹配的正则表达式。

- processor_gotime

表 4-45 提取时间

参数	类型	说明
source_key	string	原始字段名。
source_format	string	原始时间的格式。
source_location	int	原始时间的时区。参数值为空时，表示所在主机或容器的时区。
set_time	boolean	是否将解析后的时间设置为日志时间。
keep_source	boolean	被解析后的日志中是否保留原始字段。

- processor_base64_decoding

表 4-46 base64 解码

参数	类型	说明
source_key	string	原始字段名。
dest_key	string	解析后的目标字段。
keep_source_if_parse_error	boolean	解析错误是否保留原始字段。

参数	类型	说明
keep_source	boolean	被解析后的日志中是否保留原始字段。

- processor_base64_encoding

表 4-47 base64 编码

参数	类型	说明
source_key	string	原始字段名。
dest_key	string	解析后的目标字段。
keep_source	boolean	被解析后的日志中是否保留原始字段。

4. 结合以下参考示例日志，提供多种插件组合使用的方法。

原始日志示例：（以下日志示例仅供参考，请以实际日志为准）

2025-03-19:16:49:03 [INFO] [thread1] {"ref":"https://www.test.com/","curl":"https://www.test.com/so/search?spm=1000.1111.2222.3333&q=linux%20opt%testabcd&t=&u=","sign":"1234567890","pid":"so","0508":{"sign":"112233445566 English bb Error, INFO, error bb&&bb","float":15.25,"long":15},"float":15.25,"long":15}

通过插件配置组合使用解析字段，完整插件配置参考如下，您可以直接复制以下插件配置，单击“验证”，即可查看解析结果。

```
[{
  "detail": {
    "keys": ["nowtime", "level", "thread", "jsonmsg"],
    "keep_source": true,
    "regex": "(\\d{4}-\\d{2}-\\d{2}:\\d{2}:\\d{2}:\\d{2})\\s+\\[(\\w+)\\]\\s+\\[(\\w+)\\]\\s+(.*)",
    "source_key": "content"
  },
  "type": "processor_regex"
}, {
  "detail": {
    "expand_connector": ".",
    "expand_depth": 4,
    "keep_source": true,
    "source_key": "jsonmsg"
  },
  "type": "processor_json"
}, {
  "detail": {
    "keep_source": true,
    "keep_source_if_parse_error": true,
    "keys": [
      "key1",
      "key2",
      "key3"
    ],
    "source_key": "0508.sign",
    "split_sep": ",",
    "split_type": "char"
  },
  "type": "processor_split_string"
}, {
  "detail": {
    "keep_source": true,
    "keep_source_if_parse_error": true,
```

```
    "keys": [
      "a1",
      "a2",
      "a3"
    ],
    "regex": "^((\\w+)(?:[\\^ ]*){1}(\\w+)(?:[\\^ ]*){2}([\\^\\$]+))",
    "source_key": "key1"
  },
  "type": "processor_regex"
}
```

接下来分步介绍示例中用到的插件配置和解析结果。

- a. 将原始日志示例复制到组合解析下方的日志样例中。

原始日志示例：（以下日志示例仅供参考，请以实际日志为准）

```
2025-03-19:16:49:03 [INFO] [thread1] {"ref":"https://www.test.com/","curl":"https://
www.test.com/so/search?spm=1000.1111.2222.3333&q=linux%20opt
%testabcd&t=&u=","sign":"1234567890","pid":"so","0508":{"sign":"112233445566 English bb
Error, INFO, error bb&&bb","float":15.25,"long":15},"float":15.25,"long":15}
```

图 4-80 日志示例



- b. 分析原始日志示例结果如下：日志时间为2025-03-19:16:49:03，日志级别为[INFO]，线程号为[thread1]，日志内容如下，该日志内容是一个完整的json文本。

```
{"ref":"https://www.test.com/","curl":"https://www.test.com/so/search?
spm=1000.1111.2222.3333&q=linux%20opt
%testabcd&t=&u=","sign":"1234567890","pid":"so","0508":{"sign":"112233445566 English bb
Error, INFO, error bb&&bb","float":15.25,"long":15},"float":15.25,"long":15}
```

- c. 按照日志时间、日志级别、线程号、日志内容提取日志验证，每一部分都是用空格来隔开，但是考虑json文本中也可能含有空格，就不选择分隔符插件来提取，使用正则提取插件（processor_regex）提取。

正则表达式参考如下：

日志时间提取为nowtime: `(\d{4}-\d{2}-\d{2}:\d{2}:\d{2})`

日志级别提取为level: `\[(\w+)\]`

线程号提取为thread: `\[(\w+)\]`

日志内容提取为jsonmsg: `(.*)`

每部分中间都是空格分割，完整的提取正则表达式: `(\d{4}-\d{2}-\d{2}:\d{2}:\d{2})\s+\[(\w+)\]\s+\[(\w+)\]\s+(.*)`

正则提取插件（processor_regex）配置如下，需要将regex参数的值替换为正则表达式，同时将正则表达式里的\替换为\\用于转义。

```
{
  "detail": {
    "keys": ["nowtime", "level", "thread", "jsonmsg"],
    "regex": "(\\d{4}-\\d{2}-\\d{2}:\\d{2}:\\d{2})\\s+\\[(\\w+)\\]\\s+\\[(\\w+)\\]\\s+(.*)",
    "source_key": "content"
  },
  "type": "processor_regex"
}
```

- d. 将配置内容复制到插件配置框中，单击“验证”，在提取结果下方即可显示提取结果，key和value值提取正确。

- e. 接下来可以使用json插件（processor_json）将日志内容（jsonmsg）进行json展开。（每个插件配置{}中间需要加英文逗号隔开）

```
{
  "detail": {
    "expand_connector": ".",
    "expand_depth": 4,
    "keep_source": true,
    "source_key": "jsonmsg",
    "source_location": 8
  },
  "type": "processor_json"
}
```

将json插件（processor_json）配置追加到正则提取插件（processor_regex）配置后面，单击“验证”，在提取结果下方即可显示提取结果。

- f. 如果您需要将json提取得到的0508.sign字段进一步分割，可以使用分隔符解析插件（processor_split_string）中的“,”进行分隔拆分，提取key1，key2，key3字段。

```
{
  "detail": {
    "keys": [
      "key1",
      "key2",
      "key3"
    ],
    "source_key": "0508.sign",
    "split_sep": ",",
    "split_type": "char"
  },
  "type": "processor_split_string"
}
```

将分隔符解析插件（processor_split_string）配置追加到json插件（processor_json）配置后面，单击“验证”，在提取结果下方即可显示key1，key2，key3字段。

- g. 您还可以将key1使用正则提取插件（processor_regex）提取出a1，a2，a3等字段。

```
{
  "detail": {
    "keep_source": true,
    "keep_source_if_parse_error": true,
    "keys": [
      "a1",
      "a2",
      "a3"
    ],
    "regex": "^(\\w+)(?:[\\^ ]*){1}(\\w+)(?:[\\^ ]*){2}([\\^\\$]+)",
    "source_key": "key1"
  },
  "type": "processor_regex"
}
```

将正则提取插件（processor_regex）配置追加到分隔符解析插件（processor_split_string）配置后面，单击“验证”，在提取结果下方即可显示a1，a2，a3字段。

自定义日志时间

开启自定义日志时间后，使用自定义日志内容的时间作为日志搜索页面日志内容的时间，请参考[表4-48](#)设置参数。关闭自定义日志时间后使用日志采集时间作为日志搜索页面日志内容的时间。

常见的时间标准、示例及对应的时间表达式如下[表4-50](#)。

表 4-48 参数配置表

参数	说明	示例
时间字段key名称	已提取字段的名称。单击下拉框选择已提取的字段，该字段为string或long类型。	test
字段value	已提取的字段value，选择字段key后，将自动填充。	2023-07-19 12:12:00
时间格式	默认情况下，云日志服务LTS中的日志时间戳精确到秒，因此时间格式只需配置到秒，无需配置毫秒、微秒等信息。请参考表4-49。 - 若时间格式填写错误或指定字段不存在，将使用日志被采集时间作为日志时间。 - 对结构化解析进行字段名称修改、字段删除、字段类型修改等操作，都需要重新校验时间字段。	yyyy-MM-dd HH:mm:ss
操作	单击  校验图标，提示“时间格式和字段value匹配成功”则表示校验成功。	-

表 4-49 常见时间格式

时间格式	说明	示例
EEE	星期的缩写。	Fri
EEEE	星期的全称。	Friday
MMM	月份的缩写。	Jan
MMMM	月份的全称。	January
dd	每月第几天，十进制，范围为01~31。	07, 31
HH	小时，24小时制。	22
h	小时，12小时制。	11
MM	月份，十进制，范围为01~12。	08
mm	分钟，十进制，范围为00~59。	59
a	am或pm。	am、pm
h:mm:ss a	12小时制的时间组合。	11:59:59 am
HH:mm	小时和分钟组合。	23:59
ss	秒数，十进制，范围为00~59。	59
yy	年份，十进制，不带世纪，范围为00~99。	04、98

时间格式	说明	示例
yyyy	年份，十进制。	2004、1998
d	每月第几天，十进制，范围为1~31。	7、31
%s	Unix时间戳。	147618725

表 4-50 示例

示例	时间表达式	时间标准
2022-07-14T19:57:36+08:00	yyyy-MM-dd'T'HH:mm:ssXXX	自定义
1548752136	%s	自定义
27/Jan/2022:15:56:44	dd/MMM/yyyy:HH:mm:ss	自定义
2022-07-24T10:06:41.000	yyyy-MM-dd'T'HH:mm:ss.SSS	自定义
Monday, 02-Jan-06 15:04:05 MST	EEEE, dd-MMM-yy HH:mm:ss Z	RFC850
Mon, 02 Jan 2006 15:04:05 MST	EEE, dd MMM yyyy HH:mm:ss Z	RFC1123
02 Jan 06 15:04 MST	dd MMM yy HH:mm Z	RFC822
02 Jan 06 15:04 -0700	dd MMM yy HH:mm Z	RFC822Z
2023-01-02T15:04:05Z07:00	yyyy-MM-dd'T'HH:mm:ss Z	RFC3339
2022-12-11 15:05:07	yyyy-MM-dd HH:mm:ss	自定义
2025-02-24T05:24:07.085Z	yyyy-MM-dd'T'HH:mm:ss.SSSZ	自定义

4.5 使用云服务接入 LTS

4.5.1 云服务接入 LTS 概述

云日志服务（LTS）支持采集计算、存储、安全、数据库等多种华为云服务的日志数据，您可以使用LTS对云服务日志进行关键词搜索、运营数据统计分析、运行状况监控告警等多种操作。当前LTS支持采集的云服务日志如下表所示：

表 4-51 云服务接入

序号	云服务简称	云服务名称	日志类型	文档链接	仪表盘
1	AOM	应用运维管理	全量日志	应用运维管理AOM接入LTS	-
2	APIG	API网关	网关访问日志（APIG）	API网关APIG接入LTS	• APIG监控中心 • APIG访问中心 • APIG秒级监控
4	BMS	裸金属服务器	全量日志	裸金属服务BMS文本日志接入LTS	-
5	CBH	云堡垒机	操作日志	云堡垒机CBH接入LTS	-
6	CCE	云容器引擎	• 用户应用日志 • CCE管理面日志	云容器引擎CCE应用日志接入LTS	-
8	CFW	云防火墙	• 攻击日志（CFW攻击日志） • 访问日志（CFW访问控制日志） • 流量日志（CFW流量日志）	云防火墙CFW接入LTS	-
9	CTS	云审计服务	云服务管理面操作日志（CTS）	云审计服务CTS接入LTS	-
11	DDS	文档数据库服务	• 审计日志（DDS审计日志） • 错误日志（DDS错误日志） • 慢日志（DDS慢日志）	文档数据库服务DDS接入LTS	DDS审计日志中心
12	DDoS防护 AAD	DDoS防护 AAD	全量日志	DDoS防护 AAD接入LTS	-

序号	云服务简称	云服务名称	日志类型	文档链接	仪表盘
13	DMS	分布式消息服务 Kafka版	DMS重平衡日志	分布式消息服务Kafka版接入LTS	-
14	DRS	数据复制服务	访问日志	数据复制服务DRS接入LTS	-
15	DWS	数据仓库服务	• CN节点日志 • DN节点日志 • 操作系统messages日志 • 审计日志	数据仓库服务GaussDB(DWS)接入LTS	-
16	ECS	弹性云服务器	全量日志（ICAgent采集文本日志）	云主机ECS文本日志接入LTS	-
17	ELB	弹性负载均衡	7层访问日志（ELB）	弹性负载均衡ELB接入LTS	• ELB监控中心 • ELB访问中心 • ELB秒级监控
18	ER	企业路由器	全量日志（ER企业路由器）	企业路由器ER接入LTS	-
19	FunctionGraph	函数工作流	函数执行日志	函数工作流FunctionGraph接入LTS	-
20	GaussDB	云数据库GaussDB	审计日志（GAUSSV5审计日志）	云数据库GaussDB接入LTS	-
21	GES	图引擎服务	审计日志	图引擎服务GES接入LTS	-
22	TaurusDB	云数据库TaurusDB	• 错误日志（GAUSSDB_MYSQL错误日志） • 慢日志（GAUSSDB_MYSQL慢日志）	云数据库TaurusDB接入LTS	-

序号	云服务简称	云服务名称	日志类型	文档链接	仪表盘
23	Gemini DB Redis	云数据库 Gemini DB Redis	慢日志 (GeminiDB Redis慢日志)	云数据库 GeminiDB接入 LTS	-
24	Gemini DB Mongo	云数据库 Gemini DB Mongo	- 错误日志 (GeminiDB Mongo错误日志) - 慢日志 (GeminiDB Mongo慢日志)	云数据库 GeminiDB Mongo接入 LTS	-
25	Gemini DB Cassandra	云数据库 Gemini DB Cassandra	慢日志 (GeminiDB Cassandra慢日志)	云数据库 GeminiDB Cassandra接入 LTS	-
27	IoTDA	设备接入	设备运行日志	设备接入IoTDA接入 LTS	-
28	ModelArts	Model Arts	运行日志	AI开发平台 ModelArts接入 LTS	-
29	MRS	MapReduce服务	全量日志 (ICAgent采集文本日志)	MapReduce服务MRS接入 LTS	-
30	RDS for MySQL	云数据库 RDS for MySQL	- 错误日志 (MYSQL错误日志) - 慢日志 (MYSQL慢日志)	云数据库RDS for MySQL接入 LTS	-
31	RDS for PostgreSQL	云数据库 RDS for PostgreSQL	- 错误日志 (POSTGRESQL错误日志) - 慢日志 (POSTGRESQL慢日志)	云数据库RDS for PostgreSQL接入 LTS	-

序号	云服务简称	云服务名称	日志类型	文档链接	仪表盘
32	RDS for SQLServer	云数据库RDS for SQL Server	错误日志（SQLSERVER错误日志）	云数据库RDS for SQLServer接入LTS	-
33	ROMA Connect	应用与数据集成平台	网关访问日志	应用与数据集成平台ROMA Connect接入LTS	-
35	SMN	消息通知服务	消息传输日志（SMN）	消息通知服务SMN接入LTS	-
36	SecMaster	安全云脑	全量日志	安全云脑SecMaster接入LTS	-
37	ServiceStage	应用管理与运维平台	ServiceStage容器应用日志	ServiceStage容器应用日志接入LTS	-
38	ServiceStage	应用管理与运维平台	ServiceStage云主机日志	ServiceStage云主机日志接入LTS	-
39	VPC	虚拟私有云	VPC流日志（VPC）	虚拟私有云VPC接入LTS	VPC流日志
40	OBS	对象存储服务	从LTS转储到OBS的日志	对象存储服务OBS接入LTS（邀测）	-
41	WAF	Web应用防火墙	- 攻击日志 - 访问日志	Web应用防火墙WAF接入LTS	-

4.5.2 应用运维管理 AOM 接入 LTS

支持应用运维管理AOM的日志接入LTS。

具体接入方法请参见[接入LTS](#)。

4.5.3 API 网关 APIG 接入 LTS

支持API网关APIG日志接入LTS。

前提条件

已创建并使用华为云APIG实例。

设置 APIG 接入 LTS

云日志服务接入方式为API网关 APIG时，按照如下操作完成接入配置。

步骤1 登录[云日志服务控制台](#)。

步骤2 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“API网关 APIG”进行APIG接入配置。

步骤3 选择日志流。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
3. 单击“下一步：APIG配置”。

步骤4 APIG配置，单击“前往APIG配置”。详细操作请参见[日志分析](#)。

步骤5 单击“下一步：日志流配置”。

表 4-52 日志流配置参数表

参数	说明
自动对日志流进行结构化配置和索引配置	开启该按钮，日志流结构化配置为APIG系统模板，索引配置时将所有APIG解析出来的字段打开快速分析按钮。配置结构化和索引后，才能对APIG日志进行SQL分析，并提供可视化图表。
自动为日志流添加标签：log_type=apig_access	开启该按钮，自动为日志流添加标签log_type=apig_access后，APIG仪表盘模板才能匹配该日志流。
自动为日志流创建仪表盘	开启该按钮，自动为日志流创建APIG仪表盘。

步骤6 单击“提交”。

----结束

4.5.4 Astro 轻应用接入 LTS

支持Astro轻应用日志接入LTS。在运行日志中，可查看程序的运行日志，便于问题定位。同时，AstroZero支持将日志采集到云日志服务（Log Tank Service，简称LTS），提供更强大的日志收集、分析、存储等服务。用户通过云日志服务，可快速高效地进行用户业务趋势分析、安全监控审计等操作。

具体接入方法请参见[查看运行日志](#)。

4.5.5 云堡垒机 CBH 接入 LTS

支持云堡垒机（CBH）日志接入LTS。

具体接入方法请参见[配置LTS日志外发服务](#)。

4.5.6 内容分发网络 CDN 接入 LTS

支持内容分发网络CDN实时日志接入LTS。实时日志可以帮助您分析加速域名遇到的异常问题，也可以帮助您了解用户的访问情况。

具体接入方法请参见[实时日志](#)。

4.5.7 云防火墙 CFW 接入 LTS

支持云防火墙CFW日志接入LTS。LTS对于采集的日志数据，通过海量日志数据的分析与处理，可以为您提供一个实时、高效、安全的日志处理能力。

具体接入方法请参见[配置日志](#)。

4.5.8 云审计服务 CTS 接入 LTS

支持云审计服务CTS日志接入LTS。

前提条件

已购买并使用华为云CTS实例。

设置 CTS 接入 LTS

云日志服务接入方式为云审计 CTS时，按照如下操作完成接入配置。

云审计服务记录的用户操作事件日志接入到云日志服务后，在云日志服务控制台默认存储时间为30天，您在创建存放该日志的日志组时，可以对日志存储周期进行设置（1-365天）。详情请参考[管理日志组](#)

步骤1 登录[云日志服务控制台](#)。

步骤2 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“云审计 CTS”进行CTS接入配置。

步骤3 选择日志流。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
3. 单击“下一步：CTS配置”。

步骤4 单击“前往CTS配置”。具体的操作步骤及参数配置，请见[云审计服务《用户指南》](#)。

步骤5 单击“下一步：日志流配置”。

表 4-53 日志流配置参数表

参数	说明
自动对日志流进行结构化配置和索引配置	开启该按钮，日志流结构化配置为CTS系统模板，索引配置时将所有CTS解析出来的字段打开快速分析按钮。配置结构化和索引后，才能对CTS日志进行SQL分析，并提供可视化图表。

步骤6 单击“提交”。

----结束

4.5.9 分布式缓存服务 DCS 接入 LTS

支持分布式缓存服务DCS日志接入LTS。命令审计日志，是记录客户端访问DCS操作的一种日志，由华为云的云日志服务（LTS）提供存储、查询和分析等功能。

具体接入方法请参见[查看Redis实例的命令审计日志](#)。

4.5.10 文档数据库服务 DDS 接入 LTS

支持文档数据库服务DDS日志接入LTS。

具体接入方法请参见[日志配置管理](#)。

4.5.11 DDoS 防护 AAD 接入 LTS

支持DDoS防护 AAD日志接入LTS。启用Anti-DDoS防护功能后，您可以将攻击日志记录到云日志服务（Log Tank Service，简称LTS）中，通过LTS记录的Anti-DDoS日志数据，快速高效地进行实时决策分析、设备运维管理以及业务趋势分析。关于DDoS防护AAD服务的更多信息请参考[服务版本差异](#)。

- 使用DDoS原生基础防护时，具体接入方法请参见[开启日志记录](#)。
- 使用DDoS原生高级防护时，具体接入方法请参见[开启日志记录](#)。
- 使用DDoS高防时，具体接入方法请参见[开启日志记录](#)。

4.5.12 分布式消息服务 Kafka 版接入 LTS

支持分布式消息服务Kafka日志接入LTS。重平衡日志记录Rebalance的详情，包括Rebalance时间、原因和触发Rebalance的客户端等。重平衡日志存储在云日志服务（Log Tank Service，简称LTS）中，由LTS提供查询功能。

具体接入方法请参见[查看Kafka重平衡日志](#)。

4.5.13 数据复制服务 DRS 接入 LTS

支持数据复制服务（DRS）日志接入LTS。配置访问日志后，DRS实例（包含实时迁移、备份迁移、实时同步、实时灾备和录制回放实例）新生成的所有日志记录会上传到云日志服务（Log Tank Service，简称LTS）进行管理。

具体接入方法请参见[日志配置管理](#)。

4.5.14 数据仓库服务 GaussDB(DWS)接入 LTS

支持数据仓库GaussDB（DWS）日志接入LTS。
具体接入方法请参见[集群日志管理](#)。

4.5.15 弹性负载均衡 ELB 接入 LTS

支持弹性负载均衡ELB日志接入LTS。

前提条件

已创建并使用华为云ELB实例。

使用限制

当前ELB日志仅支持七层独享型负载均衡和七层共享型负载均衡，不支持四层共享型负载均衡。

设置 ELB 接入

云日志服务接入方式为负载均衡 ELB时，按照如下操作完成接入配置。

- 步骤1 登录[云日志服务控制台](#)。
- 步骤2 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“负载均衡 ELB”进行ELB接入配置。
- 步骤3 选择日志流。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。

2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。

3. 单击“下一步：ELB配置”。
- 步骤4 单击“前往ELB配置”。具体的操作步骤，请见[访问日志](#)。
- 步骤5 单击“下一步：日志流配置”。

表 4-54 日志流配置参数表

参数	说明
自动对日志流进行结构化配置和索引配置	开启该按钮，日志流结构化配置为ELB系统模板，索引配置时将所有ELB解析出来的字段打开快速分析按钮。配置结构化和索引后，才能对ELB日志进行SQL分析，并提供可视化图表。
自动为日志流添加标签：log_type=elb_7layer_access	开启该按钮，自动为日志流添加标签（log_type=elb_7layer_access）后，ELB仪表盘模板才能匹配该日志流。

参数	说明
自动为日志流创建仪表盘	开启该按钮，自动为日志流创建ELB仪表盘。

步骤6 单击“提交”。

----结束

4.5.16 企业路由器 ER 接入 LTS

支持ER企业路由器日志接入LTS。

具体接入方法请参见[创建流日志](#)。

4.5.17 函数工作流 FunctionGraph 接入 LTS

支持函数工作流FunctionGraph日志接入LTS。

具体接入方法请参见[管理函数日志](#)。

4.5.18 云数据库 GaussDB 接入 LTS

支持云数据库 GaussDB日志接入LTS。

具体接入方法请参见[对接LTS并查看数据库审计日志](#)。

4.5.19 图引擎服务 GES 接入 LTS

支持图引擎服务GES日志接入LTS。

具体接入方法请参见[对接LTS](#)。

4.5.20 云数据库 TaurusDB 接入 LTS

LTS支持云数据库 TaurusDB日志接入。

具体接入方法请参见[日志配置管理](#)。

4.5.21 云数据库 GeminiDB 接入 LTS

支持云数据库GeminiDB Redis日志接入LTS。配置访问日志后，GeminiDB Redis实例新生成的日志记录会上传到云日志服务（Log Tank Service，简称LTS）进行管理。

具体接入方法请参见[日志配置管理](#)。

4.5.22 云数据库 GeminiDB Mongo 接入 LTS

支持云数据库GeminiDB Mongo日志接入LTS。

具体接入方式请参考[日志配置管理](#)。

4.5.23 云数据库 GeminiDB Cassandra 接入 LTS

支持云数据库GeminiDB Cassandra日志接入LTS。

如果您有需要请提工单给GeminiDB云服务开通。详细操作请参考[提交工单](#)。

4.5.24 华为 HiLens 接入 LTS

支持华为HiLens日志接入LTS。

具体接入方法请参见[管理设备日志](#)。

4.5.25 设备接入 IoTDA 接入 LTS

支持设备接入IoTDA日志接入LTS。

具体接入方法请参见[查看运行日志](#)。

4.5.26 AI 开发平台 ModelArts 接入 LTS

支持AI开发平台ModelArts日志接入LTS。

具体接入方法请参见[部署模型为在线服务](#)。

4.5.27 MapReduce 服务 MRS 接入 LTS

支持MapReduce服务MRS日志接入LTS。

具体接入方法请参见[MRS服务对接云日志服务](#)。

4.5.28 云数据库 RDS for MySQL 接入 LTS

支持云数据库RDS for MySQL日志接入LTS。

具体接入方法请参见[日志配置管理](#)。

4.5.29 云数据库 RDS for PostgreSQL 接入 LTS

支持云数据库RDSfor PostgreSQL日志接入LTS。

具体接入方法请参见[日志配置管理](#)。

4.5.30 云数据库 RDS for SQLServer 接入 LTS

支持云数据库RDS for SQLServer日志接入LTS。

具体接入方法请参见[日志管理](#)。

4.5.31 应用与数据集成平台 ROMA Connect 接入 LTS

支持应用与数据集成平台ROMA Connect日志接入LTS。

具体接入方法请参见[查看API调用日志](#)。

4.5.32 视频直播 Live 接入 LTS

支持视频直播Live日志接入LTS。视频直播支持实时查看播放域名被网络用户访问的详细日志记录。配置实时日志时需要您对接云日志服务，并且已经创建需要关联的云日志组和日志流。

具体接入方法请参见[实时配置日志](#)。

4.5.33 消息通知服务 SMN 接入 LTS

支持消息通知服务SMN日志接入LTS。

具体接入方法请参见[消息传输日志](#)。

4.5.34 安全云脑 SecMaster 接入 LTS

支持安全云脑SecMaster日志接入LTS。

具体接入方法请参见[新增数据投递](#)。

4.5.35 对象存储服务 OBS 接入 LTS（邀测）

支持将对象存储服务桶内的文件一次性或定期导入到云日志服务，OBS服务接入LTS成功后，即可对日志进行搜索分析、日志加工等操作。加密桶的文件不支持导入LTS，若需要导入加密桶的文件，请先删除桶的加密配置。详细请参考[删除桶的加密配置](#)。

使用OBS日志文件导入LTS功能非实时性管道，不适用实时性要求较高的业务场景。

说明

目前此功能在邀测中，暂不支持申请开通。

设置单个对象存储 OBS 接入 LTS

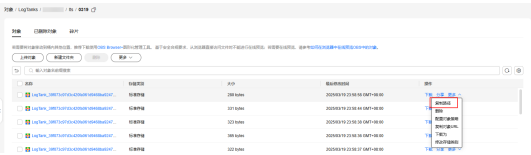
- 步骤1** 登录[云日志服务控制台](#)。
- 步骤2** 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“云服务接入-对象存储OBS”进行OBS接入配置。
- 或在左侧导航栏中，选择“日志接入 > 接入管理”，单击“创建”，在弹出的页面中，选择“云服务接入-对象存储OBS”进行OBS接入配置。
- 步骤3** 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组。若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。详细请参考[管理日志组](#)。
- 步骤4** 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流。若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。详细请参考[管理日志流](#)。
- 步骤5** 单击“下一步：采集配置”。
- 步骤6** 在采集配置页面，参考[表4-55](#)设置参数。

表 4-55 采集配置

参数类型	名称	说明
基本配置	采集配置名称	自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。

参数类型	名称	说明
	任务监控	默认开启。 开启后会将每次任务执行状态写入日志流lts-system/lts-obs2lts-statistics，您可以查看OBS文件导入LTS任务监控中心或者配置告警规则，及时发现数据导入过程中可能出现的异常问题。
OBS数据源配置	OBS桶	选择需要导入日志文件到LTS的OBS桶。
	文件夹前缀	输入待导入的OBS文件前缀，用于准确定位待导入的文件，支持输入文件前缀your_prefix/或完整路径your_prefix/file.gz。 只支持导入5GB（单个文件）以内的原始文件，且只能串行导入。 您可以参考如下方法获取日志的文件夹前缀： 1. 在OBS控制台桶列表页面，单击目标桶名称进入对象页面。 2. 在“对象”页面，找到需要导入LTS的OBS文件名称，单击该OBS文件操作列的“更多 > 复制路径”。（以下示例仅供参考，请以实际路径为准） 如果需要导入文件夹名称0319内的日志，复制路径结果为“LogTanks/cn-north-4/lts/0319/LogTank_39f073c97d3c420fa061d9468ba9247c_2025-03-19T15-56-00Z_05d6486f96e43f8a.gz”，那文件夹前缀为“LogTanks/cn-north-4/lts/0319”。
	文件正则过滤	用于过滤文件的正则表达式，只有文件名匹配该正则表达式的文件才会被导入。若不填写正则表达式，表示不过滤文件。 假设目录下有aab和aba文件： - 只匹配aab文件，正则写法：aab或aa或^aab或者aa. - 只匹配aba文件，正则写法：aba或^aba或者^ab，不能写ab因为这样aab也是能匹配上的 - 匹配aab和aba，正则写法：ab或者a.* - 如果存在正则关键字的字符需要转义，例如{}则需要转义为\{\}
	压缩格式	支持自动检测、不压缩、压缩gzip、压缩zip、压缩snappy。如果是zip压缩文件，只能包含一个文件，不能有任何文件夹。

图 4-81 复制路径



参数类型	名称	说明
	导入间隔	● 一次性：只导入一次，云日志服务不会检测新出现的文件。只支持导入标准存储的OBS文件，不支持导入归档存储的OBS文件，如需导入，请提前将归档存储的OBS文件解冻恢复为标准存储的OBS文件。 ● 固定间隔：设置固定时间导入文件，云日志服务会检测新出现的文件，并导入云日志服务。 开启“解冻归档文件”：只支持解冻对象存储类别为归档存储的OBS文件。归档存储文件需要激活此选项，归档存储文件激活需要一定时间，归档存储文件加急恢复典型值在1~5分钟。详情请参考对象恢复方式及耗时。首次单击右下角的“预览”可能超时，请再次重试单击“预览”。 - 周期任务首次开始扫描obs上文件，文件的最后修改时间区间范围是(周期首次运行时间-固定间隔时间, 周期首次运行时间]。例如周期任务首次开始12:00:00，obs导入任务配置的固定时间间隔为10分钟，那么首次扫描出来的obs文件其最后修改时间在(11:50:00, 12:00:00]区间内，第二次周期12:10:00运行，则扫描出来的文件其最后修改时间在(12:00:00, 12:10:00]区间内。 - 一次性任务如果处理某个文件失败后，扫描出来的其余文件不会再被解析上报到LTS。 - 固定间隔任务关闭后重新开启，监控数据连续性最多保持一天。 - 固定间隔任务在某个周期时间内处理文件失败后，该周期时间内扫描出来的其余文件不会再被解析上报到LTS。
	文件修改时间过滤	导入间隔选择一次性时，支持按修改时间过滤文件。 ● 选择“所有”，不按修改时间过滤。 ● 选择“某时间开始”，设置开始时间，按设置的开始时间过滤文件。 ● 选择“特定时间范围”，设置开始时间和结束时间，按设置的特定时间范围过滤文件。
数据格式配置	日志文件编码	日志文件编码支持UTF-8、GBK。 UTF-8编码是一种变长编码方式，用于表示Unicode字符集。GBK全称《汉字内码扩展规范》，中文计算机编码的一种，是ASCII码和GB2312编码的扩展。

参数类型	名称	说明
	提取模式	LTS解析OBS文件内日志时，需针对不同日志类型配置提取策略，请根据业务需求选择对应的提取模式。 当OBS日志超过1MB限制时，LTS会按日志类型采取差异化处理： ● 单行日志单行超过1MB的部分被截断丢弃。 ● 多行日志多行块超过1MB的部分被截断丢弃。 ● ORC格式日志是单行解析，如果单行日志超过1MB直接被丢弃。 ● JSON格式日志是单行解析，如果单行日志超过1MB直接被丢弃。 提取模式说明： ● 单行全文：采集完整的单行日志全文，不做结构化解析。如果您需要对日志做结构化解析，请在完成OBS文件导入配置后，请参考设置云端结构化解析日志设置。 ● 多行全文：采集完整的多行日志全文（如堆栈日志），不做结构化解析。如果您需要对日志做结构化解析，请在完成OBS文件导入配置后，请参考设置云端结构化解析日志设置。 ● ORC：采集ORC格式的日志。 不开启“自定义时间”时，使用日志被采集时间作为日志时间。 开启“自定义时间”时，可指定某一字段作为日志时间。填写时间字段Key名称、字段Value、时间格式，设置完成后，单击  校验。如果导入的数据写入到css集群，LTS不支持ORC格式日志自定义两天前的时间。自定义时间格式请参考自定义日志时间。 ● JSON：采集JSON格式的日志。 不开启“自定义时间”时，使用日志被采集时间作为日志时间。 开启“自定义时间”时，可指定某一字段作为日志时间。填写时间字段Key名称、字段Value、时间格式，设置完成后，单击  校验。如果导入的数据写入到css集群，LTS不支持JSON格式日志自定义两天前的时间。自定义时间格式请参考自定义日志时间。 设置json解析层数，取值范围为1~4，只能整数，默认值为1。将json格式日志的字段展开，例如原始日志为{"key1":{"key2":"value"}}，解析1层日志为：{"key1":{"key2":"value"}}，解析2层日志为：{"key1.key2":"value"}。

步骤7 设置完成后，单击右下角的“预览”。预览只扫描并返回符合条件的第一个文件的前10行内容。

LTS仅支持预览≤10MB的文件。若提示文件过大，您可以临时修改文件正则过滤规则，指定一个<10MB的文件进行预览。预览完成后，再将文件正则过滤规则恢复为原配置。

步骤8 在结果预览下方查看结果，确认无误后，单击“提交”。

步骤9 接入成功后，则会生成一条接入配置信息。邀测期间，最多支持10个运行中的接入任务。

- 导入间隔选择一次性导入的OBS导入LTS任务完成后，建议等待5分钟后，再重新创建新任务或者打开“接入配置开关”开关。
- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“修改”重新修改接入配置信息。不支持修改导入间隔为一次性的任务。
- 单击接入配置操作列的“标签管理”即可添加标签。
- 单击接入配置操作列的“复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。

警告

删除接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，且删除后无法恢复，请谨慎操作。

- 若不需要采集日志，关闭“接入配置开关”列的开关。若需要重新采集日志，需要重新开启“接入配置开关”列的开关。

警告

关闭接入配置后会导致日志无法正常采集，可能会影响用户日志相关业务异常，请谨慎操作。

----结束

4.5.36 虚拟私有云 VPC 接入 LTS

支持虚拟私有云VPC日志接入LTS。

前提条件

购买并使用华为云VPC实例。

使用限制

目前支持的弹性云服务器，请参见[VPC流日志简介](#)。

设置 VPC 接入

云日志服务接入方式为虚拟私有云 VPC时，按照如下操作完成接入配置。

步骤1 登录[云日志服务控制台](#)。

步骤2 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“虚拟私有云 VPC”进行VPC接入配置。

步骤3 选择日志流。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
3. 单击“下一步：VPC配置”。

步骤4 单击“前往VPC配置”。具体的操作步骤及参数配置，请参见[创建VPC流日志](#)。

步骤5 单击“下一步：日志流配置”。

表 4-56 日志流配置参数表

参数	说明
自动对日志流进行结构化配置和索引配置	开启该按钮，日志流结构化配置为VPC系统模板，索引配置时将所有VPC解析出来的字段打开快速分析按钮。配置结构化和索引后，才能对VPC日志进行SQL分析，并提供可视化图表。
自动为日志流添加标签： log_type=vpc_flow	开启该按钮，自动为日志流添加标签 log_type=vpc_flow后，VPC仪表盘模板才能匹配该日志流。
自动为日志流创建仪表盘	开启该按钮，自动为日志流创建VPC仪表盘。

步骤6 单击“提交”。

----结束

4.5.37 Web 应用防火墙 WAF 接入 LTS

支持Web应用防火墙 WAF接入LTS。

具体接入方法请参见[通过LTS记录WAF全量日志](#)。

4.6 使用 API 接入 LTS

4.6.1 API 接入概述

您可以通过调用云日志服务LTS提供的REST风格API将日志上报到LTS，具体有上报日志和上报高精度日志两个接口。

以下是两个接口的适用场景和各区域的访问域名：

表 4-57 适用场景

API名称	日志时间	举例说明	适用场景
上报日志	用户调用API上传一批日志时，通过log_time_ns字段指定一个初始时间。 每条日志的时间根据log_time_ns字段的值顺序递增。	<pre>{ "log_time_ns": "1586850540000000000", "contents": ["log1", "log2", "log3"], "labels": { "user_tag": "string" } }</pre> 上报到LTS后： log1的日志时间为： 15868505400000000000 log2的日志时间为： 15868505400000000001 log3的日志时间为： 15868505400000000002	上传的一批日志是在相近时间点、按顺序产生的。
上报高精度日志	用户调用API上传一批日志时，每一条日志都需要通过log_time_ns字段指定日志时间。	<pre>{ "contents": [{ "log_time_ns": "1586850540000000000", "log": "log4" }, { "log_time_ns": "1586850540000000008", "log": "log5" }], "labels": { "user_tag": "string" } }</pre> 上报到LTS后： log4的日志时间为： 15868505400000000000 log5的日志时间为： 15868505400000000008	上传的一批日志是在不同时间、非顺序产生的，希望每条日志的时间单独指定。

表 4-58 访问域名

区域名称	区域	访问域名（Endpoint）
华北-北京一	cn-north-1	lts-access.cn-north-1.myhuaweicloud.com
华北-北京二	cn-north-2	lts-access.cn-north-2.myhuaweicloud.com
华北-北京四	cn-north-4	lts-access.cn-north-4.myhuaweicloud.com
华北-乌兰察布一	cn-north-9	lts-access.cn-north-9.myhuaweicloud.com
华北-乌兰察布-汽车一	cn-north-11	lts-access.cn-north-11.myhuaweicloud.com
华东-上海二	cn-east-2	lts-access.cn-east-2.myhuaweicloud.com
华东-上海一	cn-east-3	lts-access.cn-east-3.myhuaweicloud.com
华东二	cn-east-4	lts-access.cn-east-4.myhuaweicloud.com
华东-青岛	cn-east-5	lts-access.cn-east-5.myhuaweicloud.com
华南-广州	cn-south-1	lts-access.cn-south-1.myhuaweicloud.com
华南-深圳	cn-south-2	lts-access.cn-south-2.myhuaweicloud.com
华南-广州友好	cn-south-4	lts-access.cn-south-4.myhuaweicloud.com
中国-香港	ap-southeast-1	lts-access.ap-southeast-1.myhuaweicloud.com
亚太-曼谷	ap-southeast-2	lts-access.ap-southeast-2.myhuaweicloud.com
亚太-新加坡	ap-southeast-3	lts-access.ap-southeast-3.myhuaweicloud.com
西南-贵阳一	cn-southwest-2	lts-access.cn-southwest-2.myhuaweicloud.com
非洲-约翰内斯堡	af-south-1	lts-access.af-south-1.myhuaweicloud.com
俄罗斯-莫斯科二	ru-northwest-2	lts-access.ru-northwest-2.myhuaweicloud.com

4.6.2 上报日志接口参考

功能介绍

该接口用于主机上报租户日志给LTS。

接入点IP可在LTS控制台安装ICAgent的安装命令中获取，端口为8102，调用时使用该参数请参见[请求示例](#)。

关于上报日志的约束与限制请参考[日志读写限制](#)。

URI

POST /v2/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/tenant/contents

表 4-59 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见： 获取项目ID 。
log_group_id	是	String	日志组ID，获取方式请参见： 获取日志组ID和日志流ID 。
log_stream_id	是	String	日志流ID，获取方式请参见： 获取日志组ID和日志流ID 。 每个logstream写入速率最大不能超过100MB/S，超过此规格可能会导致日志丢失。

请求参数

表 4-60 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	从IAM获取的用户Token，获取方式请参见： 获取用户Token 。
Content-Type	是	String	用于定义消息体的格式，该字段填为：application/json;charset=UTF-8。

表 4-61 请求 Body 参数

参数	是否必选	参数类型	描述
log_time_ns	是	Long	日志数据上报时间，UTC时间（纳秒）。 通过接口上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。
contents	是	Array of String	日志内容。
labels	是	Object	用户自定义label。 请不要将字段名称设置为 内置保留字段 ，否则可能会造成字段名称重复、查询不精确等问题。
tenant_project_id	否	String	租户项目ID，获取方式请参见： 获取项目ID 。

响应参数

状态码为200时：

表 4-62 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。
error_msg	String	调用失败响应信息描述。
result	String	响应结果。

状态码为400时：

表 4-63 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。
error_msg	String	调用失败响应信息描述。
result	String	响应结果。

状态码为401时：

表 4-64 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。
error_msg	String	调用失败响应信息描述。
result	String	响应结果。

状态码为500时:

表 4-65 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。
error_msg	String	调用失败响应信息描述。
result	String	响应结果。

状态码为503时:

表 4-66 响应 Body 参数

参数	参数类型	描述
result	String	被请求的服务无效，服务不可用。

请求示例

```
POST https://{接入点IP:8102}/v2/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/tenant/
contents

{
  "log_time_ns": 1586850540000000000,
  "contents": [
    "Fri Feb 1 07:48:04 UTC 2019 0\n",
    "Sat April 18 16:04:04 UTC 2019"
  ],
  "labels": {
    "user_tag": "string"
  }
}
```

响应示例

状态码：200

日志上报成功。

```
{
  "errorCode": "SVCSTG.ALS.200.200",
```



```
"error_msg": "Report success.",
"result": null
}
```

状态码：401

在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。

```
{
  "errorCode": "SVCSTG.ALS.403.105",
  "error_msg": "Project id is invalid.",
  "result": null
}
```

状态码

状态码	描述
200	日志上报成功。
400	非法请求。建议根据error_msg直接修改该请求，不要重试该请求。
401	在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。
500	系统内部错误。
503	被请求的服务无效，服务不可用。

4.6.3 上报高精度日志接口参考

功能介绍

该接口用于主机上报租户日志给LTS。

接入点IP可在LTS控制台安装ICAgent的安装命令中获取，端口为8102，调用时使用该参数请参见[请求示例](#)。

每次上报的时候，每条日志都必须带一个纳秒级的时间戳。在LTS界面查看日志的时候，会按照时间戳排序展示在页面上。关于上报日志的约束与限制请参考[日志读写限制](#)。

URI

POST /v2/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/tenant/contents/high-accuracy

表 4-67 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见： 获取项目ID 。

参数	是否必选	参数类型	描述
log_group_id	是	String	日志组ID，获取方式请参见： 获取日志组ID和日志流ID 。
log_stream_id	是	String	日志流ID，获取方式请参见： 获取日志组ID和日志流ID 。 每个logstream写入速率最大不能超过100MB/S，超过此规格可能会导致日志丢失。

请求参数

表 4-68 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	从IAM获取的用户Token，获取方式请参见： 获取用户Token 。
Content-Type	是	String	用于定义消息体的格式，该字段填为：application/json;charset=UTF-8。
Content-Encoding	否	String	日志压缩格式。 枚举值： • GZIP • SNAPPY • gzip • snappy

表 4-69 请求 Body 参数

参数	是否必选	参数类型	描述
contents	是	Array of LogContents	包含上报时间戳的日志内容列表。
labels	是	Object	用户自定义label。 请不要将字段名称设置为 内置保留字段 ，否则可能会造成字段名称重复、查询不精确等问题。
tenant_project_id	否	String	租户项目ID，获取方式请参见： 获取项目ID 。

表 4-70 LogContents

参数	是否必选	参数类型	描述
log_time_ns	是	Long	日志数据上报时间，UTC时间（纳秒）。通过接口上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。
log	是	String	日志内容。

响应参数

状态码为200时：

表 4-71 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。
error_msg	String	调用失败响应信息描述。
result	String	响应结果。

状态码为400时：

表 4-72 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。
error_msg	String	调用失败响应信息描述。
result	String	响应结果。

状态码为401时：

表 4-73 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。
error_msg	String	调用失败响应信息描述。

参数	参数类型	描述
result	String	响应结果。

状态码为500时：

表 4-74 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。
error_msg	String	调用失败响应信息描述。
result	String	响应结果。

状态码为503时：

表 4-75 响应 Body 参数

参数	参数类型	描述
result	String	被请求的服务无效，服务不可用。

请求示例

```
POST https://{接入点IP:8102}/v2/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/tenant/contents/high-accuracy

{
  "contents": [
    {
      "log_time_ns": 1586850540000000000,
      "log": "Fri Feb 15 15:48:04 UTC 2019"
    },
    {
      "log_time_ns": 1586850540000000001,
      "log": "Sat April 18 16:04:04 UTC 2019"
    }
  ],
  "labels": {
    "user_tag": "string"
  }
}
```

响应示例

状态码：200

日志上报成功。

```
{
  "errorCode": "SVCSTG.ALS.200.200",
  "error_msg": "Report success.",
}
```

```
"result": null
}
```

状态码：401

在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。

```
{
  "errorCode": "SVCSTG.ALS.403.105",
  "error_msg": "Project id is invalid.",
  "result": null
}
```

状态码

状态码	描述
200	日志上报成功。
400	非法请求。建议根据error_msg直接修改该请求，不要重试该请求。
401	在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。
500	系统内部错误。
503	被请求的服务无效，服务不可用。

4.7 使用 SDK 接入 LTS

4.7.1 云日志服务 Java SDK

云日志服务SDK提供了Java语言上报日志功能。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请用户根据所在区域，选择RegionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（project id），详细步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。
- 已安装Java开发环境。云日志服务Java SDK支持JDK1.8及以上，您可以执行java -version命令检查您已安装的Java版本。如未安装，可以从[Java官方网站下载安装包](#)进行安装。
- 使用Java SDK进行跨云/本地上报日志（此功能仅适用于开发调测，对于上报性能不做保障）：仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一局点。

- 通过Java SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

日志上报方式

支持以下上报方式：标准日志、**结构化日志（新版）**、结构化日志。推荐您使用**结构化日志（新版）**上报，更加灵活可变，性能更好。

1. 标准日志：

- 批量上报日志，一次可以上报多条日志。
- 字段log表示原始日志，即一条日志是一个字符串。
- 字段log_time_ns表示此条日志上报时间，单位ns纳秒，便于在LTS页面查看日志时按照时间排序。
- 可以将日志按照不同的labels标签进行分类上报。

上报一批日志结构体如下：

```
[{
  "contents": [{
    "log": "log content1",
    "log_time_ns": 1737527157333902200
  }, {
    "log": "log content2",
    "log_time_ns": 1737527157333914100
  }],
  "labels": "{\"lts-test-count\":\"2\"}"
}, {
  "contents": [{
    "log": "log content3",
    "log_time_ns": 1737527157333986200
  }, {
    "log": "log content4",
    "log_time_ns": 1737527157333987400
  }],
  "labels": "{\"lts-test-count\":\"2\"}"
}]
```

2. 推荐使用结构化日志（新版），最低SDK版本为1.1.3。

说明

目前该方式仅支持白名单用户使用，如有需要，请[提交工单](#)申请开通。

- 支持批量日志上报、单条日志上报。
- 字段mContents表示一条日志。
- 字段mKey表示当前这条日志中的一个key。
- 字段mValue表示当前这条日志中的一个key对应的value值。
- 字段mLogTime表示此条日志上报时间，单位ms毫秒。

上报一批日志结构体如下：

```
[{
  "mContents": [{
    "mKey": "content_key_1",
    "mValue": "content_value_1"
  }, {
    "mKey": "content_key_2",
    "mValue": "content_value_2"
  }, {
    "mKey": "content",
    "mValue": "sdk-new-struct-log1"
  }],
  "mLogTime": 1744159440780
}, {
```

```
"mContents": [{
  "mKey": "content_key_1",
  "mValue": "content_value_1"
}, {
  "mKey": "content_key_2",
  "mValue": "content_value_2"
}, {
  "mKey": "content", // 选填，云日志服务内部字段，此字段代表原始日志
  "mValue": "sdk-new-struct-log2"
}],
"mLogTime": 1744159440780
}]
```

3. 结构化日志：

说明

目前该方式仅支持白名单用户使用，如有需要，请[提交工单](#)申请开通。

- 批量上报日志，一次可以上报多条日志。
- 字段contents表示某几条日志。单条日志使用K-V结构，即一条日志是一个JSON体。content为LTS保留字段，用来表示原始日志，可以不上报。
- 字段time表示某几条日志上报时间，单位ms毫秒，LTS会将这几条日志在ms毫秒的基础上拓展为ns纳秒保存。这样会丢失这几条日志的先后顺序，可能会影响到您在LTS页面查看日志的先后顺序。
- 字段labels表示这批日志公共的标签。
- 字段path表示这批日志的路径。
- 字段source表示这批日志的来源。

上报一批日志结构体如下：

```
{
  "labels": {
    "label": "label"
  },
  "logs": [{
    "contents": [{
      "k1": "v1",
      "k2": "v2",
      "content": "log content1"
    }, {
      "k3": "v3",
      "k4": "v4",
      "content": "log content2"
    }
  ],
  "time": 1721784021037
}, {
  "contents": [{
    "k5": "v5",
    "k6": "v6",
    "content": "log content3"
  }
  ],
  "time": 1721784021038
}, ],
"path": "path",
"source": "source"
}
```

步骤一：安装 SDK

在Maven项目中加入依赖项。

1. 使用Maven构建项目时，settings.xml文件的mirrors节点需要增加[华为外部开源仓](#)。如下：


```
<mirror>
  <id>huaweicloud</id>
  <mirrorOf>*</mirrorOf>
  <url>https://repo.huaweicloud.com/repository/maven/</url>
</mirror>
```

- 您可以在[华为外部开源仓](#)中获取日志服务Java SDK依赖的最新版本。
- 在Maven工程中使用云日志服务Java SDK，只需在pom.xml中加入相应依赖即可，Maven项目管理工具会自动下载相关JAR包。在<dependencies>中加入如下内容：

推荐使用最新版本version：1.1.3，第2步中查看SDK版本。

```
<dependency>
  <groupId>io.github.huaweicloud</groupId>
  <artifactId>lts-sdk-common</artifactId>
  <version>version</version>
</dependency>
<dependency>
  <groupId>io.github.huaweicloud</groupId>
  <artifactId>lts-sdk-java</artifactId>
  <version>version</version>
</dependency>
```

步骤二：创建 Producer

由于Producer会消耗用户的ECS资源，请将Producer变为公共资源（单例）。发送日志时，仅需要调用producer.send()方法即可。appender支持配置永久AK、SK。

创建Producer的示例参考如下，修改示例代码时参考[表4-76](#)。

```
import com.huaweicloud.lts.appender.JavaSDKAppender;
import com.huaweicloud.lts.producer.Producer;

public class ProducerUtil {

    private static final Producer producer;

    static {
        /* 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文
        存放，使用时解密，确保安全；
        本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量
        HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。*/
        String ak = System.getenv("HUAWEICLOUD_SDK_AK");
        String sk = System.getenv("HUAWEICLOUD_SDK_SK");
        // 构建appender
        JavaSDKAppender appender = JavaSDKAppender.custom()
            // 华为云账号的项目ID（project id）
            .setProjectId("xxx")
            // 华为云账号的AK
            .setAccessKeyId(ak)
            // 华为云账号的SK
            .setAccessKeySecret(sk)
            // 华为云STS-Token，若使用临时STS-Token，则AK、SK需要填写与STS-Token配套的临时AK、SK
            // .setSecurityToken("")
            // 云日志服务的区域
            .setRegionName("xxx")
            // 单个Appender能缓存的日志大小上限
            .setTotalSizeInBytes(104857600)
            // producer发送日志时阻塞时间
            .setMaxBlockMs(0L)
            // 执行日志发送任务的线程池大小
            .setIoThreadCount(8)
            // producer发送单批日志量上限
            .setBatchSizeThresholdInBytes(524288)
            // producer发送单批日志条数上限
            .setBatchCountThreshold(4096)
            // producer发送单批日志等待时间
            .setLingerMs(2000)
```

```
// producer发送日志失败后重试次数
.setRetries(5)
// 首次重试的退避时间
.setBaseRetryBackoffMs(500L)
// 重试的最大退避时间
.setMaxRetryBackoffMs(500L)
// 默认false, true: 可以跨云上报日志, false: 仅能在华为云ecs主机上报日志
// .setEnableLocalTest(false)
.builder();
// 获取发送日志的producer
producer = appender.getProducer();
}

public static Producer getProducer() {
    return producer;
}
}
```

appender支持配置临时AK、SK、STS-Token。如果使用临时AK、SK、STS-Token，需要定期（在过期前）创建新的ProjectConfig，然后调用Producer.putProjectConfig()方法进行更新。示例代码如下：

```
// 如果使用临时AK、SK、STS-Token，需要定期（在过期前）创建新的ProjectConfig，然后调用
Producer.putProjectConfig()方法进行更新。
ProjectConfig projectConfig = new ProjectConfig("projectId", "regionName", "accessKeyId",
"accessKeySecret", "securityToken");
producer.putProjectConfig(projectConfig);
```

表 4-76 Producer 参数配置表

参数名称	描述	类型	是否需要填写	默认值
projectId	华为云账号的项目ID（project id）。	String	必填	-
accessKeyId	华为云账号的AK。认证用的AK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
accessKeySecret	华为云账号的SK。认证用的SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
securityToken	用户可以通过华为云IAM服务来获取临时访问密钥（临时AK/SK）和securitytoken。如果要使用临时访问密钥，则临时AK、SK、STS-Token为一套访问密钥，必须配合使用。	String	选填	-
regionName	云日志服务的区域。	String	必填	-
totalSizeInBytes	单个producer实例能缓存的日志大小上限。	int	选填	104857600（即100MB）

参数名称	描述	类型	是否需要填写	默认值
maxBlockMs	如果producer可用空间不足，调用者在send方法上的最大阻塞时间，单位为毫秒。默认为60000毫秒，建议为0毫秒。 - 当maxBlockMs值≥ 0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 - 当maxBlockMs值$= -1$时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60000毫秒
ioThreadCount	执行日志发送任务的线程池大小。	int	选填	客户机器空闲CPU数量，但一定大于1
batchSizeThresholdInBytes	当一个ProducerBatch中缓存的日志大小大于等于batchSizeThresholdInBytes时，该ProducerBatch缓存的日志将被发送到LTS。	int	选填	524288（即0.5MB）
batchCountThreshold	当一个ProducerBatch中缓存的日志条数大于等于batchCountThreshold时，该ProducerBatch缓存的日志将被发送到LTS。	int	选填	4096
lingerMs	一个ProducerBatch从创建到可发送的逗留时间。	int	选填	2s
retries	如果某个ProducerBatch首次发送失败，能够对其重试的次数，建议为5次。如果retries小于等于0，该ProducerBatch首次发送失败后将直接进入失败队列。	int	选填	10
baseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1s
maxRetryBackoffMs	重试的最大退避时间。	long	选填	50s

参数名称	描述	类型	是否需要填写	默认值
enableLocalTest	是否开启跨云上报日志。（此功能仅适用于开发调测，对于上报性能不做保障） - 选true时，开启跨云后用户能通过公网（仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一）访问。 - 选false时，关闭跨云后用户是通过华为云当前区域主机访问。	boolean	选填	false
proxyIp	使用自定义ip端口上报日志。	String	选填	-

步骤三：构建发送日志结构体

- 标准日志和结构化日志使用的日志结构体：

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.alibaba.fastjson.JSONObject;
import com.huaweicloud.lts.producer.model.log.LogContent;
import com.huaweicloud.lts.producer.model.log.LogItem;
import com.huaweicloud.lts.producer.model.log.StructLog;
import com.huaweicloud.lts.producer.model.log.StructLogItem;

public class LogUtil {

    private static final Logger LOGGER = LoggerFactory.getLogger(ProducerUtil.class);

    public static void doSomething() throws InterruptedException {
        LOGGER.info("Before doSomething");
        Thread.sleep(30000);
        LOGGER.info("After doSomething");
    }

    // 标准日志模型
    public static List<LogItem> getLogItems(int size) {
        List<LogItem> logItemList = new ArrayList<>();
        logItemList.add(getLogItem(size));
        return logItemList;
    }

    private static LogItem getLogItem(int count) {
        LogItem logItem = new LogItem();

        // 客户可以自行定义日志标签, 如果不需要请传入一个空Map
        Map<String, String> labels = new HashMap<>();
        labels.put("labels-k", "labels-v");
        logItem.setLabels(JSONObject.toJSONString(labels));

        List<LogContent> contents = new ArrayList<>();
        for (int i = 1; i <= count; i++) {
            LogContent logContent = new LogContent();
```

```
// 日志产生时间, 当前时间ns(必须为ns纳秒)
long logTimeNs = System.currentTimeMillis() * 1000000L + System.nanoTime() % 1000000L;
logContent.setLogTimeNs(logTimeNs);
// 日志内容
logContent.setLog("java-sdk-log-" + i);
contents.add(logContent);
}
logItem.setContents(contents);

return logItem;
}

// 结构化日志模型
public static StructLogItem getStructLog(int size) {
    // 构建日志结构体
    StructLogItem structLogItem = new StructLogItem();
    // 设置此批日志公共标签
    structLogItem.setLabels(getPostLabels());
    // 添加日志List
    structLogItem.setLogs(getLogs(size));
    // 设置此批日志公共路径
    structLogItem.setPath("test-path");
    // 设置此批日志公共来源
    structLogItem.setSource("test-source");
    return structLogItem;
}

private static JSONObject getPostLabels() {
    JSONObject jsonObject = new JSONObject();
    jsonObject.put("common-k", "common-v");
    return jsonObject;
}

private static List<StructLog> getLogs(int size) {
    List<StructLog> logs = new ArrayList<>();
    for (int i = 1; i <= size; i++) {
        logs.add(getLog(size));
    }
    return logs;
}

private static StructLog getLog(int size) {
    StructLog structLog = new StructLog();
    // 日志产生时间, 当前时间ms值(必须为ms毫秒)
    long time = System.currentTimeMillis();
    structLog.setTime(time);

    List<JSONObject> contents = new ArrayList<>();
    for (int i = 1; i <= size; i++) {
        // 创建一条日志
        JSONObject jsonLog = new JSONObject();

        jsonLog.put("log-k", "log-v");
        // LTS保留字段"content", 指原始日志。
        jsonLog.put("content", "java-struct-log-" + i);

        contents.add(jsonLog);
    }

    structLog.setContents(contents);
    return structLog;
}
}
```

- 结构化日志（新版）的结构体如下，SDK版本1.1.3及以上才支持此功能。

```
// 批量日志
public static List<LtsStructLog> getNewBatchStructLog(int size) {
    List<LtsStructLog> batchStructLog = new ArrayList<>();
    for (int i = 1; i <= size; i++) {
```

```
// 创建一条日志
LtsStructLog ltsStructLog = new LtsStructLog();

ltsStructLog.pushBack("content_key_1", "content_value_1");
ltsStructLog.pushBack("content_key_2", "content_value_2");
// LTS保留字段content, 指原始日志。
ltsStructLog.pushBack("content", "sdk_new_struct_log");
// 日志产生时间, 当前时间ms值(必须为ms毫秒)
ltsStructLog.setLogTime(System.currentTimeMillis());

batchStructLog.add(ltsStructLog);
}
return batchStructLog;
}

// 单条日志
public static LtsStructLog getNewStructLog() {
    // 创建一条日志
    LtsStructLog ltsStructLog = new LtsStructLog();

    ltsStructLog.pushBack("content_key_1", "content_value_1");
    ltsStructLog.pushBack("content_key_2", "content_value_2");
    // LTS保留字段content, 指原始日志。
    ltsStructLog.pushBack("content", "sdk_new_struct_log");
    // 日志产生时间, 当前时间ms值(必须为ms毫秒)
    ltsStructLog.setLogTime(System.currentTimeMillis());
    return ltsStructLog;
}
```

步骤四：发送日志

由于Producer是异步发送日志，会有本地缓存，所以单个Producer只能选择标准日志或者结构化日志的一种上报，不能同时上报两种日志格式。

- 异步发送日志，无需返回值进行额外操作。

```
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.huaweicloud.lts.producer.Producer;
import com.huaweicloud.lts.producer.exception.ProducerException;

public class SamplePerformance {

    private static final Logger LOGGER = LoggerFactory.getLogger(SamplePerformance.class);

    public static void main(String[] args) throws InterruptedException {
        // 获取日志发送的producer
        Producer producer = ProducerUtil.getProducer();

        int sendThreadCount = Math.max(Runtime.getRuntime().availableProcessors(), 1);
        ExecutorService executorService = Executors.newFixedThreadPool(sendThreadCount);
        final CountDownLatch latch = new CountDownLatch(sendThreadCount);

        LOGGER.info("Test started.");
        long t1 = System.currentTimeMillis();
        int n = 100;
        for (int i = 1; i <= sendThreadCount; ++i) {
            executorService.submit(new Runnable() {
                @Override
                public void run() {
                    try {
                        for (int i = 0; i < n; ++i) {

                            // 上报标准日志
                            producer.send("log_group_id", "log_stream_id", LogUtil.getLogItems(1));
                            // 使用日志组名称，日志流名称发送日志。SDK版本最低必须为1.1.0支持此功能。
                        }
                    } catch (Exception e) {
                        LOGGER.error("Error occurred while sending log items.", e);
                    }
                }
            });
        }
        latch.await();
        LOGGER.info("Test finished. Total time: {} ms", System.currentTimeMillis() - t1);
    }
}
```

```
// producer.send("log_group_name", "log_stream_name", true, logItemList);
// 注意：方法中第3个参数，请填true，代表使用名称发送日志。如果是false，则是ID
发送日志。

// 上报结构化日志
// producer.send("log_group_id", "log_stream_id", LogUtil.getStructLog(1));
// 使用日志组名称，日志流名称发送日志。SDK版本最低必须为1.1.0支持此功能。
// producer.send("log_group_name", "log_stream_name", true, structLog);
// 注意：方法中第3个参数，请填true，代表使用名称发送日志。如果是false，则是ID
发送日志。

// 上报新版结构化日志
// 备注：上报参数中的"log_group" 默认表示日志组Id、"log_stream" 默认表示日志流
Id。
// 如果需要日志组名称和日志流名称上报日志，只需要在初始化appender时，设置参
数：setLogGroupStreamByName(true)
// 上报单条日志
// producer.sendStructLog("log_group", "log_stream", LogUtil.getNewStructLog());
// 上报一批日志
// producer.sendStructLog("log_group", "log_stream",
LogUtil.getNewBatchStructLog(1));
}
} catch (Exception e) {
    LOGGER.error("Failed to send log, e=", e);
} finally {
    latch.countDown();
}
}
});
}
latch.await();
Thread.sleep(30000);
long t2 = System.currentTimeMillis();
LOGGER.info("Test end.");
LOGGER.info("====Summary====");
LOGGER.info("Total count " + sendThreadCount * n + ".");
long timeCost = t2 - t1;
LOGGER.info("Time cost " + timeCost + " millis");
try {
    producer.close();
} catch (ProducerException e) {
    LOGGER.error("Failed to close producer, e=", e);
}
}
executorService.shutdown();
}
}
```

- 通过Future来执行额外操作。

Send 方法会返回一个ListenableFuture，它除了可以像普通future那样通过调用get方法阻塞获得发送结果外，还允许您注册回调方法（回调方法会在完成 future 设置后被调用）。以下代码片段展示了ListenableFuture的使用方法，用户需要为该future注册一个FutureCallback并将其投递到应用提供的线程池EXECUTOR_SERVICE中执行。

```
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicLong;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.google.common.util.concurrent.FutureCallback;
import com.google.common.util.concurrent.Futures;
import com.google.common.util.concurrent.ListenableFuture;
import com.huaweicloud.lts.producer.Producer;
import com.huaweicloud.lts.producer.Result;
import com.huaweicloud.lts.producer.exception.ProducerException;
import com.huaweicloud.lts.producer.exception.ResultFailedException;
```

```
import com.huaweicloud.lts.producer.model.log.LogItem;
import com.huaweicloud.lts.producer.model.log.StructLogItem;

public class SampleProducerWithFuture {

    private static final Logger LOGGER = LoggerFactory.getLogger(SampleProducerWithFuture.class);

    private static final ExecutorService EXECUTOR_SERVICE = Executors.newFixedThreadPool(
        Math.max(Runtime.getRuntime().availableProcessors(), 1));

    public static void main(String[] args) throws InterruptedException {
        Producer producer = ProducerUtil.getProducer();
        int n = 100;
        int size = 20;

        final AtomicLong completed = new AtomicLong(0);

        for (int i = 0; i < n; ++i) {
            try {
                // 上报标准日志
                List<LogItem> logItems = LogUtil.getLogItems(size);
                ListenableFuture<Result> f = producer.send("log_group_id", "log_stream_id", logItems);
                Futures.addCallback(f, new SampleFutureCallback("log_group_id", "log_stream_id",
logItems, completed),
                    EXECUTOR_SERVICE);

                // 上报结构化日志
                // StructLogItem structLog = LogUtil.getStructLog(size);
                // ListenableFuture<Result> f = producer.send("log_group_id", "log_stream_id", structLog);
                // Futures.addCallback(f, new SampleFutureCallback("log_group_id", "log_stream_id",
structLog, completed),
                //     EXECUTOR_SERVICE);

                // 上报新版结构化日志
                // List<LtsStructLog> newBatchStructLog = LogUtil.getNewBatchStructLog(1);
                // ListenableFuture<Result> f = producer.sendStructLog("log_group", "log_stream",
newBatchStructLog);
                // Futures.addCallback(f, new SampleFutureCallback("log_group", "log_stream",
completed, newBatchStructLog),
                //     EXECUTOR_SERVICE);

            } catch (Exception e) {
                LOGGER.error("Failed to send logs, e=", e);
            }
        }

        LogUtil.doSomething();

        try {
            producer.close();
        } catch (ProducerException e) {
            LOGGER.info("Failed to close producer, e=", e);
        }

        EXECUTOR_SERVICE.shutdown();
        while (!EXECUTOR_SERVICE.isTerminated()) {
            EXECUTOR_SERVICE.awaitTermination(100, TimeUnit.MILLISECONDS);
        }
        LOGGER.info("All log complete, completed={}", completed.get());
    }

    private static final class SampleFutureCallback implements FutureCallback<Result> {

        private static final Logger LOGGER = LoggerFactory.getLogger(SampleFutureCallback.class);

        private final String logGroupId;

        private final String logStreamId;
```



```
private List<LogItem> logItems = null;

private StructLogItem structLog = null;

private List<LtsStructLog> newBatchStructLog = null;

private final AtomicLong completed;

SampleFutureCallback(String logGroupId, String logStreamId, List<LogItem> logItems,
AtomicLong completed) {
    this.logGroupId = logGroupId;
    this.logStreamId = logStreamId;
    this.logItems = logItems;
    this.completed = completed;
}

SampleFutureCallback(String logGroupId, String logStreamId, StructLogItem structLog,
AtomicLong completed) {
    this.logGroupId = logGroupId;
    this.logStreamId = logStreamId;
    this.structLog = structLog;
    this.completed = completed;
}

SampleFutureCallback(String logGroupId, String logStreamId, AtomicLong completed,
List<LtsStructLog> newBatchStructLog) {
    this.logGroupId = logGroupId;
    this.logStreamId = logStreamId;
    this.newBatchStructLog = newBatchStructLog;
    this.completed = completed;
}

@Override
public void onSuccess(Result result) {
    LOGGER.info("Send logs successfully.");
    completed.getAndIncrement();
}

@Override
public void onFailure(Throwable t) {
    if (t instanceof ResultFailedException) {
        Result result = ((ResultFailedException) t).getResult();
        LOGGER.error("Failed to send logs, logGroupId={}, logStreamId={}, result={}", logGroupId,
logStreamId,
        result);
    } else {
        LOGGER.error("Failed to send log, e=", t);
    }
    completed.getAndIncrement();
}
}
```

- 通过Callback来执行额外操作。

Callback由producer内部线程负责执行，并且只有在执行完毕后数据“占用”的空间才会释放。为了不阻塞producer造成整体吞吐量的下降，要避免在callback里执行耗时的操作。不建议在callback中调用send方法进行重试。SDK自带重试功能。

```
import java.util.List;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicLong;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.huaweicloud.lts.producer.Callback;
import com.huaweicloud.lts.producer.Producer;
import com.huaweicloud.lts.producer.Result;
```

```
import com.huaweicloud.lts.producer.exception.ProducerException;
import com.huaweicloud.lts.producer.model.log.LogItem;
import com.huaweicloud.lts.producer.model.log.StructLogItem;

public class SampleProducerWithCallback {

    private static final Logger LOGGER = LoggerFactory.getLogger(SampleProducerWithCallback.class);

    private static final ExecutorService EXECUTOR_SERVICE = Executors.newFixedThreadPool(
        Math.max(Runtime.getRuntime().availableProcessors(), 1));

    public static void main(String[] args) throws InterruptedException {
        Producer producer = ProducerUtil.getProducer();
        int nTask = 100;
        int size = 20;

        final AtomicLong completed = new AtomicLong(0);

        final CountDownLatch latch = new CountDownLatch(nTask);

        for (int i = 0; i < nTask; ++i) {
            EXECUTOR_SERVICE.submit(() -> {
                try {
                    // 上报标准日志
                    List<LogItem> logItems = LogUtil.getLogItems(size);
                    producer.send("log_group_id", "log_stream_id", logItems,
                        new SampleCallback("log_group_id", "log_stream_id", logItems, completed));

                    // 上报结构化日志
                    // StructLogItem structLog = LogUtil.getStructLog(size);
                    // producer.send("log_group_id", "log_stream_id", structLog,
                    //     new SampleCallback("log_group_id", "log_stream_id", structLog, completed));

                    // 上报新版结构化日志
                    // List<LtsStructLog> newBatchStructLog = LogUtil.getNewBatchStructLog(1);
                    // producer.sendStructLog("log_group", "log_stream", newBatchStructLog,
                    //     new SampleCallback("log_group", "log_stream", completed, newBatchStructLog));
                } catch (Exception e) {
                    LOGGER.error("Failed to send log, e=", e);
                } finally {
                    latch.countDown();
                }
            });
        }
        latch.await();
        EXECUTOR_SERVICE.shutdown();

        LogUtil.doSomething();

        try {
            producer.close();
        } catch (InterruptedException e) {
            LOGGER.warn("The current thread has been interrupted from close.");
        } catch (ProducerException e) {
            LOGGER.info("Failed to close producer, e=", e);
        }

        LOGGER.info("All log complete, completed={}", completed.get());
    }

    private static final class SampleCallback implements Callback {

        private static final Logger LOGGER = LoggerFactory.getLogger(SampleCallback.class);

        private final String logGroupId;

        private final String logStreamId;

        private List<LogItem> logItems = null;
    }
}
```

```
private StructLogItem structLog = null;

private List<LtsStructLog> newBatchStructLog = null;

private final AtomicLong completed;

SampleCallback(String logGroupId, String logStreamId, List<LogItem> logItems, AtomicLong
completed) {
    this.logGroupId = logGroupId;
    this.logStreamId = logStreamId;
    this.logItems = logItems;
    this.completed = completed;
}

SampleCallback(String logGroupId, String logStreamId, StructLogItem structLog, AtomicLong
completed) {
    this.logGroupId = logGroupId;
    this.logStreamId = logStreamId;
    this.structLog = structLog;
    this.completed = completed;
}

SampleCallback(String logGroupId, String logStreamId, AtomicLong completed,
List<LtsStructLog> newBatchStructLog) {
    this.logGroupId = logGroupId;
    this.logStreamId = logStreamId;
    this.completed = completed;
    this.newBatchStructLog = newBatchStructLog;
}

@Override
public void onCompletion(Result result) {
    try {
        if (result.isSuccessful()) {
            LOGGER.info("Send log successfully.");
        } else {
            LOGGER.error("Failed to send logs, logGroupId={}, logStreamId={}, result={}",
logGroupId, logStreamId,
result);
        }
    } finally {
        completed.getAndIncrement();
    }
}
}
```

步骤五：关闭 Producer

当您已经没有数据需要发送或者当前进程准备退出时，需要关闭Producer，目的是让Producer中缓存的数据全部被处理。目前，Producer提供安全关闭和有限关闭两种模式。

1. 安全关闭（推荐）：建议您使用安全关闭。安全关闭对应的方法是close()，SDK会等到Producer中缓存的数据全部被处理、线程全部停止、注册的callback全部执行、返回future全部被设置后才会关闭Producer。
2. 有限关闭：如果您的callback在执行过程中有可能阻塞，但您又希望close方法能在短时间内返回，可以使用有限关闭。有限关闭对应的方法是close(long timeoutMs)，如果超过指定的timeoutMs后Producer仍未完全关闭，它会抛出IllegalStateException异常，这意味着缓存的数据可能还没来得及处理就被丢弃，用户注册的Callback也可能不会被执行。

Producer 性能基线

上报日志时，请参考如下参数的测试性能基线，若超出基线值，可能会导致日志上报异常。

ECS虚拟机配置参考如下：

- 实例规格：通用计算增强型c7.xlarge.2
- CPU：4 vCPU
- 内存：8 GB
- 基准带宽：100 Mbit/s
- OS：Huawei Cloud EulerOS release 2.0
- JVM：OpenJDK 64-Bit Server VM (build 17.0.7+7, mixed mode)

测试程序说明（单个producer）：

- totalSizeInBytes: 104857600
- maxBlockMs: 0
- batchSizeThresholdInBytes: 1048576
- batchCountThreshold: 40960
- lingerMs: 2000
- ioThreadCount: 具体用例中调整
- JVM初始堆/最大堆大小: 1GB
- 发送日志总条数: 100,000,000
- 发送日志总大小: 约50GB

按照参数基线值设置后，使用华为云ECS机器作为日志上报环境，通过华为云内网服务入口进行上报。

标准日志SDK上报性能基线测试结果参考：

- 上报日志格式：测试上报一批日志，包含4条日志，总大小约为2.2KB。为了模拟数据的随机性，测试使用的日志数据为随机字符串，单条日志大小约为510字节。

```
[
  {
    "contents": [{
      "log": "随机字符串510字节",
      "log_time_ns": 1637527157333902200
    }, {
      "log": "随机字符串510字节",
      "log_time_ns": 1637527157333902200
    }
  ]
}, {
  "contents": [{
    "log": "随机字符串510字节",
    "log_time_ns": 1637527157333902200
  }, {
    "log": "随机字符串510字节",
    "log_time_ns": 1737527157333987400
  }
]
}
```

- 性能基线参考如下：

表 4-77 标准日志 SDK 上报性能基线

IO 线程数量	数据吞吐量	数据吞吐速率	CPU 使用率
2	7.8 MB/S	1.5W 条/S	9 %
4	15.4 MB/S	3.1W 条/S	19 %
6	19.7 MB/S	3.9W 条/S	27 %

结构化日志SDK上报性能基线测试结果参考：

- 上报日志格式：测试上报一批日志，包含4条日志，总大小约为2.2KB。单条日志包含10个键值对以及content、time字段。为了模拟数据的随机性，测试使用的数据为随机字符串，单条日志大小约为550字节。

```
{
  "logs": [
    {
      "contents": [
        {
          "content": "sdk-log-new-struct-1",
          "content_key_1": "XmGFubcemwrceBWbZYRBTgohxfxFih",
          "...": "...",
          "content_key_10": "amchrqwPdigHopmAkNLvJtNxgiPUzh"
        },
        {
          "content": "sdk-log-new-struct-2",
          "content_key_1": "zOPDFRCNYsVznSgtnFejWFbaxklkMQ",
          "...": "...",
          "content_key_10": "mLzpbYcumXslgYtQIbzisoACLtUgwS"
        }
      ],
      "time": 1645374890235
    },
    {
      "contents": [
        {
          "content": "sdk-log-new-struct-3",
          "content_key_1": "SaGsfdRqskJaHlciNAUXFyxiqCAqXe",
          "...": "...",
          "content_key_10": "wMQNuoVWonxVSsRsocQoDkEjcjiPio"
        },
        {
          "content": "sdk-log-new-struct-4",
          "content_key_1": "bHDjNmAvdiLAvWdxoETANqCYxhVMMk",
          "...": "...",
          "content_key_10": "scsxtRrPUFYVARzOvbCxSofYZBsFV"
        }
      ],
      "time": 1645374890235
    }
  ]
}
```
- 性能基线参考如下：

表 4-78 结构化日志 SDK 上报性能基线

IO 线程数量	数据吞吐量	数据吞吐速率	CPU 使用率
2	18.7 MB/S	3.5W 条/S	13 %
4	34.6 MB/S	6.4W 条/S	25 %
6	42.7 MB/S	7.8W 条/S	32 %

新版本结构化日志SDK上报性能基线测试结果参考：

- 上报日志格式：测试中使用的日志包含10个键值对以及content、logTime 字段。为了模拟数据的随机性，测试使用的数据为随机字符串。单条日志大小约为540字节。

```
content: sdk-log-new-struct-1
content_key_1: sshyaqKCfrAPCMpdlxroPuCedeuJ
content_key_2: RVFtqFBjBtTAHVvdHBYQsDsoogJc
...
content_key_9: zLKeuxDnzGtupeZrQKKilkQemXvX
content_key_10: fYxmtYxKNfBhRfqMbZEOfimlsAlo
logTime: 1645390242169
```

- 性能基线参考如下：

表 4-79 新版本结构化日志 SDK 上报性能基线

IO 线程数量	数据吞吐量	数据吞吐速率	CPU 使用率
2	23.7MB/S	4.5W条/s	11%
4	45.6MB/S	8.7W条/s	23%
6	54.7MB/S	10.3W条/s	30%

总结如下：

- CPU时间主要花费在对象的序列化和压缩上，在吞吐量较高的情况下CPU使用率比较高。但在日常环境中，单机数据流量均值为100KB/S，因此造成的CPU消耗几乎可以忽略不计。
- 增加IO线程数量可以显著提高吞吐量，尤其是当IO线程数量少于可用处理器个数时。
- 调整totalSizeInBytes对吞吐量影响不够显著，增加totalSizeInBytes会造成更多的CPU消耗，建议使用默认值。
- 当日志上报量超过单个producer时：
 - 建议拆分日志流，使用多个producer上报日志，分摊流量，以保障SDK处于正常上报状态。
 - 如果maxBlockMs为0时，SDK处于非阻塞状态，会触发保护机制自动降级，可能会对部分日志做丢弃处理。
 - 如果maxBlockMs大于0时，SDK处于阻塞状态，阻塞时间为maxBlockMs，可能会造成producer.send()发送日志方法处于阻塞状态。

参数获取方式

- 区域表

表 4-80 区域表

区域名称	区域
华北-北京二	cn-north-2
华北-北京四	cn-north-4

区域名称	区域
华北-北京一	cn-north-1
华东-上海二	cn-east-2
华东-上海一	cn-east-3
华南-广州	cn-south-1
华南-深圳	cn-south-2
西南-贵阳一	cn-southwest-2
亚太-新加坡	ap-southeast-3

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 ，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.2 云日志服务 Log4j2 SDK

云日志服务Java SDK提供了与log4j2适配的扩展插件，可以直接在log4j2中配置华为云 appender，将通过log4j2产生的日志直接上报至云日志服务。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请用户根据所在区域，选择RegionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（project id），详细步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。
- 已安装Java开发环境。云日志服务Java SDK支持JDK1.8及以上，您可以执行`java -version`命令检查您已安装的Java版本。如未安装，可以从[Java官方网站下载安装包](#)进行安装。
- Log4j2 SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一。
- 通过Log4j2 SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

安装 SDK

在Maven项目中加入依赖项。

1. 使用Maven构建项目时，settings.xml文件的mirrors节点需要增加[华为外部开源仓](#)。如下：

```
<mirror>
  <id>huaweicloud</id>
  <mirrorOf>*</mirrorOf>
  <url>https://repo.huaweicloud.com/repository/maven/</url>
</mirror>
```

2. 您可以在[华为外部开源仓](#)中获取云日志服务log4j2 SDK依赖的最新版本。
3. 在Maven工程中使用云日志服务log4j2 SDK，只需在pom.xml中加入相应依赖即可，Maven项目管理工具会自动下载相关JAR包。在<dependencies>中加入如下内容：

推荐使用最新版本version：1.1.3，第2步中查看SDK版本。

```
<dependency>
  <groupId>io.github.huaweicloud</groupId>
  <artifactId>lts-sdk-common</artifactId>
  <version>version</version>
</dependency>
<dependency>
  <groupId>io.github.huaweicloud</groupId>
  <artifactId>lts-sdk-log4j2</artifactId>
  <version>version</version>
</dependency>
```

示例代码

1. log4j2.xml示例如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">

  <Appenders>
    <!-- projectId: 华为云账号的项目ID-->
    <!-- logGroupId: 华为云日志服务日志组ID-->
    <!-- logStreamId: 华为云日志服务日志流ID-->
    <!-- accessKeyId: 华为云账号的AK, 认证用的AK硬编码到代码中或者明文存储都有很大的安全风险,
    建议密文存放, 使用时解密, 确保安全-->
    <!-- accessKeySecret: 华为云账号的SK, 认证用的SK硬编码到代码中或者明文存储都有很大的安全风险,
    建议密文存放, 使用时解密, 确保安全-->
    <!-- regionName: 云日志服务的区域-->
    <!-- totalSizeInBytes: 单个Appender能缓存的日志大小上限-->
    <!-- maxBlockMs: producer发送日志时阻塞时间-->
    <!-- ioThreadCount: 执行日志发送任务的线程池大小-->
    <!-- batchSizeThresholdInBytes: producer发送单批日志量上限-->
    <!-- batchCountThreshold: producer发送单批日志条数上限-->
    <!-- lingerMs: producer发送单批日志等待时间-->
    <!-- retries: producer发送日志失败后重试次数-->
    <!-- baseRetryBackoffMs: 首次重试的退避时间-->
    <!-- maxRetryBackoffMs: 重试的最大退避时间-->
    <!-- giveUpExtraLongSingleLog: 过1M的日志, 拆分后放弃1M以外的-->
    <!-- enableLocalTest: 默认false, true: 可以跨云上报日志, false: 仅能在华为云ecs主机上报日志-->
    <!-- isLogGroupStreamByName: 是否使用日志组/日志流名称上报日志-->
    <ltsLog name="log4j2sdkappender"
      projectId="xxx"
      logGroupId="xxx"
      logStreamId="xxx"
      accessKeyId="xxx"
      accessKeySecret="xxx"
      regionName="xxx"
      totalSizeInBytes="104857600"
      maxBlockMs="0"
      ioThreadCount="8"
      batchSizeThresholdInBytes="524288"
      batchCountThreshold="4096"
      lingerMs="2000"
      retries="5"
      baseRetryBackoffMs="500"
      maxRetryBackoffMs="500"
      giveUpExtraLongSingleLog="true"
      enableLocalTest="false"
```



```
        isLogGroupStreamByName="false"
    >
    <PatternLayout pattern="%d %-5level [%thread] %logger{0}: %msg"/>
    </PatternLayout>
</Appenders>

<Loggers>
    <Root level="WARN">
        <AppenderRef ref="log4j2sdkappender"/>
    </Root>
</Loggers>

</Configuration>
```

2. 编写代码产生的日志如下：

```
package com.huawei.log;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class TestLog4j2AppenderInfo {

    private static final Logger LOGGER = LogManager.getLogger(TestLog4j2AppenderInfo.class);

    public static void main(String[] args) {
        LOGGER.info("test~log4j2~~~info");
    }
}
```

producer 性能基线

上报日志时，请参考如下参数的测试性能基线，若超出基线值，可能会导致日志上报异常。

ECS虚拟机配置参考如下：

- 实例规格：通用计算增强型 c7.xlarge.2
- CPU：4 vCPU
- 内存：8 GB
- 基准带宽：100 Mbit/s
- OS：Huawei Cloud EulerOS release 2.0
- JVM：OpenJDK 64-Bit Server VM (build 17.0.7+7, mixed mode)

测试程序说明（单个producer）：

- totalSizeInBytes: 104857600
- maxBlockMs: 0
- batchSizeThresholdInBytes: 1048576
- batchCountThreshold: 40960
- lingerMs: 2000
- ioThreadCount: 具体用例中调整
- JVM 初始堆/最大堆大小: 1GB
- 发送日志总条数: 100,000,000
- 发送日志总大小: 约50GB

按照参数基线值设置后，使用华为云ECS机器作为日志上报环境，通过华为云内网服务入口进行上报。

SDK上报性能基线测试结果参考：

- 上报日志：测试中使用的单条日志大小约为550字节。为了模拟数据的随机性，测试使用的数据为随机字符串。
- 性能基线参考如下：

表 4-81 性能基线

IO 线程数量	数据吞吐量	数据吞吐速率	CPU 使用率
2	8 MB/S	1.6W条/s	15%
4	15.2 MB/S	3W条/s	29%
6	20.7 MB/S	4.1W条/s	36%

总结如下：

- CPU时间主要花费在对象的序列化和压缩上，在吞吐量较高的情况下CPU使用率比较高。但在日常环境中，单机数据流量均值为100KB/S，因此造成的CPU消耗几乎可以忽略不计。
- 增加IO线程数量可以显著提高吞吐量，尤其是当IO线程数量少于可用处理器个数时。
- 调整totalSizeInBytes对吞吐量影响不够显著，增加totalSizeInBytes会造成更多的CPU消耗，建议使用默认值。
- 当日志上报量超过单个producer时：
 - a. 建议拆分日志流，使用多个producer上报日志，分摊流量，以保障SDK处于正常上报状态。
 - b. 如果maxBlockMs为0时，SDK处于非阻塞状态，会触发保护机制自动降级，可能会对部分日志做丢弃处理。
 - c. 如果maxBlockMs大于0时，SDK处于阻塞状态，阻塞时间为maxBlockMs，可能会造成SDK发送日志处于阻塞状态。

配置参数说明

- Appender配置参数说明。

表 4-82 Appender 配置参数说明

参数名称	描述	类型	是否需要填写	默认值
projectId	华为云账号的项目ID（project id）。	String	必填	-
accessKeyId	华为云账号的AK。认证用的AK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-

参数名称	描述	类型	是否需要填写	默认值
accessKeySecret	华为云账号的SK。认证用的SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
regionName	云日志服务的区域。	String	必填	-
logGroupId	LTS的日志组ID。	String	必填	-
logStreamId	LTS的日志流ID。	String	必填	-
totalSizeInBytes	单个producer实例能缓存的日志大小上限。	int	选填	104857600（即100MB）
maxBlockMs	如果producer可用空间不足，调用者在send方法上的最大阻塞时间，单位为毫秒。默认为60000毫秒，建议为0毫秒。 - 当maxBlockMs值≥ 0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 - 当maxBlockMs值$= -1$时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60000毫秒
ioThreadCount	执行日志发送任务的线程池大小。	int	选填	客户机器空闲CPU数量，但一定大于1
batchSizeThresholdInBytes	当一个ProducerBatch中缓存的日志大小大于等于batchSizeThresholdInBytes时，该ProducerBatch缓存的日志将被发送到LTS。	int	选填	524288（即0.5MB）
batchCountThreshold	当一个ProducerBatch中缓存的日志条数大于等于batchCountThreshold时，该ProducerBatch缓存的日志将被发送到LTS。	int	选填	4096
lingerMs	一个ProducerBatch从创建到可发送的逗留时间。	int	选填	2s

参数名称	描述	类型	是否需要填写	默认值
retries	如果某个ProducerBatch首次发送失败，能够对其重试的次数，建议为5次。如果retries/小于等于0，该ProducerBatch首次发送失败后将直接进入失败队列。	int	选填	10
baseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1s
maxRetryBackoffMs	重试的最大退避时间。	long	选填	50s
giveUpExtraLongSingleLog	超过1M的日志, 拆分后丢弃大于1M的数据。	boolean	选填	false
enableLocalTest	是否开启跨云上报日志。 (此功能仅适用于开发测试，对于上报性能不做保障) - 选true时，开启跨云后用户能通过公网（仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一）访问。 - 选false时，关闭跨云后用户是通过华为云当前区域主机访问。	boolean	选填	false
isLogGroupStreamByName	是否使用日志组/日志流名称上报日志（SDK版本最低必须为1.1.0支持此功能）。如果为true，表明使用名称上报，则logGroupId、logStreamId参数值分别填入日志组、日志流名称即可。	boolean	选填	false
proxyIp	使用自定义ip端口上报日志。	String	选填	-

参数获取方法

- 区域表

表 4-83 区域表

区域名称	区域
华北-北京二	cn-north-2
华北-北京四	cn-north-4
华北-北京一	cn-north-1
华东-上海二	cn-east-2
华东-上海一	cn-east-3
华南-广州	cn-south-1
华南-深圳	cn-south-2
西南-贵阳一	cn-southwest-2
亚太-新加坡	ap-southeast-3

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.3 云日志服务 LogBack SDK

云日志服务Java SDK提供了与logback适配的扩展插件，可以直接在logback中配置华为云appender，将通过logback产生的日志直接上报至云日志服务。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请用户根据所在区域，选择RegionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（project id），详细步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。
- 已安装Java开发环境。云日志服务Java SDK支持JDK1.8及以上，您可以执行**java -version**命令检查您已安装的Java版本。如未安装，可以从[Java官方网站下载安装包](#)进行安装。
- SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一。
- 支持将logback.xml变为logback-spring.xml（SpringBoot），且读取springProfile中的配置。
- 支持读取logback.xml、logback-spring.xml中的自定义标签。

- 通过LogBack SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。
- 建议设置上报周期小于1分钟（例如50秒）上报一次，设置1分钟或超过1分钟上报一次，可能会出现连接失败，因为SDK连接默认1分钟超时。

安装 SDK

在Maven项目中加入依赖项。

1. 使用Maven构建项目时，settings.xml文件的mirrors节点需要增加[华为外部开源仓](#)。如下：

```
<mirror>
  <id>huaweicloud</id>
  <mirrorOf>*</mirrorOf>
  <url>https://repo.huaweicloud.com/repository/maven/</url>
</mirror>
```

2. 您可以在[华为外部开源仓](#)中获取云日志服务LogBack SDK依赖的最新版本。
3. 在Maven工程中使用云日志服务LogBack SDK，只需在pom.xml中加入相应依赖即可，Maven项目管理工具会自动下载相关JAR包。在<dependencies>中加入如下内容：

推荐使用最新版本version：1.1.3，第2步中查看SDK版本。

```
<dependency>
  <groupId>io.github.huaweicloud</groupId>
  <artifactId>lts-sdk-common</artifactId>
  <version>version</version>
<exclusions>
  <exclusion>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
  </exclusion>
</exclusions>
</dependency>
<dependency>
  <groupId>io.github.huaweicloud</groupId>
  <artifactId>lts-sdk-logback</artifactId>
  <version>version</version>
</dependency>
```

示例代码

1. 编写logback.xml、logback-spring.xml中LogbackSDKAppender，示例如下：

```
<configuration scan="true" debug="false">
  <appender name="LogInfoTest" class="com.huaweicloud.lts.appender.LogbackSDKAppender">
    <!-- 华为云账号的项目ID ( project id ) -->
    <projectId>xxxxxxxxxxxxxxxxxxxx</projectId>
    <!-- 华为云日志服务日志组ID -->
    <logGroupId>xxxxxxxxxxxxxxxxxxxx</logGroupId>
    <!-- 华为云日志服务日志流ID -->
    <logStreamId>xxxxxxxxxxxxxxxxxxxx</logStreamId>
    <!-- 华为云账号的AK, 认证用的AK硬编码到代码中或者明文存储都有很大的安全风险, 建议密文存放,
    使用时解密, 确保安全 -->
    <accessKeyId>xxxxxxxxxxxxxxxxxxxx</accessKeyId>
    <!-- 华为云账号的SK, 认证用的SK硬编码到代码中或者明文存储都有很大的安全风险, 建议密文存放, 使用
    时解密, 确保安全 -->
    <accessKeySecret>xxxxxxxxxxxxxxxxxxxx</accessKeySecret>
    <!-- 云日志服务的区域 -->
    <regionName>cn-nxxxx</regionName>
    <encoder>
      <pattern>%d %-5level [%thread] [%customerParam] %logger{0}: %msg</pattern>
    </encoder>
    <!-- 单个Appender能缓存的日志大小上限 -->
    <totalSizeInBytes>104857600</totalSizeInBytes>
```

```
<!-- producer发送日志时阻塞时间 -->
<maxBlockMs>0</maxBlockMs>
<!-- 执行日志发送任务的线程池大小 -->
<ioThreadCount>8</ioThreadCount>
<!-- producer发送单批日志量上限 -->
<batchSizeThresholdInBytes>524288</batchSizeThresholdInBytes>
<!-- producer发送单批日志条数上限 -->
<batchCountThreshold>4096</batchCountThreshold>
<!-- producer发送单批日志等待时间 -->
<lingerMs>2000</lingerMs>
<!-- producer发送日志失败后重试次数 -->
<retries>5</retries>
<!-- 首次重试的退避时间 -->
<baseRetryBackoffMs>500</baseRetryBackoffMs>
<!-- 重试的最大退避时间 -->
<maxRetryBackoffMs>500</maxRetryBackoffMs>
<!-- 默认false, true: 可以跨云上报日志, false: 仅能在华为云ecs主机上报日志 -->
<enableLocalTest>false</enableLocalTest>
<!-- 超过1M的日志, 拆分后放弃1M以外的 -->
<!-- <giveUpExtraLongSingleLog>true</giveUpExtraLongSingleLog>-->
<!-- 是否使用日志组/日志流名称上报日志-->
<!-- <logGroupStreamByName>false</logGroupStreamByName>-->
</appender>
<logger name="com.huawei.test.logback.info" level="INFO">
  <appender-ref ref="LogInfoTest"/>
</logger>
</configuration>
```

2. 编写代码中产生日志，示例如下：

```
package com.huawei.test.logback.info;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class TestLogBackAppenderInfo {

    private static final Logger LOGGER = LoggerFactory.getLogger(TestLogBackAppenderInfo.class);

    public void run() throws InterruptedException {
        LOGGER.info("test-logback~~~info");
    }

}
```

producer 性能基线

上报日志时，请参考如下参数的测试性能基线，若超出基线值，可能会导致日志上报异常。

ECS虚拟机配置：

- 实例规格：通用计算增强型 c7.xlarge.2
- CPU：4 vCPU
- 内存：8 GB
- 基准带宽：100 Mbit/s
- OS：Huawei Cloud EulerOS release 2.0
- JVM：OpenJDK 64-Bit Server VM (build 17.0.7+7, mixed mode)

测试程序说明（单个producer）：

- totalSizeInBytes: 104857600
- maxBlockMs: 0

- batchSizeThresholdInBytes: 1048576
- batchCountThreshold: 40960
- lingerMs: 2000
- ioThreadCount: 具体用例中调整
- JVM 初始堆/最大堆大小: 1GB
- 发送日志总条数: 100,000,000
- 发送日志总大小: 约50GB

按照参数基线值设置后, 使用华为云ECS机器作为日志上报环境, 通过华为云内网服务入口进行上报。

SDK上报性能基线测试结果参考:

- 上报日志: 测试中使用的单条日志大小约为550字节。为了模拟数据的随机性, 测试使用的数据为随机字符串。
- 性能基线如下:

表 4-84 性能基线

IO 线程数量	数据吞吐量	数据吞吐速率	CPU 使用率
2	8.4MB/S	1.7W条/s	15%
4	15.3MB/S	3W条/s	29%
6	21.5MB/S	4.2W条/s	36%

总结如下:

- CPU时间主要花费在对象的序列化和压缩上, 在吞吐量较高的情况下CPU使用率比较高。但在日常环境中, 单机数据流量均值为100KB/S, 因此造成的CPU消耗几乎可以忽略不计。
- 增加IO线程数量可以显著提高吞吐量, 尤其是当IO线程数量少于可用处理器个数时。
- 调整totalSizeInBytes对吞吐量影响不够显著, 增加totalSizeInBytes会造成更多的CPU消耗, 建议使用默认值。
- 当日志上报量超过单个producer时:
 - a. 建议拆分日志流, 使用多个producer上报日志, 分摊流量, 以保障SDK处于正常上报状态。
 - b. 如果maxBlockMs为0时, SDK处于非阻塞状态, 会触发保护机制自动降级, 可能会对部分日志做丢弃处理。
 - c. 如果maxBlockMs大于0时, SDK处于阻塞状态, 阻塞时间为maxBlockMs, 可能会造成SDK发送日志处于阻塞状态。

配置参数说明

- Appender配置参数说明。

表 4-85 Appender 配置参数说明表

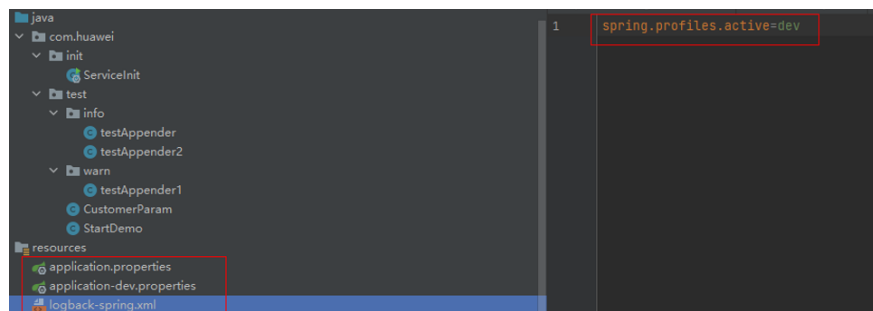
参数名称	描述	类型	是否需要填写	默认值
projectId	华为云账号的项目ID（project id）。	String	必填	-
accessKeyId	华为云账号的AK。认证用的AK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
accessKeySecret	华为云账号的SK。认证用的SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
regionName	云日志服务的区域。	String	必填	-
logGroupId	LTS的日志组ID。	String	必填	-
logStreamId	LTS的日志流ID。	String	必填	-
appName	日志标签appName字段。	String	选填	-
pathFile	日志标签pathFile字段。	String	选填	-
totalSizeInBytes	单个producer实例能缓存的日志大小上限。	int	选填	104857600（即100MB）
maxBlockMs	如果producer可用空间不足，调用者在send方法上的最大阻塞时间，单位为毫秒。默认60000毫秒，建议0毫秒。 - 当maxBlockMs值≥ 0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 - 当maxBlockMs值$= -1$时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60000毫秒

参数名称	描述	类型	是否需要填写	默认值
ioThreadCount	执行日志发送任务的线程池大小。	int	选填	客户机器空闲CPU数量，但一定大于1
batchSizeThresholdInBytes	当一个ProducerBatch中缓存的日志大小大于等于batchSizeThresholdInBytes时，该ProducerBatch缓存的日志将被发送到LTS。	int	选填	524288（即0.5MB）
batchCountThreshold	当一个ProducerBatch中缓存的日志条数大于等于batchCountThreshold时，该ProducerBatch缓存的日志将被发送到LTS。	int	选填	4096
lingerMs	一个ProducerBatch从创建到可发送的逗留时间。	int	选填	2s
retries	如果某个ProducerBatch首次发送失败，能够对其重试的次数，建议为5次。如果retries小于等于0，该ProducerBatch首次发送失败后将直接进入失败队列。	int	选填	10
baseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1s
maxRetryBackoffMs	重试的最大退避时间。	long	选填	50s
giveUpExtraLongSingleLog	超过1M的日志，拆分后丢弃大于1M的数据。	boolean	选填	false
enableLocalTest	是否开启跨云上报日志。 （此功能仅适用于开发调测，对于上报性能不做保障） - 选true时，开启跨云后用户能通过公网（仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一）访问。 - 选false时，关闭跨云后用户是通过华为云当前区域主机访问。	boolean	选填	false

参数名称	描述	类型	是否需要填写	默认值
logGroupStreamByName	是否使用日志组/日志流名称上报日志（SDK版本最低必须为1.1.0支持此功能）。如果为true，表明使用名称上报，则logGroupId、logStreamId参数值分别填入日志组、日志流名称即可。	boolean	选填	false
proxyIp	使用自定义ip端口上报日志。	String	选填	-

- 如果是SpringBoot项目，则可以将logback.xml替换为logback-spring.xml，且支持读取springProfile中的配置，可以自由切换开发、测试等环境的日志采集环境。使用说明如下：

- 在SpringBoot项目中的application.properties可以自由指定使用的配置项。



- logback-spring.xml中配置

```
<springProfile name="dev">
  <root level="INFO">
    <appender-ref ref="customerAppender"/>
  </root>
</springProfile>
```

- 如果需要logback.xml、logback-spring.xml支持自定义标签，且将标签中的value打印到日志中。执行以下步骤：

- 需要继承ClassicConverter类，重写convert方法，返回值为自定义标签的value。

```
public class CustomerParam extends ClassicConverter {
    @Override
    public String convert(ILoggingEvent iLoggingEvent) {
        return "自定义标签value";
    }
}
```

- 在logback.xml或者logback-spring.xml中引用ClassicConverter类，使用%自定义标签的名称，然后就可以使用该标签了。

```
<conversionRule conversionWord="customerParam"
converterClass="com.huawei.test.logback.CustomerParam"/>

<appender name="customer" class="com.huawei.appender.LogbackSDKAppender">
  <encoder>
    <pattern>%d %-5level [%thread] [%customerParam] %logger{0}: %msg</pattern>
  </encoder>
</appender>
```

参数获取方法

- 区域表

表 4-86 区域表

区域名称	区域
华北-北京二	cn-north-2
华北-北京四	cn-north-4
华北-北京一	cn-north-1
华东-上海二	cn-east-2
华东-上海一	cn-east-3
华南-广州	cn-south-1
华南-深圳	cn-south-2
西南-贵阳一	cn-southwest-2
亚太-新加坡	ap-southeast-3

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.4 云日志服务 Go SDK

云日志服务SDK提供了Go语言上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请用户根据所在区域，选择RegionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（project id），详细步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

使用说明

- 当用户修改权限后，权限信息在一天后生效。
- 通过Go SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

- 当前仅华北-北京四、华东-上海一、华南-广州区域使用Go SDK时支持跨云/本地上报日志，Endpoint端口使用443。

注意事项

认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

安装 Go SDK

步骤1 获取LTS Go SDK包。

```
go get github.com/huaweicloud/huaweicloud-lts-sdk-go
```

步骤2 导入日志LTS Go SDK包。

```
import github.com/huaweicloud/huaweicloud-lts-sdk-go
```

步骤3 编写上报日志代码。

可以自定义发送成功与失败时的回调方法，实现方式请参考如下命令：自定义ErrorHandler的结构体，实现Success和Fail方法，并且在发送日志时直接使用。

```
SendLogWithCallBack(groupId, streamId string, log *Log, callback CallBack)
```

示例代码：

```
package main

import (
    "github.com/huaweicloud/huaweicloud-lts-sdk-go"
    "fmt"
    "sync"
    "time"
)

var (
    ak      = ""
    sk      = ""
    pid     = ""
    groupId = ""
    streamId1 = ""
    streamId2 = ""
    region  = ""
    endpoint = ""
)

func main() {
    // 获取默认配置
    producerConfig := producer.GetConfig()
    // 上报地址
    producerConfig.Endpoint = endpoint
    // accessKeyId：华为云账号的AK，认证用的AK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全
    producerConfig.AccessKeyId = os.Getenv("accessKeyId")
    // accessKeySecret：华为云账号的SK，认证用的SK硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全
    producerConfig.AccessKeySecret = os.Getenv("accessKeySecret")
    // 区域名称
    producerConfig.RegionId = region
    // 账户projectid
    producerConfig.ProjectId = pid
    // 初始化producer实例
    producerInstance := producer.InitProducer(producerConfig)
    // 启动producer
    producerInstance.Start()
    wg := sync.WaitGroup{}
```

```
for i := 0; i < 10; i++ {
    wg.Add(1)
    go func() {
        for j := 0; j < 1000; j++ {
            labels := make(map[string]string)
            labels["keyA"] = "valueA"
            labels["keyB"] = "valueB"
            labels["keyC"] = "valueC"

            logContent := fmt.Sprintf("content for this test %d", j)
            // 生成日志 填入日志内容及labels
            log := producer.GenerateLog([]string{logContent}, labels)
            // 发送日志到日志组，日志流
            err := producerInstance.SendLog(groupId, streamId1, log)
            err = producerInstance.SendLog(groupId, streamId2, log)
            if err != nil {
                fmt.Println(err)
            }
        }

        handle := ErrorHandler{}
        // 带CallBack的发送日志方法，发送失败时会调用ErrorHandler的Fail方法
        err = producerInstance.SendLogWithCallBack("groupId", "streamId1", log, handle)
        err = producerInstance.SendLogWithCallBack("groupId", "streamId1", log, handle)
        if err != nil {
            fmt.Println(err)
        }
        // 结构化日志上报方法
        var sLog producer.StructLog
        logContent := make(map[string]string)
        logContent["keyA"] = "valueA"
        logContent["keyB"] = "valueB"
        logContent1 := make(map[string]string)
        logContent1["keyA1"] = "valueA1"
        logContent1["keyB1"] = "valueB1"
        sLog.Contents = append(sLog.Contents, logContent)
        sLog.Contents = append(sLog.Contents, logContent1)
        sLog.Time = time.Now().UnixNano() / 1000 / 1000
        // 发送结构化日志方法
        err = producerInstance.SendLogStruct("groupId2", "streamId2", &sLog)
        if nil != err {
            continue
        }
        // 带CallBack的发送结构化日志方法，发送失败时会调用ErrorHandler的Fail方法
        var callback ErrorHandler
        err = producerInstance.SendLogStructWithCallBack("groupId2", "streamId2", &sLog, callback)
        if nil != err {
            continue
        }
        time.Sleep(1 * time.Microsecond)
    }

    wg.Done()
    fmt.Printf("test func finished\n")
}()

wg.Wait()
fmt.Printf("send all complete ...")
// 关闭发送实例
producerInstance.Close(60 * 1000)
time.Sleep(10 * 60 * time.Second)
}

// errorHandler，可以针对发送成功和失败进行相关的callback调用，方便失败时打印错误日志，进行问题定位
type ErrorHandler struct{}

// 发送成功时的callback方法
func (ErrorHandler) Success(result *producer.Result) {
    fmt.Printf("send log to lts success, success flag: %v\n", result.IsSuccessful())
}
```

```
}

// 发送失败时的callback方法，可以打印result中的错误信息，方便问题定位
func (ErrorHandler) Fail(result *producer.Result) {
    fmt.Printf("send log to lts error, success flag: %v requestId: %s, httpcode: %d, errorCode: %s, errorMsg: %s\n",
        result.IsSuccessful(), result.GetRequestId(), result.GetHttpCode(), result.GetErrorCode(),
        result.GetErrorMessage())
}
```

----结束

配置参数说明

- producer config参数说明。

表 4-87 producer config 参数说明

参数名称	描述	类型	是否需要填写	默认值
ProjectId	华为云账号的项目ID（project id）。	String	必填	-
AccessKeyId	华为云账号的AK。	String	必填	-
AccessKeySecret	华为云账号的SK。	String	必填	-
RegionName	云日志服务的区域。	String	必填	-
Endpoint	上报的服务器域名或IP。	String	必填	-
TotalSizeInBytes	单个producer实例能缓存的日志大小上限。	int	选填	104857600（即100MB）
MaxBlockSec	如果producer可用空间不足，调用者在send方法上的最大阻塞时间，默认为60秒。建议为0秒。 • 当maxBlockMs值>=0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 • 当maxBlockMs值=-1时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60s
MaxIoWorkers	执行日志发送任务的任务池大小。	int	选填	默认10个

参数名称	描述	类型	是否需要填写	默认值
MaxBatchSize	当一个ProducerBatch中缓存的日志大小大于等于batchSizeThresholdInBytes时，该batch将被发送。	int	选填	524288（即0.5MB）
MaxBatchCount	当一个ProducerBatch中缓存的日志条数大于等于batchCountThreshold时，该batch将被发送。	int	选填	4096
LingerMs	一个 ProducerBatch 从创建到可发送的逗留时间。	int	选填	2s
Retries	如果某个ProducerBatch首次发送失败，能够对其重试的次数，建议为3次。如果retries小于等于0，该ProducerBatch首次发送失败后将直接进入失败队列。	int	选填	10
BaseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1s
MaxRetryBackoffMs	重试的最大退避时间。	long	选填	50s

- 日志生成GenerateLog方法类参数说明。
单次上报条数小于4096条，且小于512K。

表 4-88 GenerateLog 方法类参数说明

参数名称	描述	类型	是否需要填写
contents	批量日志内容。	[]string	是
labels	日志标签，map格式。	map[string]string	是

- 日志上报SendLog方法类参数说明。

表 4-89 SendLog 方法类参数说明

参数名称	描述	类型	是否需要填写
groupId	日志组id	long	是
streamId	日志流id	String	是
log	日志结构	struct log	是

- 结构化日志上报StructLog方法类参数说明。

表 4-90 StructLog 方法类参数说明

参数名称	描述	类型	是否需要填写
Time	日志上报时间	int64	否
Contents	日志上报内容	[]map[string]string	是

- 结构化日志上报SendLogStruct方法类参数说明。

表 4-91 SendLogStruct 方法类参数说明

参数名称	描述	类型	是否需要填写
groupId	日志组id	String	是
streamId	日志流id	String	是
log	日志结构	Struct Logstruct	是

参数获取方式

- 上线区域表，使用华为云主机时参考以下信息：

表 4-92 区域表

区域名称	区域	终端节点
华北-北京四	cn-north-4	https://lts-access.cn-north-4.myhuaweicloud.com:8102
华东-上海一	cn-east-3	https://lts-access.cn-east-3.myhuaweicloud.com:8102
华南-广州	cn-south-1	https://lts-access.cn-south-1.myhuaweicloud.com:8102
亚太-新加坡	ap-southeast-3	https://lts-access.ap-southeast-3.myhuaweicloud.com:8102

当前仅华北-北京四、华东-上海一、华南-广州区域使用SDK时支持跨云/本地上报日志，Endpoint端口使用443。

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.5 云日志服务 C++ SDK

云日志服务SDK提供了C++语言上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请用户根据所在区域，选择RegionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（project id），详细步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

使用说明

当用户修改权限后，权限信息在一天后生效。通过SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

注意事项

认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

安装 C++ SDK

步骤1 获取LTS C++ SDK代码

```
git clone https://gitee.com/lordstar-habile/huaweicloud-lts-cpp-sdk.git
```

步骤2 引用日志LTS C++头文件。

```
#include <sdk/Config.h>
#include <sdk/Producer.h>
#include <client/logger.h>
```

步骤3 编写上报日志代码。

可以自定义发送成功与失败时的回调方法，实现方式请参考如下命令：创建CallBack对象，实现Success和Fail方法，并且在发送日志时直接使用。

```
typedef void (*callback)(Result*);
class CallBack {
public:
    callback Success;
    callback Fail;
};
int Producer::SendLog(std::string groupId, std::string streamId, std::vector<Log*> log, CallBack* cb);
```

示例如下：

```
#include <sdk/Config.h>
#include <sdk/Producer.h>
#include <client/logger.h>

Config* GetConfig() {
```

```
Config* cfg = new Config();

cfg->Endpoint = "endpoint";
cfg->AccessKeyID = "ak";
cfg->AccessKeySecret = "sk";
cfg->RegionId = "region";
cfg->ProjectId = "pid";

cfg->validateProducerConfig();

return cfg;
}

Log* generateLog() {
    std::vector<std::string> logContentArr = {"content for this test no.1"};
    std::map<std::string, std::string> labels = {"keyA", "valueA"};

    return GenerateLog(logContentArr, labels);
}

void userCallBack(Result* result) {
    if (result == nullptr) {
        return;
    }

    const char* isSuccess = "false";
    if (result->successful) {
        isSuccess = "true";
    }

    LogPrint(LOG_LEVEL_WARN, "userCallBack [%s]", isSuccess);
}

int main() {
    Config *cfg = GetConfig();

    Producer *producerInstance = InitProducer(cfg);
    producerInstance->Start();

    CallBack *cb = new CallBack();
    cb->Success = userCallBack;
    cb->Fail = userCallBack;

    Log *log = generateLog();
    producerInstance->SendLog("groupId", "streamId", std::vector<Log *>{log}, cb);

    SleepMs(60 * 60 * 1000);

    producerInstance->Close();

    delete cb;

    delete cfg;
}
```

----结束

配置参数说明

- producer config参数说明。

参数名称	描述	类型	是否需要填写	默认值
ProjectId	华为云账号的项目ID（project id）。	String	必填	-

参数名称	描述	类型	是否需要填写	默认值
AccessKeyId	华为云账号的AK。	String	必填	-
AccessKeySecret	华为云账号的SK。	String	必填	-
RegionName	云日志服务的区域。	String	必填	-
Endpoint	上报目的。	String	必填	-
TotalSizeInBytes	单个producer实例能缓存的日志大小上限。	int	选填	104857600 (即100MB)
MaxBlockSec	如果producer可用空间不足，调用者在send方法上的最大阻塞时间，默认为6秒。建议为0秒。 - 当maxBlockMs值≥ 0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 - 当maxBlockMs值$= -1$时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60s
MaxIoWorkers	执行日志发送任务的任务池大小。	int	选填	默认10个
MaxBatchSize	当一个 ProducerBatch 中缓存的日志大小大于等于 batchSizeThresholdInBytes 时，该 batch 将被发送。	int	选填	524288 (即 0.5MB)
MaxBatchCount	当一个 ProducerBatch 中缓存的日志条数大于等于 batchSizeThreshold 时，该 batch 将被发送。	int	选填	4096
LingerMs	一个 ProducerBatch 从创建到可发送的逗留时间。	int	选填	2s
Retries	如果某个 ProducerBatch 首次发送失败，能够对其重试的次数，建议为 3 次。如果 retries 小于等于 0，该 ProducerBatch 首次发送失败后将直接进入失败队列。	int	选填	10

参数名称	描述	类型	是否需要填写	默认值
BaseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1s
MaxRetryBackoffMs	重试的最大退避时间。	long	选填	50s

- 日志生成GenerateLog方法类参数说明。单次上报条数小于4096条，且小于512K。

参数名称	描述	类型	是否需要填写
contents	批量日志内容。	[]string	是
labels	日志标签，map格式。	map[string]string	是

- 日志上报SendLog方法类参数说明。

表 4-93 SendLog 方法类参数说明

参数名称	描述	类型	是否需要填写
groupId	日志组id	long	是
streamId	日志流id	String	是
log	日志结构	struct log	是

参数获取方式

- 上线区域表，使用华为云主机时参考以下信息：

表 4-94 区域表

区域名称	区域	终端节点
华北-北京四	cn-north-4	https://lts-access.cn-north-4.myhuaweicloud.com:8102
华东-上海一	cn-east-3	https://lts-access.cn-east-3.myhuaweicloud.com:8102
华南-广州	cn-south-1	https://lts-access.cn-south-1.myhuaweicloud.com:8102
亚太-新加坡	ap-southeast-3	https://lts-access.ap-southeast-3.myhuaweicloud.com:8102

当前仅华北-北京四、华东-上海一、华南-广州区域使用SDK时支持跨云/本地上报日志，Endpoint端口使用443。

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.6 云日志服务 Python SDK

云日志服务SDK提供了Python语言上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请用户根据所在区域，选择RegionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（project id），详细步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。
- Python SDK适用于python 3.10.1及以上版本。

使用说明

当用户修改权限后，权限信息在一天后生效。通过SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

注意事项

认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

安装 Python SDK

步骤1 获取LTS Python SDK包源码。

```
git clone https://gitee.com/lordstar-habile/huaweicloud-lts-python-sdk.git
```

步骤2 安装相关依赖。

```
pip3 install requests  
pip3 install loguru  
pip3 install six
```

步骤3 编写上报日志代码。

可以自定义发送成功与失败时的回调方法，实现方式请参考如下命令：创建CallBack对象，实现invoke_on_success和invoke_on_failed方法，并且在发送日志时直接使用。

```
class CallBack(object):  
    def invoke_on_success(self, result):
```

```
pass

def invoke_on_failed(self, result):
    pass
```

示例如下：

```
import random
import time
from concurrent.futures import ThreadPoolExecutor
from loguru import logger
from producer.core.producer import Producer
from producer.callback.call_back import CallBack
from producer.model import lts_store
from producer.model.config import Config
from producer.utils import common
def generate_random_str(str_len=16):
    """
    生成一个指定长度的随机字符串
    """
    random_str = ""
    base_str = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789'
    length = len(base_str) - 1
    for i in range(str_len):
        random_str += base_str[random.randint(0, length)]
    return random_str
def send_log_single_producer():
    thread_num = 100
    threads = []
    begin_ms = common.current_time_ms()
    submit_send_log_thread_pool = ThreadPoolExecutor(max_workers=thread_num,
thread_name_prefix="send_log_thread")
    config = Config()
    config.endpoint = ""
    config.access_key = ""
    config.access_secret = ""
    config.region_id = ""
    config.project_id = ""
    producer = Producer.init_producer(config)
    producer.start_producer()
    group_id = ""
    stream_id = ""

    log_content = generate_random_str(1024)
    log = [log_content]
    labels = {"keyA": "valueA"}
    log_p = lts_store.generate_log(log, labels)
    for i in range(thread_num):
        new_thread = submit_send_log_thread_pool.submit(send_log_function, producer, group_id, stream_id,
log_p)
        threads.append(new_thread)
    for t in threads:
        t.result()
    end_ms = common.current_time_ms()
    logger.info("end send log, cost [{}]ms", end_ms - begin_ms)
def send_log_function(one_producer, group_id, stream_id, log_p):
    for i in range(2000):
        one_producer.send_log_with_callback(group_id, stream_id, log_p)
class MyCallBack(CallBack):
    def invoke_on_success(self, result):
        logger.info("send log success")
    def invoke_on_failed(self, result):
        logger.error("send log success")
if __name__ == '__main__':
    send_log_single_producer()
    time.sleep(1000)
```

----结束

配置参数说明

- producer config参数说明。

参数名称	描述	类型	是否需要填写	默认值
projectId	华为云账号的项目ID（project id）。	String	必填	-
ak	华为云账号的AK。	String	必填	-
sk	华为云账号的SK。	String	必填	-
region	云日志服务的区域。	String	必填	-
endpoint	上报目的。	String	必填	-
logGroup	LTS的日志组ID。	string	必填	-
logStream	LTS的日志流ID。	string	必填	-

- 日志生成GenerateLog方法类参数说明。单次上报条数小于4096条，且小于512K。

参数名称	描述	类型	是否需要填写
contents	批量日志内容。	[]string	是
labels	日志标签，map格式。	map[string]string	是

- 日志上报SendLog方法类参数说明。

表 4-95 SendLog 方法类参数说明

参数名称	描述	类型	是否需要填写
groupId	日志组id	string	是
streamId	日志流id	string	是
log	日志结构	class Log	是

参数获取方式

- 上线区域表，使用华为云主机时参考以下信息：

表 4-96 区域表

区域名称	区域	终端节点
华北-北京四	cn-north-4	https://lts-access.cn-north-4.myhuaweicloud.com:8102
华东-上海一	cn-east-3	https://lts-access.cn-east-3.myhuaweicloud.com:8102
华南-广州	cn-south-1	https://lts-access.cn-south-1.myhuaweicloud.com:8102
亚太-新加坡	ap-southeast-3	https://lts-access.ap-southeast-3.myhuaweicloud.com:8102

当前仅华北-北京四、华东-上海一、华南-广州区域使用SDK时支持跨云/本地上报日志，Endpoint端口使用443。

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.7 云日志服务 Net SDK

云日志服务SDK提供了Net语言上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请用户根据所在区域，选择RegionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（ project id ），详细步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

使用说明

当用户修改权限后，权限信息在一天后生效。通过SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

当前仅华北-北京四、华东-上海一、华南-广州区域使用SDK时支持跨云/本地上报日志，Endpoint端口使用443。

注意事项

认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

操作步骤

步骤1 获取LTS Net SDK包源码。

```
git clone https://gitee.com/lordstar-habile/huaweicloud-lts-dotnet-sdk.git
```

步骤2 安装相关依赖。

```
dotnet add package Microsoft.Extensions.Http.Polly --version 8.0.2
dotnet add package RestSharp --version 110.2.0
dotnet add package Newtonsoft.Json --version 13.0.3
```

步骤3 编写上报日志代码。

可以自定义发送成功与失败时的回调方法，实现CallBack的abstract class，实现Success和Fail方法，并且在发送日志时直接使用。

```
public Error SendLogWithCallBack(string groupId, string streamId, Log log, CallBack callBack)
```

示例如下：

```
using HuaweiCloud.SDK.LTS.Model;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;

namespace HuaweiCloud.SDK.LTS.Sample
{
    public class Sample
    {
        public static void SendLogToLts(string endpoint, string accessKey, string accessSecret, string region,
            string projectId, string groupId, string streamId,
            int threadCount, int logItemNum, int HttpTimeout, int LingerMs, string logContent, long maxIO)
        {
            Console.WriteLine("start time is: {0}", DateTimeOffset.Now.ToUnixTimeMilliseconds());
            var producerConfig = Config.GetConfig();
            producerConfig.LingerMs = LingerMs;
            producerConfig.Endpoint = endpoint;
            producerConfig.AccessKeyID = accessKey;
            producerConfig.AccessKeySecret = accessSecret;
            producerConfig.RegionId = region;
            producerConfig.ProjectId = projectId;
            producerConfig.HttpMaxTimeout = HttpTimeout;
            producerConfig.MaxIoWorkers = maxIO;
            var producerInstance = new Producer(producerConfig);

            producerInstance.Start();
            var wg = new WaitGroup();
            for (int i = 0; i < threadCount; i++)
            {
                wg.Add(1);
                Task.Run(() => Handler.handle(producerInstance, wg, groupId, streamId, logItemNum,
                    logContent));
            }

            wg.wait();
            Console.WriteLine("send all complete ...");
            Thread.Sleep(TimeSpan.FromSeconds(3000));
        }
    }

    public class WaitGroup
```

```
{
    private long _wg = 0;

    public long Wg
    {
        get => _wg;
        set => _wg = value;
    }

    public void Add(int delta)
    {
        Interlocked.Add(ref _wg, delta);
    }

    public void Done()
    {
        Interlocked.Decrement(ref _wg);
    }

    public void wait()
    {
        while (Interlocked.Read(ref _wg) != 0)
        {
            Thread.Sleep(200);
        }
    }
}

public class Handler
{
    public static void handle(Producer producer, WaitGroup wg, string groupId, string streamId, int
logItemNums, string logContent)
    {
        for (int j = 0; j < logItemNums; j++)
        {
            var labels = new Dictionary<string, string>();
            labels["keyA"] = "valueA";
            labels["keyB"] = "valueB";
            labels["keyC"] = "valueC";
            var log = LTS_Store.GenerateLog(new List<string> { logContent }, labels);
            var err = producer.SendLog(groupId, streamId, log);
            if (err != null)
            {
                Console.WriteLine(JsonConvert.SerializeObject(err));
            }

            Thread.Sleep(TimeSpan.FromMilliseconds(0.1));
        }

        wg.Done();
    }
}
}
```

----结束

配置参数说明

- producer config参数说明。

参数名称	描述	类型	是否需要填写	默认值
ProjectId	华为云账号的项目ID（project id）。	String	必填	-

参数名称	描述	类型	是否需要填写	默认值
AccessKeyId	华为云账号的AK。	String	必填	-
AccessKeySecret	华为云账号的SK。	String	必填	-
RegionName	云日志服务的区域。	String	必填	-
Endpoint	上报的服务器域名或IP。	String	必填	-
MaxBlockSec	如果producer可用空间不足，调用者在send方法上的最大阻塞时间，默认为60秒。建议为0秒。 当maxBlockMs值>=0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 当maxBlockMs值=-1时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60s
MaxIoWorkers	执行日志发送任务的任务池大小。	int	选填	默认10个
MaxBatchSize	当一个 ProducerBatch 中缓存的日志大小大于等于 batchSizeThresholdInBytes 时，该 batch 将被发送。	int	选填	524288 (即 0.5MB)
MaxBatchCount	当一个 ProducerBatch 中缓存的日志条数大于等于 batchSizeThreshold 时，该 batch 将被发送。	int	选填	4096
LingerMs	一个 ProducerBatch 从创建到可发送的逗留时间。	int	选填	2s
Retries	如果某个 ProducerBatch 首次发送失败，能够对其重试的次数，建议为 3 次。如果 retries 小于等于 0，该 ProducerBatch 首次发送失败后将直接进入失败队列。	int	选填	10
BaseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1s

参数名称	描述	类型	是否需要填写	默认值
MaxRetryBackoffMs	重试的最大退避时间。	long	选填	50s
HttpMaxTimeout	HTTP请求的超时时间	int	选填	60s

- 日志生成GenerateLog方法类参数说明。
单次上报条数小于4096条，且小于512K。

表 4-97 GenerateLog 方法类参数说明

参数名称	描述	类型	是否需要填写
contents	批量日志内容。	List<string>	是
labels	日志标签，map格式。	Dictionary<string, string>	是

- 日志上报SendLog方法类参数说明

表 4-98 SendLog 方法类参数说明

参数名称	描述	类型	是否需要填写
groupId	日志组id	string	是
streamId	日志流id	string	是
log	日志结构	class Log	是

参数获取方式

- 上线区域表，使用华为云主机时参考以下信息：

区域名称	区域	终端节点
华北-北京四	cn-north-4	https://lts-access.cn-north-4.myhuaweicloud.com:8102
华东-上海一	cn-east-3	https://lts-access.cn-east-3.myhuaweicloud.com:8102
华南-广州	cn-south-1	https://lts-access.cn-south-1.myhuaweicloud.com:8102
亚太-新加坡	ap-southeast-3	https://lts-access.ap-southeast-3.myhuaweicloud.com:8102

当前仅华北-北京四、华东-上海一、华南-广州区域使用SDK时支持跨云/本地上报日志，Endpoint端口使用443。

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.8 云日志服务鸿蒙 SDK

云日志服务鸿蒙SDK提供了ArkTS语言上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

说明

当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请[提交工单](#)申请。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请您根据所在区域，选择RegionName。
- 获取华为云账号的项目ID（project id），步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

版本更新说明

SDK如何处理个人信息请参考[华为云日志服务移动端日志采集SDK隐私声明](#)。

您集成和使用的SDK时需要遵从个人信息保护基本要求，详情请参考[华为云日志服务移动端日志采集SDK开发者合规指南](#)。

表 4-99 版本更新说明

版本号	下载地址	更新说明	系统
0.0.3	单击 下载	将SDK写入和上报日志逻辑迁移到子线程，添加代码混淆与单元测试用例。	HarmonyOS NEXT及以上。
0.0.2	单击 下载	将日志状态存入数据库。	HarmonyOS NEXT及以上。
0.0.1	单击 下载	LTS鸿蒙SDK上线，支持缓存上报、立即上报等功能。	HarmonyOS NEXT及以上。

安装鸿蒙 SDK

步骤1 集成接入SDK。 SDK包使用语言为ArkTS，只能在ArkTS文件中引入。

1. 鸿蒙仓库集成。

- 在终端中运行安装命令。

```
ohpm install lts-harmony-sdk
```

- 在app/oh-package.json5中添加依赖。

```
dependencies {  
  ...  
  "lts-harmony-sdk": "0.0.3"  
  ...  
}
```

- 在终端中运行以下命令使能SDK。

```
ohpm install
```

2. 手动集成。

- 下载日志SDK包。
- 下载后解压到指定目录。注：直接解压即可，不需要额外操作。
- 将解压后的har静态库文件添加到您的项目工程中。

- 在app/oh-package.json5中添加依赖。

```
dependencies {  
  ...  
  "lts-harmony-sdk": "file:../plugin/LTS_SDK/default/lts-harmony-sdk.har" // HAR包存放地址  
  ...  
}
```

- 在终端中运行以下命令使能SDK。

```
ohpm install
```

步骤2 初始化，详细参数请参考表4-102。

LTSSDK支持多实例日志上报机制，根据不同配置创建实例进行上报。

- 添加依赖

```
import { LTSSDK, ConfigParam } from 'lts-harmony-sdk';
```

- ArkTS：初始化代码示例。

```
// LTS参数配置  
const config:ConfigParam = {  
  // 必填参数  
  region: string, // 上报region  
  projectId: string, // 华为云项目ID  
  groupId: string, // LTS日志组ID  
  streamId: string, // LTS日志流ID  
  accessKey: string, // 华为云访问密钥  
  secretKey: string, // 华为云秘密访问密钥  
  // 选填参数  
  url: string, // 上报地址  
  cacheThreshold: number, // 上报条数阈值  
  timeInterval: number, // 上报时间阈值  
  isReportBackground: boolean // 是否开启后台上报  
};  
// 获取ApplicationContext  
let applicationContext = this.context.getApplicationContext();  
// LTS初始化方法  
const ltssdk = new LTSSDK(applicationContext, config)
```

步骤3 上报日志，LTS提供两种上报日志的方法。

表 4-100 两种上报日志的方法

Method	Description
report(content, labels)	上报日志：先存入本地数据库，根据配置中设定的策略实时上报。
reportImmediately(content, labels)	立即上报日志。

表 4-101 Parameters 参数

Name	Description
content	日志内容，支持字典和字典数组；键值对最多300个；content转JSON字符串最大支持长度为30*1024，超出部分被截断。
labels	日志标签。支持字典或空值；最外层键值对最多50个；最外层key最大长度为64，支持字母、数字和下划线组合，首字符须是字母；字典转JSON字符串最大支持长度为30720，超出则该条日志无法上报。

ArkTS代码示例如下：

```
const ltssdk = new LTSSDK(applicationContext, config)
let fruit = {
  "fruit_1": "apple",
  "fruit_2": "pear",
  "fruit_3": "banana"
};
let food = { "food_1": "rice" };
let labels = { "date": "2023-10-01" };
let contents = [food, fruit];
ltssdk.report(food); // 缓存上报单条 不带标签
ltssdk.report(food, labels); // 缓存上报单条 带标签
ltssdk.reportImmediately(food); // 立即上报单条 不带标签
ltssdk.reportImmediately(food, labels); // 立即上报单条 带标签
ltssdk.report(contents); // 缓存上报多条 不带标签
ltssdk.report(contents, labels); // 缓存上报多条 带标签
ltssdk.reportImmediately(contents); // 立即上报多条 不带标签
ltssdk.reportImmediately(contents, labels); // 立即上报多条 带标签
```

----结束

配置参数说明

表 4-102 初始化参数说明

参数名称	类型	是否必填	默认值	描述
projectId	string	必填	-	华为云账号的项目ID。
groupId	string	必填	-	LTS的日志组ID。

参数名称	类型	是否必填	默认值	描述
streamId	string	必填	-	LTS的日志流ID。
accessKey	string	必填	-	华为云账号的访问密钥，简称AK。注意：认证用的华为云账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
secretKey	string	必填	-	华为云账号的秘密访问密钥，简称SK。注意：认证用的华为云账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
region	string	必填	-	LTS的服务区域。
url	string	选填	null	用于上报的公网地址域名，如未设置，将根据region自动生成链接，格式如下：https://{region}.myhuaweicloud.com
cacheThreshold	number	选填	200条	当本地数据库日志存储条数达到该值会触发上报，取值范围为30-1000条。
timeInterval	number	选填	3秒	定时器每隔该值会触发上报，取值范围为1-1800秒。
isReportBackground	boolean	选填	true	是否开启APP切入后台时上报功能。

上报日志策略说明

LTSSDK支持三种日志上报策略：阈值上报、定时上报、应用切后台时上报，根据场景选择合适的值配置cacheThreshold、timeInterval、isReportBackground等参数。

权限配置

LTS-SDK 需要鸿蒙的网络权限，如果没有配置，请将下面内容的写入APP的module.json5 文件中：

```
{
  "module": {
    ...
    "requestPermissions": [
      ...
      {
        "name": "ohos.permission.INTERNET"
      },
    ]
  }
}
```

```
{
  "name": "ohos.permission.GET_NETWORK_INFO"
},
...
```

接入调试

在开发过程中可以启用调试模式，借助控制台日志实时查看上报日志的记录情况，观察具体结果并根据需要进行调整。

需要通过调用LTSSDK的静态方法来修改，SDK支持Debug、Info、Warning、Error和Off 5个级别的默认为静默输出（Off）。5个级别分别对应的参数：LogLevel.DEBUG、LogLevel.INFO、LogLevel.WARNING、LogLevel.ERROR、LogLevel.OFF。代码示例如下：

```
ltssdk.setLogLevel(LogLevel.DEBUG)
```

参数获取方式

- 区域表

表 4-103 区域表

区域名称	区域
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击“日志组名称”可查看日志流列表，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.9 云日志服务 Web SDK

云日志服务Web SDK提供了Web SDK上报日志的一系列方法，如果您需要收集和分析用户在网站上的信息，例如用户的浏览器、浏览行为记录、终端设备记录，设备异常记录，网络使用记录等，可以直接使用Web SDK上报日志到LTS。

使用Web SDK上报日志到LTS的场景下，需要开启日志流的匿名写入功能，开启后上报日志没有经过有效鉴权，可能产生脏数据。详细操作请参考[管理日志流](#)。

说明

Web SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请[提交工单](#)申请。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请您根据所在区域，选择RegionName。
- 获取华为云账号的项目ID（project id），步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

版本更新说明

表 4-104 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.21	• 废弃config方法，优先使用new SDK创建一个新的实例。 • 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	• 调整日志级别等级。 • 支持labels嵌套。
1.0.15	新增多实例。

使用说明

为确保Web小程序日志上报正常，需将上报地址添加到Web开发者平台域名列表，上报地址请参见[配置参数说明](#)。

NPM 方式安装 SDK

1. 安装方法：在项目根目录下通过运行“npm i lts-web-sdk”命令，安装SDK软件包。您可以在[开源仓地址](#)下载最新的SDK。

2. 示例代码：

```
const LTS_WEB_SDK = require('lts-web-sdk').default;
// import LTS_WEB_SDK from 'lts-web-sdk';
// 初始化
const weblog = new LTS_WEB_SDK({
  // 上报region
  region: string,
  // 项目ID
  projectId: string,
  // 上报地址
  url: string,
  // LTS日志组ID
  groupId: string,
  // LTS日志流ID
  streamId: string,
  // 调试日志等级
  debug: string,
  // 上报条数阈值
  cacheThreshold: number,
  // 上报时间阈值
```

```
    timeInterval: number,
  });
// 立即上报单条带标签
weblog.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
// 立即上报单条 不带标签
weblog.reportImmediately({ key: 'value', number: 1, array: [], json: { json: 'json' } }, { 'key': 'value' });
// 缓存上报多条 带标签
weblog.report({ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }, { 'key': 'value' });
// 缓存上报多条 不带标签
weblog.report({ 'name': 'xiaohong', 'age': 18 }, { key: 'value', number: 1, array: [], json: { json: 'json' } });
// 缓存上报多条 带多个标签（最多50个）
weblog.report({ 'name': 'xiaohong', 'name': 'xiaolan' }, { 'version': '1.0.0', 'render': 'web', 'link': '/', from: 'web' });
```

CDN 同步方式安装 SDK

1. 安装方法：

在您的html文件中，通过以下方式引用SDK。

```
<script src="https://res.hc-cdn.com/web-sdk-cdn/版本号/websdk.min.js"></script>
```

2. 示例代码：

```
// 页面引入SDK
<script src="https://res.hc-cdn.com/web-sdk-cdn/1.0.15/websdk.min.js"></script>
// 初始化
const weblog = new LTS_WEB_SDK({
  // 上报region
  region: string,
  // 项目ID
  projectId: string,
  // 上报地址
  url: string,
  // LTS日志组ID
  groupId: string,
  // LTS日志流ID
  streamId: string,
  // 调试开关，开启后可以看到输出的调试日志
  debug: boolean,
  // 上报条数阈值
  cacheThreshold: number,
  // 上报时间条数
  timeInterval: number,
});
// 立即上报单条带标签
weblog.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
// 立即上报单条 不带标签
weblog.reportImmediately({ key: 'value', number: 1, array: [], json: { json: 'json' } }, { 'key': 'value' });
// 缓存上报多条 带标签
weblog.report({ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }, { 'key': 'value' });
// 缓存上报多条 不带标签
weblog.report({ 'name': 'xiaohong', 'age': 18 }, { key: 'value', number: 1, array: [], json: { json: 'json' } });
// 缓存上报多条 带多个标签（最多50个）
weblog.report({ 'name': 'xiaohong', 'name': 'xiaolan' }, { 'version': '1.0.0', 'render': 'web', 'link': '/', from: 'web' });
```

配置参数说明

- 配置参数说明

表 4-105 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。
url	string	选填	-	用于上报日志的公网地址域名，支持指定端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443。如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com - 使用SDK在跨云或本地上报日志时，端口使用443。 - 使用华为云主机在华为云内网上报日志时，端口使用8102。
debug	string	选填	OFF	- 控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级。 - 填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。 - 值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 30 <= cacheThreshold <= 1000
timeInterval	number	选填	3	默认3s 上报阈值时间，当达到阈值时间上报缓存中的数据 1 <= timeInterval <= 1800

- 日志上报Report方法参数说明。

表 4-106 Report 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	必填	-	需要上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明。

表 4-107 reportImmediately 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	选填	-	立即上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的日志。 用户的日志字段名称不允许包含双下划线（__）。

字段	类型	是否必填	默认值	描述
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

表 4-108 区域表

区域名称	区域
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 ，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.10 云日志服务 iOS SDK

云日志服务iOS SDK提供了Objective-C & Swift语言上报日志的一系列方法，如果您需要收集和分析用户在网站上的信息，例如用户的浏览器、浏览行为记录、终端设备记录，设备异常记录，网络使用记录等，可以直接使用iOS SDK上报日志到LTS。

说明

当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请[提交工单](#)申请。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请您根据所在区域，选择RegionName。

- 获取华为云账号的项目ID（ project id ），步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

版本更新说明

SDK如何处理个人信息请参考[华为云日志服务移动端日志采集SDK隐私声明](#)。

您集成和使用的SDK时需要遵从个人信息保护基本要求，详情请参考[华为云日志服务移动端日志采集SDK开发者合规指南](#)。

表 4-109 版本更新说明

版本号	下载地址	检验信息下载地址	更新说明	系统
1.0.27	单击 下载	单击 下载	优化性能。	iOS10、Xcode11及以上。
1.0.26	单击 下载	单击 下载	优化性能。	iOS10、Xcode11及以上。
1.0.24	单击 下载	单击 下载	● 支持更多region：华东-上海一、华南-广州。 ● 优化内存使用、日志发送等。	iOS10、Xcode11及以上。
1.0.21	单击 下载	单击 下载	● 优化初始化函数 initWithConfig。 ● 为配置函数config添加废弃声明。	iOS10、Xcode11及以上。
1.0.19	单击 下载	单击 下载	● 调整配置项阈值。 ● 修复兼容性问题。	iOS10、Xcode11及以上。
1.0.18	单击 下载	单击 下载	● 增加设置调试日志级别接口： setLogLevel。 ● 在控制台输出配置项、上报接口参数、请求发送内容。 ● 上报接口参数labels支持嵌套。 ● 请求发送控制、数据库大小限制等优化。	iOS10、Xcode11及以上。
1.0.15	单击 下载	单击 下载	首次发布，提供四个接口：初始化 initWithConfig、配置 config、上报日志 report、立即上报日志 reportImmediately。	iOS10、Xcode11及以上。

安装 iOS SDK

步骤1 集成接入SDK。

方法1：通过CocoaPods集成。

1. 在Podfile中添加CocoaPods官方仓库。
`source 'https://github.com/CocoaPods/Specs.git'`
2. 在Podfile中添加依赖。
`pod 'LTSSDK', '1.0.27'`
3. 在终端执行。
`pod install --repo-update`

方法2：手动集成。

- 下载SDK，解压到指定目录。注：直接解压即可，不需要额外操作。
- 将解压后的xcframework静态库添加到您的项目工程中。
待导入的xcframework静态库必须和工作空间在相同的磁盘空间里，如果不在，您可以选择注意勾选“Copy items if needed”和“Create groups”，将待导入的xcframework库工程文件复制到工作空间。

步骤2 初始化，详细参数请参考表4-112。

- Objective-C：初始化代码示例。

```
// LTS参数配置
LTSSDKConfigParams *params = [[LTSSDKConfigParams alloc] init];
// 必填参数
params.region = @"云日志服务的区域";
params.projectId = @"账号的项目ID";
params.groupId = @"LTS的日志组ID";
params.streamId = @"LTS的日志流ID";
// 注意：认证用的账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
params.accessKey = @"访问密钥";
params.secretKey = @"秘密访问密钥";
// 选填参数
// params.url = @"url";
// params.cacheThreshold = 200;
// params.timeInterval = 3;
// params.reportWhenEnterBackgroundEnabled = YES;
// params.reportWhenAPPLaunchEnabled = NO;
// LTS初始化方法
LTSSDK *lts = [[LTSSDK alloc] initWithConfig:params];
```

- Swift：初始化代码示例。

LTSSDK支持多实例日志上报机制，根据不同配置创建实例进行上报。

```
// LTS参数配置
let params = LTSSDKConfigParams()
// 必填参数
params.region = "云日志服务的区域"
params.projectId = "账号的项目ID"
params.groupId = "LTS的日志组ID"
params.streamId = "LTS的日志流ID"
// 注意：认证用的账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
params.accessKey = "访问密钥"
params.secretKey = "秘密访问密钥"
// 选填参数
// params.url = @"url";
// params.cacheThreshold = 200
// params.timeInterval = 3
// params.reportWhenEnterBackgroundEnabled = true
// params.reportWhenAPPLaunchEnabled = false
// LTS初始化方法
let lts = LTSSDK(config:params)
```

步骤3 上报日志。

LTS提供两种上报日志的方法。

表 4-110 两种上报日志的方法

Method	Description
- (BOOL)report:(id)content labels:(nullable NSDictionary<NSString *, id> *)labels;	上报日志：先存入本地数据库，根据配置中设定的策略实时上报。
- (BOOL)reportImmediately:(id)content labels:(nullable NSDictionary<NSString *, id> *)labels;	立即上报日志。

表 4-111 Parameters 参数

Name	Description
content	日志内容。支持字典或字典数组；最外层键值对最多300个；字典转JSON字符串最大支持长度为30720，超出部分将被截断再上报。
labels	日志标签。支持字典或空值；最外层键值对最多50个；最外层key最大长度为64，支持字母、数字和下划线组合，首字符须是字母；字典转JSON字符串最大支持长度为30720，超出则该条日志无法上报。

- Objective-C代码示例如下：

```
//单个上报
[its report:@{@"content_key": @"content_value"} labels:@{@"label_key": @"label_value"}];
[its reportImmediately:@{@"content_key": @"content_value"} labels:@{@"label_key":
@"label_value"}];

//批量上报
[its report:@{@"content1_key": @"content1_value"}, @{@"content2_key": @"content2_value"}]
labels:@{@"label_key": @"label_value"}];
[its reportImmediately:@{@"content1_key": @"content1_value"}, @{@"content2_key":
@"content2_value"}] labels:@{@"label_key": @"label_value"}];
```

- Swift代码示例如下：

```
//单个上报
its.report(["content_key": "content_value"], labels: ["label_key": "label_value"])
its.reportImmediately(["content_key": "content_value"], labels: ["label_key": "label_value"])

//批量上报
its.report(["content1_key": "content1_value", ["content2_key": "content2_value"]], labels:
["label_key": "label_value"])
its.reportImmediately(["content1_key": "content1_value", ["content2_key": "content2_value"]],
labels: ["label_key": "label_value"])
```

----结束

配置参数说明

表 4-112 初始化参数说明

参数名称	类型	是否必填	默认值	描述
projectId	NSString	必填	-	账号的项目ID。
accessKey	NSString	必填	-	账号的访问密钥，简称AK。 注意：硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
secretKey	NSString	必填	-	账号的秘密访问密钥，简称SK。 注意：硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
region	NSString	必填	-	LTS的服务区域。
groupId	NSString	必填	-	LTS的日志组ID。
streamId	NSString	必填	-	LTS的日志流ID。
url	NSString	选填	nil	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com - 使用SDK在跨云或本地上报日志时，端口使用443。 - 使用华为云主机在华为云内网上报日志时，端口使用8102。
cacheThreshold	NSUInteger	选填	200条	当本地数据库日志存储条数达到该值会触发上报，取值范围为30-1000条。
timeInterval	NSUInteger	选填	3秒	定时器每隔该值会触发上报，取值范围为1-1800秒。

参数名称	类型	是否必填	默认值	描述
reportWhenEnterBackgroundEnabled	BOOL	选填	YES	是否开启APP切后台上报功能。
reportWhenAPPLaunchEnabled	BOOL	选填	NO	是否开启APP启动时上报功能。如需开启，请在UIApplicationDidFinishLaunchingNotification通知发送前完成配置。

上报日志策略说明

LTSSDK支持四种日志上报策略：阈值上报、定时上报、应用切后台和启动时上报，根据场景选择合适的值配置cacheThreshold、timeInterval、reportWhenEnterBackgroundEnabled、reportWhenAPPLaunchEnabled等参数。

开启调试

在开发过程中可以启用调试模式，借助控制台日志实时查看上报日志的记录情况，观察具体结果并根据需要进行调整。在初始化LTSSDK之前，调用类方法+(void)setLogLevel:(LTLoggerLevel)logLevel设置调试级别：Debug、Info、Warn、Error、Off。

Objective-C语言示例代码：

```
#ifdef DEBUG
[LTSSDK setLogLevel:LTLoggerLevelDebug];
#endif
```

Swift语言示例代码：

```
#if DEBUG
LTSSDK.setLogLevel(.debug)
#endif
```

参数获取方式

- 区域表

表 4-113 区域表

区域名称	区域
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。

- 日志流ID：单击“日志组名称”可查看日志流列表，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.11 云日志服务 Android SDK

云日志服务Android SDK提供了Kotlin & Java语言上报日志的一系列方法，如果您需要收集和分析用户在网站上的信息，例如用户的浏览器、浏览行为记录、终端设备记录，设备异常记录，网络使用记录等，可以直接使用Android SDK上报日志到LTS。

 说明

当前仅华北-北京四、华东-上海一、华南-广州区域的白名单用户支持使用该功能，如有需要请[提交工单](#)申请。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请您根据所在区域，选择RegionName。
- 获取华为云账号的项目ID（ project id ），步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

版本更新说明

SDK如何处理个人信息请参考[华为云日志服务移动端日志采集SDK隐私声明](#)。

您集成和使用的SDK时需要遵从个人信息保护基本要求，详情请参考[华为云日志服务移动端日志采集SDK开发者合规指南](#)。

表 4-114 版本更新说明

版本号	下载地址	检验信息 下载地址	更新说明	系统
1.0.28	单击 下载	单击 下载	修复了1.0.26版本以来工程配置导致的配置冲突。	Android 7 及以上
0.0.5	请通过maven Central 下载。	-	功能与1.0.28版本相同，可支持 kotlin 1.3.x。	Android 7 及以上。
1.0.27	单击 下载	单击 下载	优化了缓存效率。	Android 7 及以上
0.0.4	请通过maven Central 下载。	-	功能与1.0.27版本相同，可支持 kotlin 1.3.x。	Android 7 及以上。

版本号	下载地址	检验信息 下载地址	更新说明	系统
1.0.26	单击 下载	单击 下载	修复了因异步修改入参导致数据异常的问题。	Android 7 及以上。
1.0.25	单击 下载	单击 下载	增强代码强壮性，添加参数保护。	Android 7 及以上。
1.0.24	单击 下载	单击 下载	- 支持更多region：华东-上海一、华南-广州。 - 修改了程序目录，修改为 com.cloud.lts.*。	Android 7 及以上。
1.0.21	单击 下载	单击 下载	废弃setconfig方法，使用LTSSDK的构造方法代替。	Android 7 及以上。
1.0.19	单击 下载	单击 下载	- 修改了上报时间间隔阈值。 - 修复了修改配置信息后导致无法获取缓存的问题。	Android 7 及以上。
1.0.18	单击 下载	单击 下载	- 增强代码强壮性，修复多实例时可能会产生的崩溃。 - 修复最低 Android API 支持版本定义错误的问题。 - 增加设置调试日志级别接口：setLogLevel。 - 修改了 日志发送的入参 label 的类型，支持多层嵌套。 - 日志发送入参 content 支持数组类型，方便使用。 - 添加缓存存储条目最大值支持，如果超过会丢弃后再存储。 - 修改了请求失败时的控制策略，支持递进式等待。 - 对本地错误日志输出做了整理。	Android 7 及以上。
1.0.17	单击 下载	单击 下载	首次发布，提供四个接口：初始化、配置config、上报日志report、立即上报日志reportImmediately。	Android 7 及以上。

安装 Android SDK

步骤1 集成接入SDK。

1. maven仓库集成。

- a. 在build.gradle中添加依赖。

```
plugins {  
    ...  
    id 'org.jetbrains.kotlin.android' version 'KOTLIN_VERSION' apply false // 添加kotlin插件，1.x.x  
    版本最低支持1.6.20版本，0.0.x版本最低可支持1.3.61版本  
    ...  
}
```

- b. 在app/build.gradle中添加依赖。

```
dependencies {  
    ...  
    implementation 'io.github.lts-sdk:lts-sdk-android:1.0.28'  
    ...  
}
```

- c. 在 settings.gradle中添加maven仓库源。

```
pluginManagement{  
    ...  
    repositories {  
        ...  
        // 加入下面内容  
        mavenCentral()  
    }  
}  
dependencyResolutionManagement {  
    ...  
    repositories {  
        ...  
        // 加入下面内容  
        mavenCentral()  
    }  
}
```

2. 手动集成。

- a. 下载日志SDK包。
b. 下载后解压到指定目录。注：直接解压即可，不需要额外操作。
c. 将解压后的aar静态库文件添加到您的项目工程中。

- d. 在build.gradle中添加依赖。

```
plugins {  
    ...  
    id 'org.jetbrains.kotlin.android' version 'KOTLIN_VERSION' apply false // 添加kotlin插件，1.x.x  
    最低支持1.6.20版本，0.0.x最低可支持1.3.61版本  
    ...  
}
```

- e. 在app/build.gradle中添加依赖。

```
dependencies {  
    ...  
    implementation fileTree(dir: $dir, include: ['*.aar']) // 填写aar所在的文件夹，例如'libs'  
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.4.0' // 0.0.x版本依赖版本为1.3.3  
    implementation 'androidx.core:core-ktx:1.7.0' // 0.0.x版本依赖版本为1.2.0  
    implementation 'com.google.code.gson:gson:2.8.9'  
    kapt 'androidx.room:room-compiler:2.3.0' // 0.0.x版本依赖版本为2.2.6  
    api 'androidx.room:room-runtime:2.3.0' // 0.0.x版本依赖版本为2.2.6  
    api 'androidx.room:room-common:2.3.0' // 0.0.x版本依赖版本为2.2.6  
    ...  
}
```

- f. 在 settings.gradle中添加maven仓库源。

如果您的Gradle版本低于7.0 则需要将maven仓库源添加到build.gradle文件中。

```
pluginManagement{  
    ...  
    repositories {  
        ...  
        // 加入下面内容
```

```
        google()
        mavenCentral()
    }
}
dependencyResolutionManagement {
    ...
    repositories {
        ...
        // 加入下面内容
        google()
        mavenCentral()
    }
}
```

步骤2 初始化，详细参数请参考表4-117。

1. 添加依赖

```
import com.cloud.lts.UserConfig
import com.cloud.lts.LTSSDK
```

2. Kotlin：初始化代码示例。

//注意：认证用的账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。

```
val ak: String = getData("HUAWEICLOUD_SDK_AK")
val sk: String = getData("HUAWEICLOUD_SDK_SK")
// LTS参数配置
val userConfig = UserConfig.Builder()
// 必填参数
.setRegion(USER_REGION) // 云日志服务的区域
.setProjectId(PROJECT_ID) // 账号的项目ID
.setGroupId(LOG_GROUP_ID) // LTS的日志组ID
.setStreamId(LOG_STREAM_ID) // LTS的日志流ID
.setAccessKey(ak) // 访问密钥
.setSecretKey(sk) // 秘密访问密钥
// 选填参数
.setUrlHost(url) // LTS的公网地址域名
.setCacheThreshold(200)
.setTimeInterval(60)
.setIsReportBackground(true)
.setIsReportLaunch(false)
.build()
// LTS初始化方法
ltssdk = LTSSDK( </* Application Instance */>, userConfig)
```

3. Java：初始化代码示例。LTSSDK支持多实例日志上报机制，根据不同配置创建实例进行上报。

// 注意：认证用的账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。

```
String ak = getData("HUAWEICLOUD_SDK_AK")
String sk = getData("HUAWEICLOUD_SDK_SK")
// LTS参数配置
UserConfig userConfig = new UserConfig.Builder()
// 必填参数
.setRegion(USER_REGION) // 云日志服务的区域
.setProjectId(PROJECT_ID) // 账号的项目ID
.setGroupId(LOG_GROUP_ID) // LTS的日志组ID
.setStreamId(LOG_STREAM_ID) // LTS的日志流ID
.setAccessKey(ak) // 访问密钥
.setSecretKey(sk) // 秘密访问密钥
// 选填参数
.setUrlHost(url) // LTS的公网地址域名
.setCacheThreshold(200)
.setTimeInterval(60)
.setIsReportBackground(true)
.setIsReportLaunch(false)
.build();
// LTS初始化方法
ltssdk = new LTSSDK( </* Application Instance */>, userConfig);
```


步骤3 上报日志。 LTS提供两种上报日志的方法。**表 4-115** 两种上报日志的方法

Method	Description
report(content, labels)	上报日志：先存入本地数据库，根据配置中设定的策略实时上报。
reportImmediately(content, labels)	立即上报日志。

表 4-116 Parameters 参数

Name	Description
content	日志内容，支持字典和字典数组；键值对最多300个；content转JSON字符串最大支持长度为30*1024，超出部分被截断。
labels	日志标签。 - 支持字典或空值。 - 最外层键值对最多50个。 - 最外层key最大长度为64。 - 支持字母、数字和下划线组合，首字符须是字母。 - 字典转JSON字符串最大支持长度为30720，超出则该条日志无法上报。

• Kotlin代码示例如下：

```
var ltssdk: LTSSDK
val food = hashMapOf(Pair("food_1", "rice"), Pair("food_2", "wheat"), Pair("food_3", "egg"))
val fruit = hashMapOf(Pair("fruit_1", "apple"), Pair("fruit_2", "pear"), Pair("fruit_3", "banana"))
val label = hashMapOf(Pair("date", "2023-10-01"))
val contents = listOf(food, fruit).toTypedArray()
ltssdk.report(food, label) // 缓存上报单条 带标签
ltssdk.report<String, String>(food) // 缓存上报单条 不带标签
ltssdk.reportImmediately(food, label) // 立即上报单条 带标签
ltssdk.reportImmediately<String, String>(food) // 立即上报单条 不带标签
ltssdk.report(contents, label) // 缓存上报多条 带标签
ltssdk.report<String, String>(contents) // 缓存上报多条 不带标签
ltssdk.reportImmediately(contents, label) // 立即上报多条 带标签
ltssdk.reportImmediately<String, String>(contents) // 立即上报多条 不带标签
```

• Java代码示例如下：

```
LTSSDK ltssdk;
HashMap fruit = new HashMap<String, String>(){
    put("fruit_1", "apple");
    put("fruit_2", "pear");
    put("fruit_3", "banana");
};
HashMap food = new HashMap<String, String>(){
    put("food_1", "rice");
    put("food_2", "wheat");
    put("food_3", "egg");
};
```

```
HashMap<String, String> labels = new HashMap(){  
    put("date","2023-10-01");  
};  
HashMap[] contents = {food, fruit};  
ltssdk.report(food); // 缓存上报单条 不带标签  
ltssdk.report(food, labels); // 缓存上报单条 带标签  
ltssdk.reportImmediately(food); // 立即上报单条 不带标签  
ltssdk.reportImmediately(food, labels); // 立即上报单条 带标签  
ltssdk.report(contents); // 缓存上报多条 不带标签  
ltssdk.report(contents, labels); // 缓存上报多条 带标签  
ltssdk.reportImmediately(contents); // 立即上报多条 不带标签  
ltssdk.reportImmediately(contents, labels); // 立即上报多条 带标签
```

----结束

配置参数说明

表 4-117 初始化参数说明

参数名称	类型	是否必填	默认值	描述
projectId	String	必填	-	账号的项目ID。
accessKey	String	必填	-	账号的访问密钥，简称AK。 认证用的账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
secretKey	String	必填	-	账号的秘密访问密钥，简称SK。注意：认证用的华为云账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
region	String	必填	-	LTS的服务区域。
groupId	String	必填	-	LTS的日志组ID。
streamId	String	必填	-	LTS的日志流ID。
url	String	选填	null	用于上报的公网地址域名，如未设置，将根据region自动生成链接，格式如下：https://{region}.myhuaweicloud.com
cacheThreshold	Long	选填	200条	当本地数据库日志存储条数达到该值会触发上报，取值范围为30-1000条。
timeInterval	Long	选填	3秒	定时器每隔该值会触发上报，取值范围为1-1800秒。

参数名称	类型	是否必填	默认值	描述
isReportBackground	boolean	选填	true	是否开启APP切入后台时上报功能。
isReportLaunch	boolean	选填	false	是否开启APP启动时上报功能。

上报日志策略说明

LTSSDK支持四种日志上报策略：阈值上报、定时上报、应用切后台和启动时上报，根据场景选择合适的值配置cacheThreshold、timeInterval、isReportBackground、isReportLaunch等参数。

权限配置

LTSSDK 需要 Android 的网络权限，如果您没有配置，请将下面内容的写入APP 的 Manifest.xml 文件中：

```
manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
...
// 加入下面的内容
<uses-permission android:name="android.permission.INTERNET" />
...
</manifest>
```

混淆配置

如App对代码进行乱序混淆，则在混淆配置文件中添加代码段：

```
-keep class com.cloud.**{*};
```

接入调试

在开发过程中可以启用调试模式，借助控制台日志实时查看上报日志的记录情况，观察具体结果并根据需要进行调整。

需要通过调用LTSSDK的静态方法来修改，SDK支持Debug、Info、Warning 、Error 和 Off 5个级别的默认为静默输出（Off）。5个级别分别对应的参数：LogLevel.DEBUG、LogLevel.INFO、LogLevel.WARNING、LogLevel.ERROR、LogLevel.OFF。代码示例如下：

```
LTSSDK.setLogLevel(LogLevel.DEBUG)
```

参数获取方式

- 区域表

表 4-118 区域表

区域名称	区域
华北-北京四	cn-north-4

区域名称	区域
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击“日志组名称”可查看日志流列表，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.12 云日志服务百度小程序 SDK

云日志服务百度小程序SDK提供了百度小程序上报日志的一系列方法，如果您需要收集和分析用户在网站上的信息，例如用户的浏览器、浏览行为记录、终端设备记录，设备异常记录，网络使用记录等，可以直接使用百度小程序SDK上报日志到LTS。

使用百度小程序SDK上报日志到LTS的场景下，需要开启日志流的匿名写入功能，开启后上报日志没有经过有效鉴权，可能产生脏数据。详细操作请参考[管理日志流](#)。

说明

百度小程序SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请[提交工单](#)申请。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请您根据所在区域，选择RegionName。
- 获取华为云账号的项目ID（project id），步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

版本更新说明

表 4-119 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	• 废弃config方法，优先使用new SDK创建一个新的实例。 • 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。

版本号	更新说明
1.0.18	- 调整日志级别等级。 - 支持labels嵌套。
1.0.15	小程序支持创建多个日志流同时上报日志到LTS。

使用说明

需将上报地址添加到百度开发者平台域名列表中，上报地址请参考[配置参数说明](#)。

安装 SDK

在小程序的根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。您可以在[开源仓地址](#)下载最新的SDK。

示例代码

```
const miniSDK = require('lts-mini-sdk').default;
// import miniSDK from 'lts-mini-sdk';
App({
  onLaunch(options) {
    // 初始化
    const baidumini = new miniSDK({
      // 上报region
      region: string,
      // 华为云项目ID
      projectId: string,
      // 上报地址
      url: string,
      // LTS日志组ID
      groupId: string,
      // LTS日志流ID
      streamId: string,
      // 日志所属组
      group: string,
      // 调试日志等级
      debug: string,
      // 当前小程序所属平台
      platform: string,
      // 上报条数阈值
      cacheThreshold: number,
      // 上报时间阈值
      timeInterval: number,
    });
    // 立即上报单条带标签
    baidumini.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
    // 立即上报单条 不带标签
    baidumini.reportImmediately([{ key: 'value', number: 1, array: [], json: { json: 'json' } }, { 'key': 'value' }]);
    // 缓存上报多条 带标签
    baidumini.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }, { 'key': 'value' }]);
    // 缓存上报多条 不带标签
    baidumini.report([{ 'name': 'xiaohong', 'age': 18 }, { key: 'value', number: 1, array: [], json: { json: 'json' } }]);
    // 缓存上报多条 带多个标签（最多50个）
    baidumini.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }, { 'version': '1.0.0', 'render': 'mini', 'link': '/', from: 'baidu' }]);
  }
});
```

配置参数说明

- 配置参数说明

表 4-120 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。
url	string	选填	-	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。该域名地址需要在百度小程序后台开发设置服务器域名处（具体操作界面以小程序官网为准）注册为合法域名，128个字符上限。 - 使用SDK在跨云或本地上报日志时，端口使用443。 - 使用华为云主机在华为云内网上报日志时，端口使用8102。
debug	string	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 30 <= cacheThreshold <= 1000。
timeInterval	number	选填	3	默认3s 上报阈值时间，当达到阈值时间上报缓存中的数据 1 <= timeInterval <= 1800。

字段	类型	是否必填	默认值	描述
platform	string	选填	baidu	支持上报的平台。目前支持以下平台： • baidu：百度小程序。 • wx：微信小程序。 • dd：钉钉小程序。 • my：支付宝小程序。 • quick：快应用。
group	string	选填	-	日志组ID。

- 日志上报report方法参数说明

表 4-121 report 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	必填	-	需要上报的日志对象或者对象数组，每个对象（Object）最多允许300个键值对，且转成JSON字符串最大支持长度1024*30个字符，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	-	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

表 4-122 reportImmediately 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	选填	-	立即上报的日志对象或者对象数组，每个对象（Object）最多允许300个键值对，且转成JSON字符串最大支持长度1024*30个字符，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的数据。用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	-	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

表 4-123 区域表

区域名称	区域
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.13 云日志服务微信小程序 SDK

云日志服务微信小程序SDK提供了微信小程序上报日志的一系列方法，如果您需要收集和分析用户在网站上的信息，例如用户的浏览器、浏览行为记录、终端设备记录，设备异常记录，网络使用记录等，可以直接使用微信小程序SDK上报日志到LTS。

使用微信小程序SDK上报日志到LTS的场景下，需要开启日志流的匿名写入功能，开启后上报日志没有经过有效鉴权，可能产生脏数据。详细操作请参考[管理日志流](#)。

说明

微信小程序SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请[提交工单](#)申请。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请您根据所在区域，选择RegionName。
- 获取华为云账号的项目ID（project id），步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

版本更新说明

表 4-124 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	• 废弃config方法，优先使用new SDK创建一个新的实例。 • 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	• 调整日志级别等级。 • 支持labels嵌套。
1.0.15	小程序支持创建多个日志流同时上报日志到LTS。

使用说明

为确保微信小程序日志上报正常，需将上报地址添加到微信开发者平台域名列表，上报地址请参见[配置参数说明](#)。

安装 SDK

在小程序的根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。

您可以在[开源仓地址](#)下载最新的SDK。可以将软件包中的app.js复制到项目文件目录下使用或者项目开启npm，使用node_module下的文件。

集成 SDK

- 采用npm方式集成SDK。
 - a. 确保项目有“package.json”文件，如果项目中没有“package.json”文件，可以在项目的根目录下，使用以下命令来创建：npm init。
 - b. 运行安装SDK的命令，安装SDK软件包。
 - c. 单击开发者工具菜单栏中的“工具 > 构建npm”，构建当前工程的npm库文件。
- 采用文件引入方式集成SDK。
 - a. 运行安装SDK的命令，安装SDK软件包。
 - b. 找到SDK文件夹中的app.js文件“node_modules > lts-mini-sdk > app.js”，将app.js文件从node_module复制到根路径并改名。

示例代码

```
const miniSDK = require('lts-mini-sdk').default;
// import miniSDK from 'lts-mini-sdk';
App({
  onLaunch(options) {
    // 初始化
    const wxmini = new miniSDK({
      // 上报region
      region: string,
      // 项目ID
      projectId: string,
      // 上报地址
      url: string,
      // LTS日志组ID
      groupId: string,
      // LTS日志流ID
      streamId: string,
      // 调试日志等级
      debug: string,
      // 当前小程序所属平台
      platform: string,
      // 上报条数阈值
      cacheThreshold: number,
      // 上报时间阈值
      timeInterval: number,
    });
    // 立即上报单条带标签
    wxmini.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
    // 立即上报单条 不带标签
    wxmini.reportImmediately([{ key: 'value', number: 1, array: [], json: { json: 'json' } }, { 'key':
'value' }]);
    // 缓存上报多条 带标签
    wxmini.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }, { 'key': 'value' }]);
    // 缓存上报多条 不带标签
    wxmini.report([{ 'name': 'xiaohong', 'age': 18 }, { key: 'value', number: 1, array: [], json: { json:
'json' } }]);
    // 缓存上报多条 带多个标签（最多50个）
    wxmini.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }, { 'version': '1.0.0', 'render': 'mini', 'link': '/', from:
'wx' }]);
  }
});
```

配置参数说明

- 配置参数说明

表 4-125 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。
url	string	选填	-	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。该域名地址需要在微信小程序后台开发设置服务器域名处（具体操作界面以小程序官网为准）注册为合法域名，128个字符上限。 - 使用SDK在跨云或本地上报日志时，端口使用443。 - 使用华为云主机在华为云内网上报日志时，端口使用8102。
debug	string	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 30 <= cacheThreshold <= 1000
timeInterval	number	选填	3	默认3s 上报阈值时间，当达到阈值时间上报缓存中的数据 1 <= timeInterval <= 1800
platform	string	选填	baidu	支持上报的平台。目前支持以下平台：baidu：百度小程序，wx：微信小程序，dd：钉钉小程序，my：支付宝小程序，quick：快应用

- 日志上报report方法参数说明

表 4-126 report 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	必填	-	需要上报的日志对象或者对象数组，每个对象（Object）最多允许300个键值对，且转成JSON字符串最大支持长度1024*30个字符，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

表 4-127 reportImmediately 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	选填	-	立即上报的日志对象或者对象数组，每个对象（Object）最多允许300个键值对，且转成JSON字符串最大支持长度1024*30个字符，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的日志。 用户的日志字段名称不允许包含双下划线（__）

字段	类型	是否必填	默认值	描述
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）

参数获取方式

- 区域表

表 4-128 区域表

区域名称	区域
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.14 云日志服务钉钉小程序 SDK

云日志服务钉钉小程序SDK提供了钉钉小程序上报日志的一系列方法，如果您需要收集和分析用户在网站上的信息，例如用户的浏览器、浏览行为记录、终端设备记录、设备异常记录、网络使用记录等，可以直接使用钉钉小程序SDK上报日志到LTS。

使用钉钉小程序SDK上报日志到LTS的场景下，需要开启日志流的匿名写入功能，开启后上报日志没有经过有效鉴权，可能产生脏数据。详细操作请参考[管理日志流](#)。

 说明

钉钉小程序SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请[提交工单](#)申请。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请您根据所在区域，选择RegionName。
- 获取华为云账号的项目ID（project id），步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

版本更新说明

表 4-129 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	• 废弃config方法，优先使用new SDK创建一个新的实例。 • 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	• 调整日志级别等级。 • 支持labels嵌套。
1.0.15	小程序支持创建多个日志流同时上报日志到LTS。

使用说明

为确保钉钉小程序日志上报正常，需将上报地址添加到钉钉开发者平台域名列表，上报地址请参见[配置参数说明](#)。

安装 SDK

在小程序的根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。

您可以在[开源仓地址](#)下载最新的SDK。

示例代码

```
const miniSDK = require('lts-mini-sdk').default;
// import miniSDK from 'lts-mini-sdk';
App({
  onLaunch(options) {
    // 初始化
    const ddmini = new miniSDK({
      // 上报region
      region: string,
      // 华为云项目ID
      projectId: string,
      // 上报地址
      url: string,
      // LTS日志组ID
      groupId: string,
      // LTS日志流ID
    });
  }
});
```

```
streamId: string,
// 调试日志等级
debug: string,
// 当前小程序所属平台
platform: 'dd',
// 上报条数阈值
cacheThreshold: number,
// 上报时间阈值
timeInterval: number,
});
// 立即上报单条带标签
ddmini.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
// 立即上报单条 不带标签
ddmini.reportImmediately([{ key: 'value', number: 1, array: [], json: { json: 'json' } }], { 'key':
'value' });
// 缓存上报多条 带标签
ddmini.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value' });
// 缓存上报多条 不带标签
ddmini.report([{ 'name': 'xiaohong', 'age': 18 }, { key: 'value', number: 1, array: [], json: { json:
'json' } }]);
// 缓存上报多条 带多个标签（最多50个）
ddmini.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], { 'version': '1.0.0', 'render': 'mini', 'link': '/', from:
'dd' });
}
});
```

配置参数说明

- 配置参数说明

表 4-130 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。

字段	类型	是否必填	默认值	描述
url	string	选填	https://lts-access.cn-north-4.myhuaweicloud.com	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。该域名地址需要在钉钉小程序开发设置服务器域名处（具体操作界面以小程序官网为准）注册为合法域名，128个字符上限。 - 使用SDK在跨云或本地上报日志时，端口使用443。 - 使用华为云主机在华为云内网上报日志时，端口使用8102。
debug	boolean	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 $30 \leq \text{cacheThreshold} \leq 1000$ 。
timeInterval	number	选填	3	默认3s 上报阈值时间，当达到阈值时间上报缓存中的数据 $1 \leq \text{timeInterval} \leq 1800$ 。
platform	string	选填	baidu	支持上报的平台。目前支持以下平台：baidu：百度小程序，wx：微信小程序，dd：钉钉小程序，my：支付宝小程序，quick：快应用。

- 日志上报report方法参数说明

表 4-131 report 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	必填	-	需要上报的日志对象或者对象数组，每个对象（Object）最多允许300个键值对，且转成JSON字符串最大支持长度1024*30个字符，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

表 4-132 reportImmediately 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	选填	-	立即上报的日志对象或者对象数组，每个对象（Object）最多允许300个键值对，且转成JSON字符串最大支持长度1024*30个字符，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的日志。 用户的日志字段名称不允许包含双下划线（__）。

字段	类型	是否必填	默认值	描述
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

表 4-133 区域表

区域名称	区域
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击“日志组名称”可查看日志流列表，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.15 云日志服务支付宝小程序 SDK

云日志服务支付宝小程序SDK提供了支付宝小程序上报日志的一系列方法，如果您需要收集和分析用户在网站上的信息，例如用户的浏览器、浏览行为记录、终端设备记录、设备异常记录、网络使用记录等，可以直接使用支付宝小程序SDK上报日志到LTS。

使用支付宝小程序SDK上报日志到LTS的场景下，需要开启日志流的匿名写入功能，开启后上报日志没有经过有效鉴权，可能产生脏数据。详细操作请参考[管理日志流](#)。

说明

支付宝小程序SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请[提交工单](#)申请。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请您根据所在区域，选择RegionName。
- 获取华为云账号的项目ID（ project id ），步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

版本更新说明

表 4-134 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	• 废弃config方法，优先使用new SDK创建一个新的实例。 • 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	• 调整日志级别等级。 • 支持labels嵌套。
1.0.15	小程序支持创建多个日志流同时上报日志到LTS。

使用说明

为确保支付宝小程序日志上报正常，需将上报地址添加到支付宝开发者平台域名列表，上报地址请参见[配置参数说明](#)。

安装 SDK

在小程序的根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。
您可以在[开源仓地址](#)下载最新的SDK。

示例代码

```
const miniSDK = require('lts-mini-sdk').default;
// import miniSDK from 'lts-mini-sdk';
App({
  onLaunch(options) {
    // 初始化
    const mymini = new miniSDK({
      // 上报region
      region: string,
      // 华为云项目ID
      projectId: string,
      // 上报地址
      url: string,
      // LTS日志组ID
      groupId: string,
      // LTS日志流ID
    })
  }
})
```

```
streamId: string,
// 调试日志等级
debug: string,
// 当前小程序所属平台
platform: 'my',
// 上报条数阈值
cacheThreshold: number,
// 上报时间阈值
timeInterval: number,
});
// 立即上报单条带标签
mymini.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
// 立即上报单条 不带标签
mymini.reportImmediately([{ key: 'value', number: 1, array: [], json: { json: 'json' } }], { 'key':
'value' });
// 缓存上报多条 带标签
mymini.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value' });
// 缓存上报多条 不带标签
mymini.report([{ 'name': 'xiaohong', 'age': 18 }, { key: 'value', number: 1, array: [], json: { json:
'json' } }]);
// 缓存上报多条 带多个标签（最多50个）
mymini.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], { 'version': '1.0.0', 'render': 'mini', 'link': '/', from:
'zhifubao' });
}
```

配置参数说明

- 配置参数说明

表 4-135 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。

字段	类型	是否必填	默认值	描述
url	string	选填	-	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。该域名地址需要在支付宝小程序开发设置服务器域名处（具体操作界面以小程序官网为准）注册为合法域名，128个字符上限。 - 使用SDK在跨云或本地上报日志时，端口使用443。 - 使用华为云主机在华为云内网上报日志时，端口使用8102。
debug	boolean	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 $30 \leq \text{cacheThreshold} \leq 1000$ 。
timeInterval	number	选填	3	默认3s 上报阈值时间，当达到阈值时间上报缓存中的数据 $1 \leq \text{timeInterval} \leq 1800$ 。
platform	string	选填	baidu	支持上报的平台。目前支持以下平台：baidu：百度小程序，wx：微信小程序，dd：钉钉小程序，my：支付宝小程序，quick：快应用。

- 日志上报report方法参数说明

表 4-136 report 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	必填	-	需要上报的日志对象或者对象数组，每个对象（Object）最多允许300个键值对，且转成JSON字符串最大支持长度1024*30个字符，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

表 4-137 reportImmediately 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	选填	-	立即上报的日志对象或者对象数组，每个对象（Object）最多允许300个键值对，且转成JSON字符串最大支持长度1024*30个字符，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的日志。 用户的日志字段名称不允许包含双下划线（__）。

字段	类型	是否必填	默认值	描述
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

表 4-138 区域表

区域名称	区域
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击“日志组名称”可查看日志流列表，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.16 云日志服务快应用 SDK

云日志服务快应用SDK提供了快应用上报日志的一系列方法，如果您需要收集和分析用户在网站上的信息，例如用户的浏览器、浏览行为记录、终端设备记录、设备异常记录、网络使用记录等，可以直接使用快应用SDK上报日志到LTS。

使用快应用SDK上报日志到LTS的场景下，需要开启日志流的匿名写入功能，开启后上报日志没有经过有效鉴权，可能产生脏数据。详细操作请参考[管理日志流](#)。

说明

快应用SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请[提交工单](#)申请。

传输协议

HTTPS

使用前提

- 参考[注册华为账号并开通华为云](#)中操作，完成注册。
- 确认云日志服务的区域，请您根据所在区域，选择RegionName。
- 获取华为云账号的项目ID（project id），步骤请参见[API凭证](#)。
- 获取需要上报到LTS的[日志组ID](#)和[日志流ID](#)。

版本更新说明

表 4-139 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	• 废弃config方法，优先使用new SDK创建一个新的实例。 • 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	• 调整日志级别等级。 • 支持labels嵌套。
1.0.15	小程序支持创建多个日志流同时上报日志到LTS。

安装 SDK

在快应用的根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。

您可以在[开源仓地址](#)下载最新的SDK。

示例代码

```
const miniSDK = require('lts-mini-sdk/quick-app').default;
// import miniSDK from 'lts-mini-sdk/quick-app';
App({
  onLaunch(options) {
    // 初始化
    const quickAppSdk = new miniSDK({
      // 上报region
      region: string,
      // 华为云项目ID
      projectId: string,
      // 上报地址
      url: string,
      // LTS日志组ID
      groupId: string,
      // LTS日志流ID
      streamId: string,
      // 日志所属组
      group: string,
      // 调试日志等级
      debug: string,
      // 当前小程序所属平台
      platform: string,
```



```
// 上报条数阈值
cacheThreshold: number,
// 上报时间阈值
timeInterval: number,
});
// 立即上报单条带标签
quickAppSdk.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
// 立即上报单条 不带标签
quickAppSdk.reportImmediately([{ key: 'value', number: 1, array: [], json: { json: 'json' } },
{ 'key': 'value' }]);
// 缓存上报多条 带标签
quickAppSdk.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value' });
// 缓存上报多条 不带标签
quickAppSdk.report([{ 'name': 'xiaohong', 'age': 18 }, { key: 'value', number: 1, array: [], json:
{ json: 'json' } }]);
// 缓存上报多条 带多个标签（最多50个）
quickAppSdk.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], { 'version': '1.0.0', 'render': 'mini', 'link': '/',
from: 'baidu' });
}
});
```

配置参数说明

- 配置参数说明

表 4-140 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。
url	string	选填	-	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。 - 使用SDK在跨云或本地上报日志时，端口使用443。 - 使用华为云主机在华为云内网上报日志时，端口使用8102。

字段	类型	是否必填	默认值	描述
debug	string	选填	OFF	控制台调试信息的输出等级，有 OFF、ERROR、WARN、INFO、DEBUG 五个等级，填入无效值会默认为 OFF，最高等级为 DEBUG，层级往下递减。值为 true 时开启 DEBUG 等级的日志，值为 false 时日志等级为 OFF。
cacheThreshold	int	选填	30	默认 30 条上报条数阈值，当缓存到达阈值时上报缓存中的数据 $30 \leq \text{cacheThreshold} \leq 1000$ 。
timeInterval	number	选填	3	默认 3s 上报阈值时间，当达到阈值时间上报缓存中的数据 $1 \leq \text{timeInterval} \leq 1800$ 。
platform	string	选填	baidu	支持上报的平台。目前支持以下平台：baidu：百度小程序，wx：微信小程序，dd：钉钉小程序，my：支付宝小程序，quick：快应用。
group	string	选填	-	日志组 ID。

• 日志上报 report 方法参数说明

表 4-141 report 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	必填	-	需要上报的日志对象或者对象数组，每个对象（Object）最多允许 300 个键值对，且转成 JSON 字符串最大支持长度 1024*30 个字符，超出部分将被截断。 array<Object>：数组中的每条 Object 数据最多 300 个键值对，转成 JSON 字符串最大支持长度 1024*30 个字节，超出部分将被截断。 用户的日志字段名称不允许包含双下划线（__）。

字段	类型	是否必填	默认值	描述
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

表 4-142 reportImmediately 方法参数说明

字段	类型	是否必填	默认值	描述
content	Object Array<Object>	选填	-	立即上报的日志对象或者对象数组，每个对象（Object）最多允许300个键值对，且转成JSON字符串最大支持长度1024*30个字符，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的数据。不填入内容时，将立即上报缓存中的日志。 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

表 4-143 区域表

区域名称	区域
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.7.17 LTS SDK 接入错误码

当您使用LTS SDK报错时，请参见[表4-144](#)进行处理。

表 4-144 错误码

状态码	错误码	错误信息	描述	处理措施
参数类错误 (LTS.00XX)	LTS.0001	%@ is null.	参数%@为空。	请设置非空的参数后重试。
	LTS.0002	%@ is invalid.	参数%@的类型、格式无效。	请设置有效的参数后重试。
	LTS.0003	The length of %@ exceeds the maximum value of %@.	参数%@的长度超过了最大值%@。	请修改参数长度后重试。
	LTS.0004	The value of %@ must be between %@ and %@.	参数%@的值必须在%@-%@之间。	请修改参数值后重试。
	LTS.0006	%@ doesn't match pattern.	参数%@不匹配正则表达式。	请设置符合正则表达式的参数后重试。
	LTS.0007	Invalid configuration parameters.	无效的配置参数。	请设置正确的配置项参数后重试。
	LTS.0009	Unsupported region.	上报日志不支持该地区。	请使用下列有效区域：cn-north-4。
集成类错误 (LTS.01XX)	LTS.0100	The LTSSDK is not initialized.	SDK未初始化。	请设置符合规则的配置参数，正确初始化SDK。

状态码	错误码	错误信息	描述	处理措施
	LTS.0102	The LTSSDK requires %@ or higher.	SDK要求%@及以上操作系统。	请根据提示调整最低操作系统版本。
	LTS.0103	Open database failed, code= %@.	打开数据库失败。	请根据sqlite3 error code查询具体情况。
	LTS.0104	Create database table failed, code= %@, error= %@.	创建数据表失败。	请根据sqlite3 error code查询具体情况。
	LTS.0105	Insert into database failed, code= %@, error= %@.	插入数据失败。	请根据sqlite3 error code查询具体情况。
	LTS.0106	Delete from database failed, code= %@, error= %@.	删除数据失败。	请根据sqlite3 error code查询具体情况。
	LTS.0107	Select from database failed, code= %@, error= %@.	查询数据失败。	请根据sqlite3 error code查询具体情况。
	LTS.0108	Close database failed, code= %@.	关闭数据库失败。	请根据sqlite3 error code查询具体情况。
	LTS.0109	Reduce database size failed, code= %@, error= %@.	减小数据库大小失败。	请根据sqlite3 error code查询具体情况。
	LTS.0110	Begin transaction failed, code= %@, error= %@.	开启事务失败。	请根据sqlite3 error code查询具体情况。
	LTS.0111	Commit transaction failed, code= %@, error= %@.	提交事务失败。	请根据sqlite3 error code查询具体情况。

状态码	错误码	错误信息	描述	处理措施
	LTS.0112	The count of database records exceeds the maximum value of %@, will delete %@ records before inserting.	数据库达到最大日志条数限制。	将自动删除适量老日志，再插入新日志。
	LTS.0113	Add column(%@) to database failed, code=%d, error=%s.	添加数据库表字段失败。	请根据sqlite3 error code查询具体情况。
	LTS.0114	Update column(%@) to database failed, code=%d, error=%s.	更新数据库表字段失败。	请根据sqlite3 error code查询具体情况。
系统类错误 (LTS.0 2XX)	LTS.0201	Request signature failed.	网络请求签名失败。	请检查region和设备时间（时区）是否正确。
	LTS.0202	Write to file failed.	写入文件失败。	请检查文件写权限。
	LTS.0203	No file read and write permissions.	无文件读写权限。	请授权文件/目录读写权限。
网络类错误 (LTS.0 3XX)	LTS.0300	Request failed, response code=%@, error=%@.	请求发送失败。	请参考API参考中的 错误码 。

4.8 跨 IAM 账号接入 LTS

当您选择了跨账号接入-日志流映射方式时，通过创建委托，您可以将委托账号的日志流映射到被委托方账号的日志流下，被委托账号即当前云日志服务登录账号。

跨账号接入成功后，若委托账号在IAM中删除了委托关系，云日志服务将无法感知到该委托被删除，配置的跨账号接入依然生效；如果您不再使用跨账号接入功能，请直接通知被委托账号删除接入配置。

前提条件

已创建委托关系。

被委托账号必须要授权Agent Operator系统角色，用于被委托账号在使用LTS委托功能时进行委托校验。授权时Agent Operator选择授权范围方案为“所有资源”，授权成功后，在授权记录查看Agent Operator的“项目[所属区域]”显示为“所有资源 [包含未来新增项目]”，如果不是，请参考[创建用户组并授权](#)重新授权。

限制条件

数据未同步完成前，目标日志流数据与源日志流可能会有一定偏差。建议1小时后，查看接入数据。

设置跨账号接入

日志服务接入方式选择跨账号接入时，按照如下操作完成接入配置。

- 步骤1 登录[云日志服务控制台](#)。
- 步骤2 在左侧导航栏中，选择“日志接入 > 接入中心”，单击“跨账号接入 - 日志流映射”进行跨账号接入配置。

或在左侧导航栏中，选择“日志接入 > 接入管理”，单击“创建”，在弹出的页面中，选择“跨账号接入 - 日志流映射”进行跨账号接入配置。
- 步骤3 选择委托。

配置相关参数，请参见[表4-145](#)，完成后，单击“下一步：日志流映射”。
- 委托账号和被委托账号需是不同租户。

表 4-145 委托参数配置

参数	说明
委托名称	填写委托人在IAM中创建的委托名称。委托人账号可通过 创建委托 将资源管理权限委托给其他华为云账号。
委托人账号名称	创建委托时的账号名称，用来验证委托关系。

- 步骤4 日志流映射。
- 在日志流映射页面，配置接入规则，有两种方式：自动配置和手动配置。
- 自动配置
 - 在日志流映射页面，单击“自动配置”。
 - 在弹出的自动配置页面中，配置相关参数信息，完成后，单击“确定”。

表 4-146 自动配置接入规则

参数	说明
规则名称前缀	填写规则名称前缀，自动配置将使用您配置的规则名称前缀，产生多条接入规则。 只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。可不填写，默认规则名称前缀为rule。

参数	说明
从委托账号中选择您希望接入的日志组/日志流	选择希望接入的日志组/日志流，最多支持选择20条。

通过自动配置的接入规则，被委托方中的目标日志组、目标日志流名称默认同委托方中源日志组、源日志流名称保持一致，也支持手动修改。

- c. 单击“预览”，查看预览结果。
- i. 预览结果有两种：
- **将创建新的目标日志流**：被委托方中新建的目标日志组/日志流。
 - **接入已存在的目标日志流**：被委托方中已存在的目标日志组/日志流。
- ii. 预览报错情况如下：
- 源日志流xxx，已配置为目标日志流。
 - 目标日志流xxx，已配置为源日志流。
 - 目标日志流xxx，已存在于其它日志组。
 - 目标日志流xxx，存在于不同目标日志组。
 - 规则名称重复。
 - 源日志流xxx，已存在映射关系。
 - 日志组数量已达上限，请选择存在的日志组进行接入。
- 当提示以上报错时，须删除日志流对应的接入规则。
- d. 预览完成后，单击“提交”。

● **手动配置**

- a. 在日志流映射页面，单击“添加规则”，参考[表4-147](#)设置规则。

表 4-147 参数说明

参数		说明
规则名称		默认为rule_xxx，也可根据您的需要进行自主命名。 只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。
委托方	源日志组	委托方的日志组，在原有的日志组中进行选择。
	源日志流	委托方的日志流，在原有的日志流中进行选择。
被委托方	目标日志组	被委托方的日志组，可在原有的日志组中进行选择或直接输入名称进行新建日志组。
	目标日志流	被委托方的日志流，可在原有的日志流中进行选择或直接输入名称进行新建日志流。

- b. 单击“预览”，查看预览结果。
 - i. 预览结果有两种：
 - **将创建新的目标日志流**：被委托方中新建的目标日志组/日志流。
 - **接入已存在的目标日志流**：被委托方中已存在的目标日志组/日志流。
 - ii. 预览报错情况如下：
 - 源日志流xxx，已配置为目标日志流。
 - 目标日志流xxx，已配置为源日志流。
 - 目标日志流xxx，已存在于其它日志组。
 - 目标日志流xxx，存在于不同目标日志组。
 - 规则名称重复。
 - 源日志流xxx，已存在映射关系。
 - 日志组数量已达上限，请选择存在的日志组进行接入。当提示以上报错时，须删除日志流对应的接入规则。
- c. 预览完成后，单击“提交”，等待创建日志接入成功。

步骤5 配置接入完成。

配置完成后，数据将会在1小时内完成同步，请您耐心等待。

- 当接入多个日志流时，单击“返回接入配置列表”可查看日志接入列表。
- 当接入单个日志流时，单击“返回接入配置列表”可查看日志接入列表；单击“查看日志流”可查看已接入的日志详情。

----结束

4.9 使用 KAFKA 协议上报日志到 LTS

您可以通过Kafka协议上报日志到日志服务，目前支持各类Kafka Producer SDK或采集工具，仅依赖于Kafka协议。支持以下场景：

- **场景1**：已有基于开源采集的自建系统，仅修改配置文件便可以将日志上报到LTS，例如Logstash。
- **场景2**：希望通过 Kafka producer SDK 来采集日志并上报，不必再安装采集ICAgent。

说明

目前此功能仅支持华北-北京四、华北-北京一、华南-广州、华东-上海一、华东-上海二、亚太-新加坡、中国-香港、华北-乌兰察布一，其他局点需要[提交工单](#)申请使用。

前提条件

- 使用云日志SDK前，您需要注册用户账号，并开通云日志服务。
- 确认云日志服务的区域，请用户根据所在区域，获取regionid。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（project id），详情请参见“我的凭证 > [API凭证](#)”。

- 获取需要上报到LTS的日志组ID、日志流ID。
- 当前仅支持内网上报，需要在ECS主机上使用。

相关限制

- 当前仅支持内网上报，端口固定为9095，IP根据所在局点进行配置。
- 支持 Kafka 协议版本为：1.0.X, 2.X.X, 3.X.X。
- 支持压缩方式：gzip, snappy, lz4。
- KAFKA认证方式为 SASL_PLAINTEXT 认证。
- KAFKA协议的ACKS参数支持设置为0、1、all，默认设置为1。

配置方式

- 使用Kafka协议上报日志时，需要使用到的通用参数如下。

表 4-148 通用参数

参数名称	描述	类型
projectId	用户账号的项目ID（ project id ）	String
logGroupId	LTS的日志组ID	String
logStreamId	LTS的日志流ID	String
regionName	云日志服务的区域	String
accessKey	用户账号的AK	String
accessSecret	用户账号的SK	String

- 使用Kafka协议上报日志时，需要配置以下参数。

表 4-149 配置参数

参数名称	说明
连接类型	当前支持SASL_PLAINTEXT
hosts	Kafka的IP和PORT地址，格式为lts-kafka.\${regionName}.\${external_global_domain_name}:9095或lts-access.\${regionName}.\${external_global_domain_name}:9095 其中IP根据局点进行配置，PORT固定为9095。详细请参考 参数获取方式 ，例如北京四局点对应hosts为 lts-kafka.cn-north-4.myhuaweicloud.com:9095。
topic	Kafka的topic名称，格式为 \${日志组ID}_\${日志流ID}，即LTS的日志组ID和日志流ID通过下划线连接，作为topic的名称。 • 若日志格式为JSON，可以将topic名称设置为\${日志组ID}_\${日志流ID}_json，即可实现JSON日志自动展开。 • 若需要指定日志的采集时间，可以将topic名称设置为\${日志组ID}_\${日志流ID}_format，确保上报符合要求的规范化日志。

参数名称	说明
username	Kafka访问用户名，配置为用户账号的项目ID。
password	Kafka访问密码，格式为\${accessKey}#\${accessSecret}，即用户账号的AK和SK通过#连接，作为Kafka的访问密码。

• JSON日志自动展开

若日志格式均为JSON，推荐使用JSON日志自动展开功能，将topic名称设置为\${日志组ID}_\${日志流ID}_json_\${index}，系统会自动解析JSON类型的日志。

默认JSON解析层数为1层，topic名称中index的值代表解析层数，index取值范围为 [1 4]，即最大解析层数为4层。

若日志非JSON类型导致解析失败，会将日志内容上报至_content_parse_fail_字段。

• 指定日志采集时间

若需要指定日志采集时间，将topic名称设置为\${日志组ID}_\${日志流ID}_format，上报日志格式需要为JSON结构且包含以下字段，请参考表4-150。

若需要指定日志采集时间，同时需要开启JSON日志自动展开功能，Kafka的topic名称格式需要为\${日志组ID}_\${日志流ID}_format_json_\${index}。

表 4-150 日志参数

参数名称	是否必选	参数类型	描述
content	是	String	日志内容。
log_time_ns	是	Long	日志数据采集时间，UTC时间（纳秒级时间戳）。 采集时间需在实际当前时间的前后1天范围内，否则指定的采集时间失效，采用当前时间作为采集时间。 采集时间需在日志存储时间范围之内，否则上报日志会被删除。比如日志组的日志存储时间是7天，则此参数不应早于当前时间的7天前。
labels	是	Object	用户自定义label。 请不要将字段名称设置为 内置保留字段 ，否则可能会造成字段名称重复、查询不精确等问题。

日志示例

```
{
  "log_time_ns": "XXXXXXXXXXXXXXXXXXXX",
  "content": "This is a log 1"
  "labels": {
    "type": "kafka"
  }
}
```

调用示例

1. Beat系列软件调用（FileBeat等）。以FileBeat为例，配置参数如下：

```
output.kafka:
hosts: ["${ip}:${port}"]
partition.round_robin:
reachable_only: false
username: "${projectId}"
password: "${accessKey}#${accessSecret}"
topic: "${logGroupId}_${logStreamId}"
sasl.mechanism: "PLAIN"
security.protocol: "SASL_PLAINTEXT"
acks: "0"
compression: gzip
```

2. 通过Logstash软件上报日志。

```
input {
  stdin {}
}
output {
  kafka {
    # 配置地址
    bootstrap_servers => "${ip}:${port}"
    # 配置topic
    topic_id => "${logGroupId}_${logStreamId}"
    # 配置消息确认机制
    acks => "0"
    # 配置压缩方式
    compression_type => "gzip"
    # 配置认证方式
    security_protocol => "SASL_PLAINTEXT"
    sasl_mechanism => "PLAIN"
    # 用户名 projectId 密码 accessKey#accessSecret
    sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginModule required username='${projectId}' password='${accessKey}#${accessSecret}';"
  }
}
```

3. 通过Flume软件上报日志。

```
#Name
a1.sources = r1
a1.channels = c1
a1.sinks = k1
#Source
a1.sources.r1.type = TAILDIR
a1.sources.r1.channels = c1
a1.sources.r1.filegroups = f1
a1.sources.r1.filegroups.f1 = /tmp/test.txt
a1.sources.r1.fileHeader = true
a1.sources.r1.maxBatchCount = 1000
#Channel
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 100
#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";
#Bind
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

SDK 调用示例

1. Java SDK调用示例。

maven依赖（示例kafka协议版本为2.7.1）：

```
<dependencies>
  <dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>2.7.1</version>
  </dependency>
</dependencies>
```

代码示例：

```
package org.example;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerRecord;
import java.util.Properties;
public class ProducerDemo {
    public static void main(String[] args) {
        Properties props = new Properties();
        // 配置地址
        props.put("bootstrap.servers", "${ip}:${port}");
        // 配置消息确认机制
        props.put("acks", "0");
        // 配置认证方式
        props.put("security.protocol", "SASL_PLAINTEXT");
        props.put("sasl.mechanism", "PLAIN");
        // 用户名 projectId 密码 accessKey#accessSecret
        props.put("sasl.jaas.config",
            "org.apache.kafka.common.security.plain.PlainLoginModule required username='${projectId}'
            password='${accessKey}#${accessSecret}';");
        // 配置压缩方式
        props.put("compression.type", "${compress_type}");
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        // 1.创建一个生产者对象
        Producer<String, String> producer = new KafkaProducer<>(props);
        // 2.调用send方法
        for (int i = 0; i < 1; i++) {
            ProducerRecord record = new ProducerRecord<>("${logGroupId}_${logStreamId}", "${message}");
            // 配置recordHeader
            // record.headers().add(new RecordHeader("LTS_LOG_TYPE","FORMAT".getBytes()));
            producer.send(record);
        }
        // 3.关闭生产者
        producer.close();
    }
}
```

2. Python SDK调用示例。

```
from kafka import KafkaProducer
producer = KafkaProducer(
    # 配置地址
    bootstrap_servers="${ip}:${port}",
    # 配置消息确认机制
    acks="0",
    # 配置压缩方式
    compression_type="${compression_type}"
    # 配置认证方式
    sasl_mechanism="PLAIN",
    security_protocol="SASL_PLAINTEXT",
    # 用户名 projectId 密码 accessKey#accessSecret
    sasl_plain_username="${projectId}",
    sasl_plain_password="${accessKey}#${accessSecret}"
)
print('start producer')
for i in range(0, 3):
```

```
data = bytes("${message}", encoding="utf-8")
future = producer.send("${logGroupId}_${logStreamId}", data)
result = future.get(timeout=10)
print(result)
print('end producer')
```

报错说明

当参数错误或不匹配时，会有相应的报错提示。

表 4-151 报错说明

报错信息	报错原因
TopicAuthorizationException	projectId（项目ID）、accessKey（AK）、accessSecret（SK）参数错误或者不匹配。
UnknownTopicOrPartitionException	logGroupId（日志组ID）、logStreamId（日志流ID）参数错误或者不匹配。
InvalidRecordException	日志格式错误或者日志中的projectId（项目ID）、logGroupId（日志组ID）、logStreamId（日志流ID）与外部设置参数不一致。

参数获取方式

表 4-152 区域表

区域名称	终端节点
华北-北京四	lts-kafka.cn-north-4.myhuaweicloud.com
华北-北京一	lts-access.cn-north-1.myhuaweicloud.com
华东-上海一	lts-access.cn-east-3.myhuaweicloud.com
华东-上海二	lts-kafka.cn-east-2.myhuaweicloud.com
华南-广州	lts-access.cn-south-1.myhuaweicloud.com
亚太-新加坡	lts-kafka.ap-southeast-3.myhuaweicloud.com
中国-香港	lts-access.ap-southeast-1.myhuaweicloud.com
华北-乌兰察布一	lts-access.cn-north-9.myhuaweicloud.com

4.10 使用 Flume 采集器上报日志到 LTS

Flume是一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统，Flume支持在日志系统中定制各类数据发送方，用于收集数据；同时，Flume提供对数据进行简单处理，并写到各种数据接受方的能力。

用户使用Flume系统采集日志，并且通过LTS侧提供的KAFKA协议方式上报日志。以下是部分常用数据采集场景示例：

1. [使用Flume采集文本日志上报到LTS](#)
2. [使用Flume采集数据库表数据并且上报至LTS](#)
3. [使用Flume采集syslog协议传输的日志上报到LTS](#)
4. [通过Flume采集TCP/UDP协议传输的日志上报到LTS](#)
5. [通过Flume采集SNMP协议上报的设备管理数据并发送到LTS](#)
6. [使用默认拦截器处理日志](#)
7. [自定义拦截器处理日志](#)
8. [使用外部数据源丰富日志内容并上报到LTS](#)

前提条件

- 用户机器已经安装了JDK。
- 用户已经安装Flume，并且需要在Flume中配置文件中配置JDK路径。

使用 Flume 采集文本日志上报到 LTS

支持使用Flume采集文本日志内容上报至LTS，参考如下示例添加采集文本日志的conf文件。以下示例中的参数介绍请参考[使用KAFKA协议上报日志](#)。

```
#Named
a1.sources = r1
a1.channels = c1
a1.sinks = k1

#Source
a1.sources.r1.type = TAILDIR
a1.sources.r1.channels = c1
a1.sources.r1.filegroups = f1
a1.sources.r1.filegroups.f1 = /tmp/test.txt
a1.sources.r1.fileHeader = true
a1.sources.r1.maxBatchCount = 1000

#Channel
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 100

#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";

#Bind
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

使用 Flume 采集数据库表数据并且上报至 LTS

使用Flume采集数据库表数据并且上报至LTS，实现对表数据变动监控。以下示例中的参数介绍请参考[使用KAFKA协议上报日志](#)。

步骤1 在<https://github.com/keedio/flume-ng-sql-source>页面下载flume-ng-sql-source插件，转换为jar包并取名为flume-ng-sql-source.jar，打包前注意将pom文件中的flume-ng-core 版本与flume安装版本保持一致，并且将jar包放在安装Flume包路径的lib目录下面，例如FLUME_HOME/lib目录下（例子中的FLUME_HOME为Flume安装路径，仅供参考，请以实际安装路径为准）。

步骤2 添加MySQL驱动到FLUME_HOME/lib目录下：

1. 下载MySQL驱动。
`wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.35.tar.gz`
2. 将驱动包解压并打为jar包。
`tar xzf mysql-connector-java-5.1.35.tar.gz`
3. 将jar包存放在FLUME_HOME/lib/路径。
`cp mysql-connector-java-5.1.35-bin.jar FLUME_HOME/lib/`

步骤3 添加采集MySQL的conf文件。

```
# a1表示agent的名称
# source是a1的输入源
# channels是缓冲区
# sinks是a1输出目的地，本例子sinks使用了kafka
a1.channels = c1
a1.sources = r1
a1.sinks = k1

#source
a1.sources.r1.type = org.keedio.flume.source.SQLSource
# 连接mysql的一系列操作，{mysql_host}改为您虚拟机的ip地址，可以通过ifconfig或者ip addr查看，
{database_name}改为数据库名称
# url中要加入?useUnicode=true&characterEncoding=utf-8&useSSL=false，否则有可能连接失败
a1.sources.r1.hibernate.connection.url = jdbc:mysql://{mysql_host}:3306/{database_name}?
useUnicode=true&characterEncoding=utf-8&useSSL=false
# Hibernate Database connection properties
# mysql账号，一般都是root
a1.sources.r1.hibernate.connection.user = root
# 填入您的mysql密码
a1.sources.r1.hibernate.connection.password = xxxxxxxx
a1.sources.r1.hibernate.connection.autocommit = true
# mysql驱动
a1.sources.r1.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
a1.sources.r1.hibernate.connection.driver_class = com.mysql.jdbc.Driver
# 存放status文件
a1.sources.r1.status.file.path = FLUME_HOME/bin
a1.sources.r1.status.file.name = sqlSource.status
# Custom query
# 填写需要采集的数据表名{table_name}，也可以使用下面的方法：
a1.sources.r1.custom.query = select * from {table_name}

#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";

a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 10000
a1.channels.c1.byteCapacityBufferPercentage = 20
a1.channels.c1.byteCapacity = 800000
```


步骤4 启动Flume后，即可开始采集数据库中的表数据到LTS。

----结束

使用 Flume 采集 syslog 协议传输的日志上报到 LTS

Syslog协议是一种用于在IP网络中传输日志消息的协议，通过Flume将syslog协议传输的日志采集并上报到LTS。以下示例中的参数介绍请参考[使用KAFKA协议上报日志](#)。

- 接收UDP日志，参考如下示例添加采集Syslog协议的conf文件。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type=syslogudp
#host_port为syslog服务器的端口
a1.sources.r1.port = {host_port}
#host_ip为syslog服务器的ip地址
a1.sources.r1.host = {host_ip}
a1.sources.r1.channels = c1

a1.channels.c1.type = memory

#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";
a1.sinks.k1.channel = c1
```

- 接收TCP日志，参考如下示例添加采集Syslog协议的conf文件。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type=syslogtcp
#host_port为syslog服务器的端口
a1.sources.r1.port = {host_port}
#host_ip为syslog服务器的ip地址
a1.sources.r1.host = {host_ip}
a1.sources.r1.channels = c1

a1.channels.c1.type = memory

#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";
a1.sinks.k1.channel = c1
```

通过 Flume 采集 TCP/UDP 协议传输的日志上报到 LTS

通过Flume采集TCP/UDP协议传输的日志上报到LTS。以下示例中的参数介绍请参考[使用KAFKA协议上报日志](#)。

- 采集TCP端口日志，参考如下示例添加采集端口的conf文件。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type = netcat
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = {host_port}

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";

a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

- 采集UDP端口日志，参考如下示例添加采集端口的conf文件。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type = netcatudp
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = {host_port}

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";

a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

通过 Flume 采集 SNMP 协议上报的设备管理数据并发送到 LTS

通过Flume采集SNMP协议上报的设备管理数据并发送到LTS。以下示例中的参数介绍请参考[使用KAFKA协议上报日志](#)。

- 监听SNMP协议通信端口号161。参考如下示例添加SNMP协议接受日志的conf。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type = netcatudp
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = 161

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
```

```
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";

a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

- 监听SNMP协议陷阱(Trap)通信的端口号162，参考如下示例添加SNMP协议接受日志的conf。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type = netcatudp
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = 162

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";

a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

使用默认拦截器处理日志

使用Flume采集器时，拦截器是简单的插件式组件，设置在Source和Channel之间。Source接收到的事件Event，在写入Channel之前，拦截器都可以进行转换或者删除这些事件。每个拦截器只处理同一个Source接收到的事件。

- 时间戳拦截器

该拦截器的作用是将时间戳插入到flume的事件报头中。如果不使用任何拦截器，flume接收到的只有message。时间戳拦截器的配置，参数默认值描述type，类型名称timestamp，也可以使用类名的全路径preserveExisting为false。如果设置为true，若事件中报头已经存在，不会替换时间戳报头的值。source连接到时间戳拦截器的配置：

```
a1.sources.r1.interceptors = timestamp
a1.sources.r1.interceptors.timestamp.type=timestamp
a1.sources.r1.interceptors.timestamp.preserveExisting=false
```

- 正则过滤拦截器

在日志采集的时候，可能有一些数据是不需要的，添加过滤拦截器可以过滤掉不需要的日志，也可以根据需要收集满足正则条件的日志。参数默认值描述type，类型名称REGEX_FILTER。excludeEvents为false时默认收集匹配到的事件。如果

为true，则会删除匹配到的event，收集未匹配到的。source连接到正则过滤拦截器的配置：

```
a1.sources.r1.interceptors = regex
a1.sources.r1.interceptors.regex.type=REGEX_FILTER
a1.sources.r1.interceptors.regex.regex=(today)|(Monday)
a1.sources.r1.interceptors.regex.excludeEvents=false
```

这样配置的拦截器就只会接收日志消息中带有today或者Monday的日志。

- 搜索并替换拦截器

拦截器基于Java正则表达式提供简单的基于字符串的搜索和替换功能。配置如下：

```
# 拦截器别名
a1.sources.r1.interceptors = search-replace
# 拦截器类型，必须是search_replace
a1.sources.r1.interceptors.search-replace.type = search_replace

# 删除事件正文中的字符，根据正则匹配event内容
a1.sources.r1.interceptors.search-replace.searchPattern = today
# 替换匹配到的event内容
a1.sources.r1.interceptors.search-replace.replaceString = yesterday
# 设置字符集，默认是utf8
a1.sources.r1.interceptors.search-replace.charset = utf8
```

自定义拦截器处理日志

在Flume中自定义拦截器的方式主要流程如下（以java语言为例），以下示例中的FLUME_HOME表示Flume的安装路径，例如/tools/flume（仅供参考），实际配置的时候，请以用户安装Flume的实际路径为准。

步骤1 创建MAVEN工程项目，引入Flume依赖。

根据集群中的 Flume 版本，引入 Flume 依赖，如下所示：

```
<dependencies>
  <dependency>
    <groupId>org.apache.flume</groupId>
    <artifactId>flume-ng-core</artifactId>
    <version>1.10.1</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

无需将该依赖打包进最后的JAR包中，故将其作用域设置为provided。

步骤2 创建类实现拦截器接口Interceptor，并且实现相关方法。

- initialize() 方法：初始化拦截器操作，读取配置信息、建立连接等。
- intercept(Event event) 方法：用于拦截单个事件，并对事件进行处理。接收一个事件对象作为输入，并返回一个修改后的事件对象。
- intercept(List<Event> list) 方法：事件批处理，拦截事件列表，并对事件列表进行处理。
- close() 方法：关闭拦截器，在这里释放资源、关闭连接等。

```
import org.apache.flume.Event;
import org.apache.flume.interceptor.Interceptor;

import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

public class TestInterceptor implements Interceptor {
    @Override
```

```
public void initialize() {  
    }  
    @Override  
    public Event intercept(Event event) {  
        // 获取事件数据  
        String eventData = new String(event.getBody(), StandardCharsets.UTF_8);  
        // 检查事件数据中是否包含指定字符串  
        if (eventData.contains("hello")) {  
            // 如果包含指定字符串，则过滤掉该事件，返回 null  
            return null;  
        }  
        return event;  
    }  
    @Override  
    public List<Event> intercept(List<Event> events) {  
        // 创建一个新的列表，存储处理过后的事件  
        List<Event> interceptedEvents = new ArrayList<>();  
        for (Event event : events) {  
            Event interceptedEvent = intercept(event);  
            if (interceptedEvent != null) {  
                interceptedEvents.add(interceptedEvent);  
            }  
        }  
        return interceptedEvents;  
    }  
    @Override  
    public void close() {  
    }  
}
```

步骤3 构建拦截器，拦截器的创建和配置通常是通过 Builder 模式来完成的，完整的代码如下所示：

```
import org.apache.flume.Context;  
import org.apache.flume.Event;  
import org.apache.flume.interceptor.Interceptor;  
  
import java.nio.charset.StandardCharsets;  
import java.util.ArrayList;  
import java.util.List;  
  
public class TestInterceptor implements Interceptor {  
    @Override  
    public void initialize() {  
    }  
    @Override  
    public Event intercept(Event event) {  
        // 获取事件数据  
        String eventData = new String(event.getBody(), StandardCharsets.UTF_8);  
        // 检查事件数据中是否包含指定字符串  
        if (eventData.contains("hello")) {  
            // 如果包含指定字符串，则过滤掉该事件，返回 null  
            return null;  
        }  
        return event;  
    }  
    @Override  
    public List<Event> intercept(List<Event> events) {  
        List<Event> interceptedEvents = new ArrayList<>();  
        for (Event event : events) {  
            Event interceptedEvent = intercept(event);  
            if (interceptedEvent != null) {  
                interceptedEvents.add(interceptedEvent);  
            }  
        }  
        return interceptedEvents;  
    }  
}
```

```
    }  
    }  
    return interceptedEvents;  
}  
  
@Override  
public void close() {  
  
}  
  
// 拦截器构建  
public static class Builder implements Interceptor.Builder {  
  
    @Override  
    public void configure(Context context) {  
  
    }  
  
    @Override  
    public Interceptor build() {  
        return new TestInterceptor();  
    }  
  
}  
}
```

步骤4 转换为jar包，并且将其上传至Flume安装路径下的lib文件夹下（请以用户安装Flume的实际路径为准）。

步骤5 编写配置文件，需要将自定义的拦截器配置进去。

拦截器全类名配置时需要注意，格式为拦截器的全类名 + \$Builder。

```
# 拦截器配置  
# 拦截器定义  
a1.sources.r1.interceptors = i1  
# 拦截器全类名  
a1.sources.r1.interceptors.i1.type = TestInterceptor$Builder
```

步骤6 运行Flume即可。

----结束

KV解析日志：用空格分隔字符串并且转换为Map类型字符串。

```
public class TestInterceptor implements Interceptor {  
    @Override  
    public void initialize() {  
    }  
  
    @Override  
    public Event intercept(Event event) {  
        // 获取事件数据  
        String eventData = new String(event.getBody(), StandardCharsets.UTF_8);  
        Map<String, Object> splitMap = new HashMap<>();  
        String[] splitList = eventData.split(" ");  
        for (int i = 0; i < splitList.length; i++) {  
            splitMap.put("field" + i, splitList[i].trim());  
        }  
        eventData.setBody(splitMap.toString().getBytes(StandardCharsets.UTF_8));  
        return event;  
    }  
  
    @Override  
    public List<Event> intercept(List<Event> events) {  
        List<Event> interceptedEvents = new ArrayList<>();  
        for (Event event : events) {  
            Event interceptedEvent = intercept(event);  
            if (interceptedEvent != null) {  
                interceptedEvents.add(interceptedEvent);  
            }  
        }  
        return interceptedEvents;  
    }  
}
```

```
        interceptedEvents.add(interceptedEvent);
    }
}
return interceptedEvents;
}

@Override
public void close() {
}
}
```

使用外部数据源丰富日志内容并上报到 LTS

Flume数据传输的基本单元，以Event的形式将数据从源头传输至目的地。Event由Header和Body两部分组成，Header用来存放该Event的一些属性，为K-V结构，Body用来存放该条数据，形式为字节数组。

有外部数据源时，如果您需要丰富日志内容，例如修改日志内容、添加字段、删除内容等操作，将修改内容添加至Event的body中，Flume才能将日志内容上报到LTS。例如使用Java自定义扩展日志内容。以下示例中的参数介绍请参考[使用KAFKA协议上报日志](#)。

```
import com.alibaba.fastjson.JSONObject;

import org.apache.flume.Context;
import org.apache.flume.Event;
import org.apache.flume.interceptor.Interceptor;

import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

public class TestInterceptor implements Interceptor {
    @Override
    public void initialize() {
    }

    @Override
    public Event intercept(Event event) {
        // 获取事件数据，将原数据转换为json字符串并且添加额外字段
        String eventData = new String(event.getBody(), StandardCharsets.UTF_8);
        JSONObject object = new JSONObject();
        object.put("content", eventData);
        object.put("workLoadType", "RelipcaSet");
        eventData = object.toJSONString();
        eventData.setBody(eventData.getBytes(StandardCharsets.UTF_8));
        return event;
    }

    @Override
    public List<Event> intercept(List<Event> events) {
        List<Event> interceptedEvents = new ArrayList<>();
        for (Event event : events) {
            Event interceptedEvent = intercept(event);
            if (interceptedEvent != null) {
                interceptedEvents.add(interceptedEvent);
            }
        }
        return interceptedEvents;
    }

    @Override
    public void close() {
    }
}
```

4.11 使用匿名写入采集日志

如果您需要收集和分析用户在浏览器、小程序上的信息，例如用户的浏览器、浏览行为记录、购买行为记录、停留时间，可以使用匿名写入功能。只需在日志流页面开启匿名写入开关，就可以将用户行为信息上传到云日志服务的日志流中。

说明

使用匿名写入采集日志功能仅支持华北-北京四、华东-上海一、华南-广州区域的白名单用户使用，如有需要，请[提交工单](#)，其他区域暂不支持申请开通。

背景信息

云日志服务LTS的匿名写入功能用于在浏览器、小程序中跟踪用户行为并将行为数据上传到LTS中，便于您分析用户行为等。您可以通过匿名写入采集各种浏览器、小程序中的用户信息，例如：

- 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：用户在网站上的单击行为、购买行为等信息）。
- 用户在小程序中的停留时间、是否活跃等信息。

开启匿名写入

- 步骤1 登录[云日志服务控制台](#)。
- 步骤2 单击日志组名称对应的  按钮。
- 步骤3 单击“创建日志流”，在创建日志流页面，参考[表4-153](#)填写日志流相关信息，需要开启“匿名写入”，适用于端侧SDK上报日志，打开匿名写入则表示该日志流打开匿名写入权限，不会经过有效鉴权，可能产生脏数据。

图 4-82 创建日志流

创建日志流 

日志组名称

lts

日志流名称

lts-topic-kn8n

日志流名称不能与其他日志流的名称或原始名称相同

企业项目

default

 [查看企业项目](#)

日志存储

☒

开启日志存储：日志将会被存入搜索引擎，能使用日志全量功能。关闭日志存储：日志不会落盘存储，可节约索引流量和存储费用，只能使用日志生成指标、转储功能，不能使用日志搜索分析、告警、消费加工等其他功能。 [了解更多](#)

日志存储时间(天)

☐ ?

标签

键	值	操作
+ 添加标签 您还可以添加20个标签（系统标签不占配额） 了解更多		

匿名写入

☒

匿名写入适用于安卓/iOS/小程序/浏览器端上报日志，打开匿名写入则表示该日志流打开匿名写入权限，不会经过有效鉴权，可能产生脏数据。

备注

0/1024

表 4-153 日志流参数说明

参数	说明
日志组名称	默认显示所属日志组名称，不支持修改。
日志流类型	说明 白名单功能，如果需要使用，请提交工单申请开通。 根据业务需要选择日志流类型。 - 标准型包含LTS完整功能，适用于构建完整可观测性系统。 - 搜索型仅支持关键词搜索。不支持快速分析/统计图表/日志分析/SQL告警，适用于日志搜索场景，其优点是索引流量费用是标准型的31%。更多信息请参考计费项。
日志流名称	LTS会自动生成一个默认的日志流名称，建议您根据业务自定义日志流的名称，创建成功后支持修改日志流名称。命名规范如下： - 只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。 - 长度为1-64个字符。 - 日志流名称不能重复。
企业项目	选择业务需要的企业项目，默认为default。也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。 - 企业项目需要开通后才能使用，请参考如何开通企业项目。 - 支持将该企业项目资源迁出，详细请参考迁出企业项目资源。如果创建转储任务时选择多个日志流，则这些日志流无法进行企业项目迁移。
日志存储	若关闭日志存储，则无法开启日志存储时间。 - 开启“日志存储”：日志将会被存入搜索引擎，能使用日志全量功能。 - 关闭“日志存储”：日志不会存储到LTS，可节约索引流量和存储费用，只能使用日志生成指标、转储功能，不能使用日志搜索分析、告警、消费加工等其他功能。
日志存储时间(天)	日志流的存储时间，即日志上报到LTS后日志存储的时间。 日志数据默认存储30天，可以在1~365天之间设置。 - 打开日志流的“日志存储时间”：日志的存储时间使用日志流设置的日志存储时间。 - 关闭日志流的“日志存储时间”：日志的存储时间使用日志组设置的日志存储时间。 - 超出存储时间的日志将会被自动删除，您可以按需将日志数据转储至OBS桶中进行长期存储。 说明 目前仅支持白名单用户提交工单申请开通日志存储时间为1095天，但是存量Region需要根据底层资源评估是否支持1095天。

参数	说明
智能冷存储	白名单功能，开启日志存储时间后，根据业务需要设置智能冷存储功能。详细请参考智能冷存储。 开启智能冷存储需要日志存储时间大于7天。
标准存储层数据保存时间	开启智能冷存储后，需要设置标准存储层数据的保存时间。 标准存储数据至少需保存7天才能转换为智能冷存储层数据。当日志保存时间大于标准存储层数据保存时间，且小于日志存储时间时，日志数据将自动转换为智能冷存储层数据存储。
匿名写入	匿名写入默认关闭，适用于安卓/IOS/小程序/浏览器端上报日志才能使用匿名写入。 需要提交工单开通白名单。 打开匿名写入则表示该日志流打开匿名写入权限，不会经过有效鉴权，可能产生脏数据。关于SDK使用请参考使用SDK接入LTS。 说明 使用匿名写入采集日志功能仅支持华北-北京四、华东-上海一、华南-广州区域的白名单用户使用，如有需要，请提交工单，其他区域暂不支持申请开通。
分区数量	分区数量仅对消费日志有效。每个分区支持5MB/S写流量，10MB/S读流量；当读写流量超过分区的读写能力时，LTS会自动扩容读写分区数量，您也可以手动扩容读写分区数量，来提供更高的读写能力。输入值范围1-64。 单击“查看当前分区详情”进入分区详情页面，查看分区、状态、创建时间。 说明 白名单功能，如果需要使用，请提交工单申请开通。

参数	说明
标签	按照业务需求对不同的日志流添加对应的标签，单击“添加”，分别填写标签键key和标签值value。如需添加多个标签可重复该步骤，最多支持添加20个标签。 标签键key限制条件： • 标签键可以包含任意语种的字母、数字和空格，以及_!:=+~@字符，但首尾不能包含空格，且不能以_sys_开头。 • 标签键长度不能超过128个字符。 • 标签键名称不可重复。 标签值value限制条件： • 标签值可以包含任意语种的字母、数字和空格，以及_!:=+~@字符。 • 标签值长度不能超过255个字符。 标签策略： 若您的组织已经设定云日志服务的相关标签策略，则需按照标签策略规则为日志组、日志流、日志接入、主机组添加标签。标签如果不符合标签策略的规则，则可能会导致日志组、日志流、日志接入、主机组创建失败，请联系组织管理员了解标签策略详情。标签策略详细介绍请参考标签策略概述，标签管理详细介绍请参考标签管理。 删除标签： 如需删除标签，请单击标签操作列的“删除”。 警告 删除标签无法恢复，请谨慎操作。 如果配置转储时使用了该标签，删除标签后，请同步修改转储配置信息。关于转储信息请参考日志转储至OBS。
备注	自定义填写备注信息，字符长度0-1024个字符。

步骤4 单击“确定”。

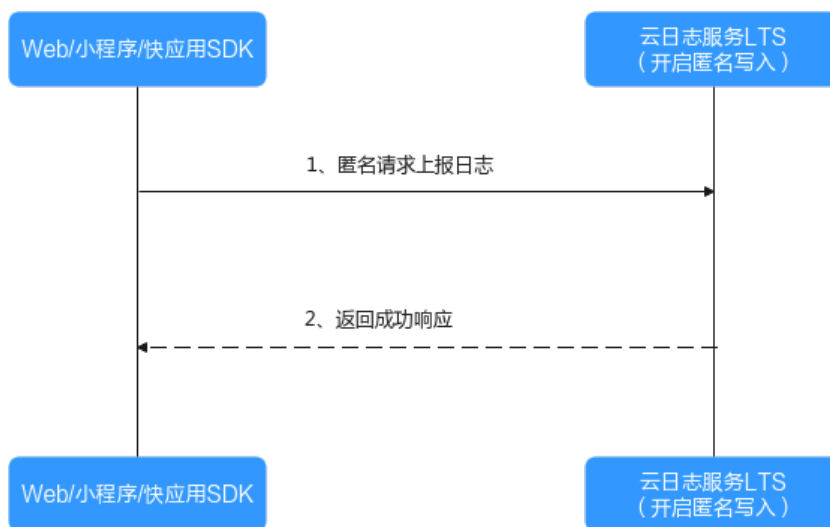
----结束

使用匿名写入上报日志

LTS支持通过客户端SDK和HTTP GET请求的方式使用匿名写入上报日志。

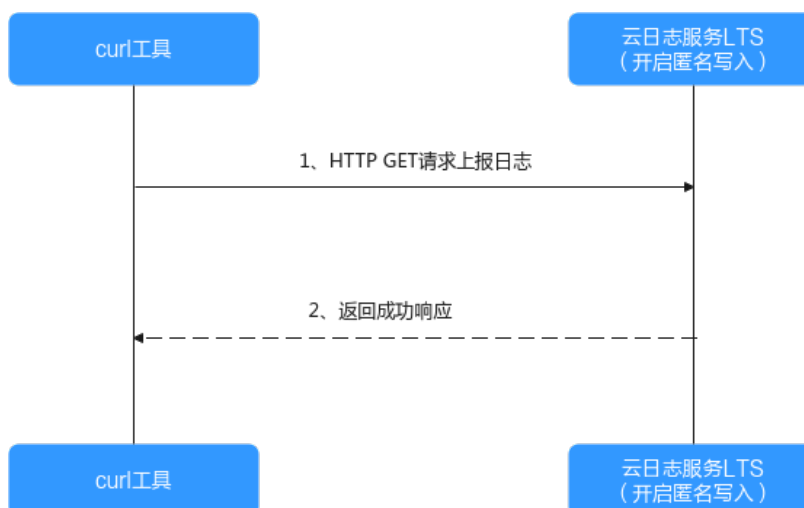
- 客户端SDK：通过Web SDK/小程序SDK/快应用SDK，将自定义类型的日志上传到云日志服务LTS，例如表单提交、页面提交、异常报告等。关于SDK上报日志的方法请参考[使用SDK接入LTS](#)。

图 4-83 客户端 SDK



- HTTP GET请求：通过在网站的前端代码中发送GET请求，将日志上传到云日志服务LTS，适合简单日志的收集。详细操作请参考[通过HTTP GET请求上传日志](#)。

图 4-84 HTTP GET 请求



通过 HTTP GET 请求上传日志

开通匿名写入后，您可以通过HTTP GET请求方法上传日志到日志流中。

HTTP GET请求，开通GET请求的白名单后，请[提交工单](#)联系技术支持工程师提供{endpoint}地址才能使用此功能。

请求路径：

```
/v3/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/struct/logs?key1=val1&key2=val2
```

GET接口总数据量不能超过16KB，超过可能会上报失败。不支持压缩。

- 配置参数说明

表 4-154 配置参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见： 获取账号ID、项目ID、日志组ID、日志流ID 。
log_group_id	是	String	日志组ID，获取方式请参见： 获取账号ID、项目ID、日志组ID、日志流ID 。
log_stream_id	是	String	日志流ID，获取方式请参见： 获取账号ID、项目ID、日志组ID、日志流ID 。
key1=val1&key2=val2	是	String	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。

- 对于上报日志中的特殊字符，请您使用url编码，否则可能出现日志上报乱码、失败。
- 日志长度小于16KB，指的是url编码后的长度。假如您的日志大小只有13KB，但是经过编码后的大小为17KB，也是超过长度限制。

● 响应参数

状态码为 200 时:

表 4-155 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。
errorMessage	String	调用失败响应信息描述。
result	String	响应结果。

● 请求示例

```
curl -k "https://lts-access.cn-north-4.myhuaweicloud.com:443/v3/xxxxxxProject_id/lts/groups/xxxxxxLog_group_id/streams/xxxxxxLog_stream_id/struct/logs?key1=val1&key2=val2"
```

支持在上报日志的公网地址域名后指定端口号，本示例中使用的端口号为443：

- 公网场景下，通过SDK跨云或本地服务器上报告志到LTS时，端口号使用443。
- 华为云内网场景下，待上报日志的华为云主机与LTS在同一Region时，端口号使用8102。

● 响应示例

状态码： 200

日志上报成功。

```
{
  "errorCode": "SVCSTG.ALS.200.200",
  "errorMessage": "Report success.",
}
```

```
"result": null
}
```

- 状态码

表 4-156 状态码

状态码	描述
200	请求响应成功。
400	非法请求。建议根据error_msg直接修改该请求，不要重试该请求。
401	在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。
403	请求被拒绝访问。返回该状态码，表明请求能够到达服务端，且服务端能够理解用户请求，但是拒绝做更多的事情，因为该请求被设置为拒绝访问，建议直接修改该请求，不要重试该请求。
500	系统内部错误。
503	被请求的服务无效，服务不可用。

4.12 自建中间件（邀测）

4.12.1 中间件-MySQL 日志

MySQL是一种广泛使用的关系型数据库管理系统。MySQL日志系统包含了多种不同类型的日志，用于记录MySQL服务器在运行时的各种活动和事件。这些日志对于数据库管理、性能调优、故障排查等方面都非常重要。以下是MySQL中常见的几种日志及其说明：

1. 错误日志（Error Log）：
 - 记录MySQL服务器启动、运行或停止时出现的问题，以及服务器运行时的错误或警告信息。
 - 对于诊断和解决问题非常有用。
 - 在MySQL安装目录下的data文件夹中，文件名通常为hostname.err，其中hostname是MySQL服务器的主机名。
2. 查询日志（General Query Log）：
 - 记录MySQL服务器接收到的所有客户端连接和执行的SQL语句。
 - 可以用于分析查询的性能和执行情况。
 - 需要手动启动，默认情况下不启用。
3. 慢查询日志（Slow Query Log）：
 - 记录执行时间超过指定阈值（long_query_time）的查询语句。
 - 可以用于找出执行时间较长的查询语句，以便进行性能优化。
 - 同样需要手动启动。
4. 二进制日志（Binary Log）：

- 记录了所有的修改数据库数据的操作（如UPDATE、INSERT、DELETE等）。
 - 主要用于数据备份、恢复和复制。
 - 可以通过配置参数log-bin来启用。
5. 中继日志（Relay Log）：
- 在MySQL复制架构中，中继日志用于从主服务器的二进制日志文件中复制事件，并在从服务器上执行这些事件。
 - 主要用于MySQL的主从复制。
6. 事务日志（Transaction Log）：
- 主要针对支持事务的存储引擎（如InnoDB）而言，记录了事务的开始、提交、回滚等操作。
 - 用于保证数据库的一致性和持久性。
7. Undo日志（Undo Log）：
- 主要与InnoDB存储引擎相关，记录了正在进行的事务对数据库的修改操作。
 - 用于回滚事务或实现多版本并发控制（MVCC）。

采集方法

使用ICAgent采集日志，接入前请参考[安装ICAgent（区域内主机）](#)完成ICAgent安装。然后再通过采集ECS-文本日志方式将日志上报至LTS服务，详细操作请参考[云主机ECS文本日志接入LTS](#)。

Mysql 日志示例以及字段含义

介绍错误日志、查询日志、慢查询日志的示例和字段说明。

- 错误日志（Error Log）
2023-03-29T14:30:15.123456Z 0 [ERROR] InnoDB: Unable to lock ./ibdata1, error: 11

表 4-157 错误日志字段说明

字段	说明
2023-03-29T14:30:15.123456Z	日志条目的时间戳，采用ISO 8601格式，包含时区信息（Z表示UTC时间）。
0	线程ID，对于错误日志，这通常是0，表示这是一个全局消息，而不是与特定客户端连接相关。
[ERROR]	日志级别，这里表示这是一个错误消息。
InnoDB: Unable to lock ./ibdata1, error: 11	具体的错误消息，指示InnoDB存储引擎无法锁定ibdata1文件，并给出了错误代码11（通常表示“资源暂时不可用”）。

- 查询日志（General Query Log）
190109 14:23:00 3 Connect root@localhost on testdb
3 Query SELECT * FROM users WHERE id = 1

表 4-158 查询日志字段说明

字段	说明
190109 14:23:00	日志条目的时间戳，采用YYMMDD HH:MM:SS 格式。
3	线程ID，表示这个操作是由线程ID为3的客户端连接执行的。
Connect	操作类型，这里表示一个客户端连接。
root@localhost	连接的用户名和主机名。
on testdb	连接使用的数据库名。
Query	操作类型，这里表示执行了一个SQL查询。
SELECT * FROM users WHERE id = 1	执行的SQL语句。

- 慢查询日志（ Slow Query Log ）
190109 14:23:00 3 Connect root@localhost on testdb
 3 Query SELECT * FROM users WHERE id = 1

表 4-159 慢查询字段说明

字段	说明	示例
Time	日志条目的时间戳。	2023-03-29T15:00:01.234567Z
User@Host	执行查询的用户和主机信息。	root[root] @ localhost [127.0.0.1]
Id	线程ID。	3
Query_time	查询执行的时间长度（ 秒 ）。	5.000234
Lock_time	查询等待表锁的时间长度（ 秒 ）。	0.000123
Rows_sent	查询返回的行数。	1000
Rows_examined	查询检查的行数（ 即扫描的行数，可能包括未返回的行 ）。	100000
SET timestamp=	设置当前会话的时间戳（ 这通常是为了复制日志的一致性 ）。	1617012001
SELECT ...	执行的SQL查询。	SELECT * FROM big_table WHERE some_column = 'value'

日志位置

1. 错误日志（Error Log）：常见的位置是/var/log/mysql/error.log。可以通过查看MySQL的配置文件（通常是/etc/my.cnf、/etc/mysql/my.cnf或~/.my.cnf）中的log-error参数来确定错误日志的确切位置。
2. 查询日志（General Query Log）：如果已启用，其位置由general_log_file参数在配置文件中指定，默认为/var/log/mysql/general.log。
3. 慢查询日志（Slow Query Log）：如果已启用，其位置由slow_query_log_file参数在配置文件中指定，默认为/var/log/mysql/mysql-slow.log。
4. 二进制日志（Binary Log）：如果已启用，其位置由log-bin参数在配置文件中指定，位于/var/lib/mysql/。

4.12.2 中间件-Redis 日志

Redis的日志主要用于记录Redis服务器的运行状态、相关事件以及客户端执行的命令等信息。Redis的日志主要有系统日志和慢查询日志。

1. 系统日志（System Log）：
 - 记录Redis服务器的启动、关闭、配置加载、持久化操作（如 RDB 和 AOF）等系统级事件。
 - Redis的系统日志级别包括debug、verbose、notice、warning和error。您可以通过配置文件redis.conf 中的loglevel参数来设置日志级别。
 - 系统日志的输出方式可以是标准输出（stdout）或文件输出（file）。在配置文件中，使用logfile参数来指定日志文件的路径。
2. 慢查询日志（Slow Query Log）：
 - 用于记录执行时间超过指定阈值的命令。这有助于识别和优化性能瓶颈。
 - 慢查询日志的阈值（即命令执行时间的上限）可以通过 redis.conf 中的slowlog-log-slower-than 参数来设置，单位是微秒。
 - 慢查询日志的最大长度（即可以保存的慢查询记录数）可以通过 slowlog-max-len 参数来设置。
 - 慢查询日志存储在内存中，不会写入磁盘文件。但您可以使用Redis命令SLOWLOG来查看和获取慢查询日志的内容。

采集方法

使用ICAgent采集日志，接入前请参考[安装ICAgent（区域内主机）](#)完成ICAgent安装。然后再通过采集ECS-文本日志方式将日志上报至LTS服务，详细操作请参考[云主机ECS文本日志接入LTS](#)。

Redis 日志示例以及字段含义

1. 系统日志（System Log）：

```
[12345] 2024-04-30 15:30:00.123456 [info] # Server started, Redis version 6.x.x
[12345] 2024-04-30 15:30:15.654321 [notice] * 1 client connected (IP: 127.0.0.1,port: 6379)
[12345] 2024-04-30 15:30:30.987654 [warning] # There are 100 clients waiting in the background,
you may need to check your latency.
[12345] 2024-04-30 15:31:00.000000 [debug] * DB saved on disk
[12345] 2024-04-30 15:32:00.000000 [error] # OOM command not allowed when used memory >
'maxmemory'.
```

表 4-160 系统日志字段说明

字段	说明	示例
PID	Redis服务器的进程ID（PID）。它用于在系统级别上识别和管理Redis进程。	[12345]
时间戳	日志条目的时间戳，通常包括日期、时间和毫秒数。	2024-04-30 15:30:00.123456
日志级别	日志级别 ([info]、[notice]、[warning]、[debug]、[error])：Redis使用不同的日志级别来指示日志条目的重要性。常见的日志级别包括： • [info]：信息性消息，通常表示服务器正常运行的通知。 • [notice]：通知性消息，通常表示重要但不紧急的事件。 • [warning]：警告性消息，表示可能出现问题的情况，但尚未导致故障。 • [debug]：调试信息，通常用于开发人员调试和故障排除。 • [error]：错误消息，表示发生了错误或异常情况。	[info]
日志内容	描述了发生的具体事件或操作。例如“Server started”表示Redis服务器已启动，“1 client connected”表示有一个客户端连接到Redis服务器，“OOM command not allowed”表示由于超出了最大内存限制，Redis拒绝执行某个命令。	Server started, Redis version 6.x.x
附加信息	在某些日志条目中，可能包含有关事件的附加信息。例如在“1 client connected”条目中，包含了连接客户端的IP地址和端口号。	IP: 127.0.0.1,port: 637

2. 慢查询日志（Slow Query Log）

1) 1) (integer) 12345 2) (integer) 1598765432 3) (integer) 123456 4) 1) "GET" 2) "somekey"	

表 4-161 系统日志字段说明

字段	说明	示例
唯一标识符	慢查询日志条目的唯一标识符。随着时间的推移，Redis会继续为新的慢查询日志条目增加序号。	(integer) 12345

字段	说明	示例
时间戳	录了这条慢查询日志产生的时间戳，通常是Unix时间戳（以秒为单位）。	(integer) 1598765432
执行时间	执行慢查询命令所花费的时间，单位是微秒（ μs ）。	(integer) 123456
命令及其参数	"GET": 这是被记录为慢查询的命令本身，这里是GET命令。 "somekey": 这是GET命令的参数，即被检索的键（key）。	-

日志位置

在Redis的配置文件redis.conf中指定日志文件的路径。找到logfile配置项，例如：

```
logfile "/var/log/redis/redis-server.log"
```

4.12.3 中间件-Kafka 日志

Kafka的系统日志（也称为应用日志或运行日志）主要记录了Kafka集群在运行过程中产生的各种信息，例如启动和停止信息、配置加载、节点连接和断开、请求处理等。

采集方法

使用ICAgent采集日志，接入前请参考[安装ICAgent（区域内主机）](#)完成ICAgent安装。然后再通过采集ECS-文本日志方式将日志上报至LTS服务，详细操作请参考[云主机ECS文本日志接入LTS](#)。

Kafka 日志示例以及字段含义

由于Kafka本身并不直接提供系统日志的详细格式或字段定义，因为这部分可能会根据使用的日志框架而有所不同，例如 Log4j，所以本文提供一个基于典型日志框架（如 Log4j）的示例，并对其中的一些通用字段进行解析。

1. 系统日志示例。

```
2023-10-23 15:33:45,123 INFO [KafkaServer id=0] onBrokerStartup (KafkaServer.scala:220) starting
2023-10-23 15:33:45,567 WARN [ReplicaManager broker=0] topic=my_topic partition=0 Broker 1
disconnected; no more in-sync replicas
2023-10-23 15:33:46,987 ERROR [Controller id=0] onControllerFailover (KafkaController.scala:678)
Error during controller failover
java.lang.RuntimeException: Failed to elect new leader for partition [my_topic,0]
    at kafka.controller.KafkaController
    $$
anonfun$onControllerFailover$1
    $$
anonfun$apply$mcV$sp$1.apply$mcV$sp(KafkaController.scala:629)
    at kafka.controller.KafkaController
    $$
anonfun$onControllerFailover$1
    $$
anonfun$apply$mcV$sp$1.apply(KafkaController.scala:609)
    at kafka.controller.KafkaController
    $$
anonfun$onControllerFailover$1
    $$
```

```
anonfun$apply$mcV$sp$1.apply(KafkaController.scala:609)
... (stack trace continues)
```

2. 字段含义。
- 在上面的示例中，第一个日志条目是KafkaServer在启动时产生的信息日志；第二个是一个警告，表明某个分区的一个同步副本（in-sync replica）与broker断开了连接；第三个是一个错误日志，表明在控制器故障转移期间发生了错误，并尝试为某个分区选择新的领导者时失败了。

表 4-162 字段含义

字段	说明	示例
时间戳	日志条目的生成时间。	2023-10-23 15:33:45,123
日志级别	如INFO、WARN、ERROR等，表示日志条目的重要性或紧急性。	INFO
线程或进程信息	提供了产生日志的上下文，如KafkaServer的ID或ReplicaManager的broker ID。	[KafkaServer id=0]
日志来源	指示了日志条目的来源函数和代码位置。	onBrokerStartup (KafkaServer.scala:220)
日志内容	记录了具体的日志信息，可能是描述性的文本、错误消息、堆栈跟踪等。	starting

日志位置

默认情况下，Kafka的系统日志文件可能位于Kafka安装目录的logs子目录下，但具体位置可能会因配置而异。以下是一些常见的方法用于找到Kafka的系统日志文件位置：

- 查看Kafka安装目录。
首先，检查Kafka的安装目录。通常，系统日志文件会存储在安装目录下的一个名为logs的子目录中。例如，如果Kafka安装在`/usr/local/kafka`，则日志文件可能位于`/usr/local/kafka/logs`。
- 检查Kafka配置文件：
在`${KAFKA_HOME}/bin/kafka-run-class.sh`中修改Kafka的logs目录。

```
# Log directory to use
if [ "$LOG_DIR" = "x" ]; then
  # LOG_DIR="$base_dir/logs"
  LOG_DIR="/data/kafka_logs"
fi
```

4.12.4 中间件-Nginx 日志

Nginx是一个开源的、高性能的HTTP和反向代理服务器，以及IMAP/POP3/SMTP代理服务器。它具有其稳定性、丰富的功能集、简单的配置和低系统资源消耗等特点。Nginx常被用于处理静态文件、负载均衡、反向代理以及缓存等功能，适用于各种规模的网站和应用程序。

Nginx日志主要分为两种：访问日志（access_log）和错误日志（error_log）。这些日志对于系统监控、性能分析、故障排查等都非常有用。

- 访问日志（access_log）：访问日志记录了客户端访问Nginx服务器的详细信息。通过访问日志，您可以了解到用户的IP地址、访问的URL、访问时间、请求的处理时间、HTTP响应状态码、请求的头部信息等。
- 错误日志（error_log）：错误日志记录了Nginx服务器在处理请求过程中遇到的错误和警告信息。通过查看错误日志，您可以定位到问题的源头，从而进行故障排除。

采集方法

使用ICAgent采集日志，接入前请参考[安装ICAgent（区域内主机）](#)完成ICAgent安装。然后再通过采集ECS-文本日志方式将日志上报至LTS服务，详细操作请参考[云主机ECS文本日志接入LTS](#)。

Nginx 日志示例以及字段含义

介绍访问日志和错误日志的示例和字段说明。

表 4-163 访问日志字段说明

参数	说明	示例
\$remote_addr	客户端地址。	211.28.65.253
\$remote_user	客户端用户名称。	--
\$time_local	访问时间和时区。	18/Jul/2012:17:00:01 +0800
\$request	请求的URI和HTTP协议。	"GET /article-10000.html HTTP/1.1"
\$http_host	请求地址，即浏览器中您输入的地址（IP或域名）。	www.wang.com 192.168.100.100
\$status	HTTP请求状态。	200
\$upstream_status	upstream状态。	200
\$body_bytes_sent	发送给客户端文件大小。	1547
\$http_referer	url跳转来源。	https://www.example.com/
\$http_user_agent	用户终端浏览器等信息。	"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; SV1; GTB7.0; .NET4.0C;
\$ssl_protocol	SSL协议版本。	TLSv1
\$ssl_cipher	交换数据中的算法。	RC4-SHA
\$upstream_addr	后台upstream的地址，即真正提供服务的主机地址。	10.10.10.100:80
\$request_time	整个请求的总时间。	0.205

参数	说明	示例
\$upstream_response_time	请求过程中，upstream响应时间。	0.002

错误日志示例：

2023/07/10 17:00:00 [error] 12345#12345: *1 recv() failed (104: Connection reset by peer) while reading response header from upstream, client: 192.168.1.100, server: example.com, request: "GET /api/data HTTP/1.1", upstream: "http://10.0.0.1:8080", host: "example.com"

表 4-164 错误日志字段说明

示例	说明
2023/07/10 17:00:00	错误发生的时间。
[error]	表示这是一个错误日志。
12345#12345	进程ID。
*1	连接的编号。
recv() failed (104: Connection reset by peer) while reading response header from upstream	错误信息，表明在从上游读取响应头时接收失败，错误原因是“连接被对方重置”。
client: 192.168.1.100	客户端的IP地址。
server: example.com	服务器的名称。
request: "GET /api/data HTTP/1.1"	请求的方法和路径。
upstream: "http://x.x.x.x:8080"	上游服务器的地址。
host: "example.com"	请求的主机名。

日志位置

在Nginx配置文件中，您可以通过指令来设置访问日志和错误日志的存放位置和格式。例如：

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

    server {
        ...
        access_log /var/log/nginx/access.log main;
        error_log /var/log/nginx/error.log warn;
        ...
    }
}
```

在这个例子中，log_format指令定义了一个名为main的日志格式，然后在server块中使用access_log指令指定了访问日志的存放位置 /var/log/nginx/access.log，错误日志

被存放在/var/log/nginx/error.log路径下，并且日志级别被设置为warn。Nginx支持多种日志级别，包括debug、info、notice、warn、error、crit和alert。

4.12.5 中间件-ElasticSearch 日志

ElasticSearch是一个开源的分布式搜索和分析引擎，被广泛用于构建实时的、可扩展的搜索和分析应用程序。它的主要作用是存储、搜索和分析大量的数据。

ElasticSearch的日志是记录其运行过程中各种信息的重要工具，主要有以下两种类型：

1. 主要日志（cluster-name.log，以集群名称命名：记录了ElasticSearch运行时所发生一切的综合信息，如某个查询失败、一个新的节点加入集群等。这是了解ElasticSearch集群整体运行状况的重要日志。
2. 慢搜索日志（cluster-name_index_search_slowlog.log，以集群名称命名）：当某个查询运行得很慢时，ElasticSearch会在这里进行记录。默认情况下，如果一个查询花费的时间多于半秒，将在这里写入一条记录。这对于分析和优化查询性能非常有用。

除了上述两种主要的日志类型外，ElasticSearch还提供了其他类型的日志，如弃用日志（用于记录已弃用的功能或配置）、访问日志（记录所有查询请求的详细信息）等。

采集方法

使用ICAgent采集日志，接入前请参考[安装ICAgent（区域内主机）](#)完成ICAgent安装。然后再通过采集ECS-文本日志方式将日志上报至LTS服务，详细操作请参考[云主机ECS文本日志接入LTS](#)。

Elasticsearch 日志示例以及字段含义

- 主要日志

```
{
  "log.level": "INFO",
  "@timestamp": "2023-10-23T15:46:30.123Z",
  "log.origin.file.name": "Elasticsearch.java",
  "log.origin.file.line": 123,
  "message": "version[7.17.2], pid[12345], build[default/tar/1234567/2023-10-01T12:00:00.000Z], OS[Linux/5.10.0-10-amd64/amd64], JVM[Eclipse Adoptium/OpenJDK 64-Bit Server VM/11.0.14.1/11.0.14.1+1-LTS]",
  "type": "server",
  "cluster.name": "my-cluster",
  "node.name": "node-1",
  "node.id": "abcdefg123",
  "ecs.version": "1.8.0"
}
{
  "log.level": "WARN",
  "@timestamp": "2023-10-23T15:47:00.456Z",
  "log.origin.file.name": "SlowLog.java",
  "log.origin.file.line": 456,
  "message": "[index][search_slowlog][index_name] took[5.6s], took_millis[5600], total_hits[1000], types[], stats[], search_type[QUERY_THEN_FETCH], source[...]",
  "type": "slowlog",
  "cluster.name": "my-cluster",
  "node.name": "node-1",
  "index.name": "index_name"
}
```



```
{
  "index": "my_index",
  "shard": "0",
  "search_type": "QUERY_THEN_FETCH",
  "total_hits": 12345,
  "status": 200,
  "forced_refresh": false,
  "request_cache": true,
  "scroll_size": 0
}
```

表 4-166 慢搜索日志字段说明

字段	说明	示例
type	日志类型。	"slowlog"，表示这是一个慢查询日志条目。
timestamp	日志条目的时间戳。	2023-10-23T16:15:30,234Z
level	日志级别。	"WARN"，表示这是一个警告级别的日志条目。
component	产生日志的组件。	"o.e.i.s.q"，代表Elasticsearch内部的一个特定组件或模块。
cluster.name	Elasticsearch 集群的名称。	my-cluster
node.name	产生日志的节点的名称。	node-1
node.id	产生日志的节点的ID。	abcdefg123
took	查询执行的总时间，以人类可读的格式（如 "10.5s"）表示。	10.5s
took_millis	查询执行的总时间，以毫秒为单位。	10500
source	触发慢查询的原始查询语句（JSON格式）。	{"query":{"match_all":{}}}\n
shards	慢查询涉及的分片信息数组。	-
index	分片所在的索引名称。	my_index
shard	分片的ID。	0
search_type	查询类型。	QUERY_THEN_FETCH
total_hits	查询返回的总命中数。	12345
status	查询的HTTP状态码。	200
forced_refresh	是否强制刷新分片。	false
request_cache	是否使用请求缓存。	true

字段	说明	示例
scroll_size	滚动查询的大小（如果不适用则为 0）。	0

4.12.6 中间件-Zookeeper 日志

ZooKeeper是一个分布式的、开放源码的分布式应用程序协调服务，是将复杂且容易出错的分布式一致性服务封装起来，构成一个高效可靠的系统，并以一系列简单易用的原子操作提供给用户使用。ZooKeeper的主要功能包括配置维护、域名服务、分布式同步、组服务等。它提供了一个分布式独享锁、选举、队列的接口，主要用来解决分布式系统中一致性、可用性和并发控制等问题。

ZooKeeper 的日志主要分为三种：系统日志（System Logs）、事务日志（Transaction Log）和快照日志（Snapshot Log）：

1. 事务日志（Transaction Log）：

- 事务日志记录了ZooKeeper集群中所有的事务操作，包括客户端的请求和服务端端的响应。
- 事务日志的默认存储位置由zoo.cfg配置文件中的dataLogDir参数指定。如果没有配置dataLogDir，则事务日志将存储在dataDir指定的目录下。
- 在事务日志的目录中，存在version-2的子目录，这个目录表示当前ZooKeeper使用的事务日志格式版本号。在该目录下，存在一系列名为log.<txid>的文件，其中<txid>是一个十六进制数，表示写入该日志的第一个事务的ID。
- 由于事务日志是二进制文件，不能直接通过文本编辑器查看其内容。可以使用ZooKeeper自带的工具（如zkServer.sh脚本中的printlog命令）来查看和解析事务日志。

2. 快照日志（Snapshot Log）：

- 快照日志是ZooKeeper内存数据状态的序列化表示，它包含了ZooKeeper数据树在某个时间点的完整副本。
- 快照日志的存储位置与事务日志相同，也在dataLogDir或dataDir指定的目录下。在快照日志的目录中，存在名为version-2的子目录，以及一系列名为snapshot.<zxid>的文件，其中<zxid>是一个十六进制数，表示触发快照时提交的最后一个事务的ID。
- 与事务日志不同，快照日志是文本文件，可以使用文本编辑器直接查看其内容。由于快照日志包含了大量的数据，因此文件可能会非常大。

3. 系统日志（System Logs）：

ZooKeeper的系统日志（也称为运行日志或应用日志）通常记录了ZooKeeper集群在运行过程中产生的各种信息，包括但不限于启动和停止信息、配置加载、节点连接和断开、请求处理、选举过程等。

采集方法

使用ICAgent采集日志，接入前请参考[安装ICAgent（区域内主机）](#)完成ICAgent安装。然后再通过采集ECS-文本日志方式将日志上报至LTS服务，详细操作请参考[云主机ECS文本日志接入LTS](#)。

Zookeeper 日志示例以及字段含义

```
2024-05-06 10:28:25,123 [myThread-1] INFO org.apache.zookeeper.server.ZooKeeperServer - Accepted socket connection from /192.168.1.100:56789
```

表 4-167 Zookeeper 日志字段说明

字段	说明	示例
时间戳	日志生成的具体时间，以yyyy-MM-dd HH:mm:ss,SSS的格式展示。	2024-05-06 10:28:25,123
线程ID	生成该日志的线程ID。	myThread-1
日志级别	常见的日志级别包括DEBUG（调试）、INFO（信息）、WARN（警告）和ERROR（错误）等。	INFO
日志来源	日志是由org.apache.zookeeper.server.ZooKeeperServer这个类生成的。	org.apache.zookeeper.server.ZooKeeperServer
日志内容	-	Accepted socket connection from / 192.168.1.100:56789 ZooKeeper服务器接受了一个来自IP地址192.168.1.100、端口号56789的socket连接。

日志位置

ZooKeeper的系统日志文件（也称为运行日志或应用日志）的位置通常取决于ZooKeeper的配置。可以通过log4j.properties文件中的配置来指定。如log4j.properties 中的zookeeper.log.dir配置。在log4j.properties文件中，您可以找到一个名为zookeeper.log.dir的配置项，它指定了日志文件的目录。例如：

```
zookeeper.log.dir=/path/to/zookeeper/logs
```

4.12.7 中间件-IIS 日志

Internet Information Services（IIS）简称互联网信息服务，它是Windows Server的一部分，可以用来为网络上的客户端提供HTML、ASP.NET、PHP等类型的Web应用程序和服务。IIS支持多种协议，包括HTTP、HTTPS、FTP、SMTP等，广泛应用于Windows操作系统上的网站部署和管理。

Internet Information Services (IIS) 日志是记录IIS服务器上Web活动的重要文件。这些日志通常包含了关于HTTP请求和响应的详细信息，如请求时间、请求来源、请求的资源、使用的协议版本、用户代理、响应状态等。

IIS日志的字段通常包括：

- 1. 日期和时间：记录请求发生的具体日期和时间。
- 2. 客户端IP地址：发起请求的客户端的IP地址。

- 3. 用户名：如果请求是通过身份验证的，这里会显示用户名；否则可能是“-”。
- 4. 请求方法：如GET、POST等。
- 5. 请求的资源：请求的URL路径。
- 6. 协议版本：如HTTP/1.1或HTTP/2等。
- 7. 用户代理：客户端用来描述自己的字符串，如浏览器类型和版本。
- 8. 响应状态：如200（成功）、404（未找到）等HTTP状态码。
- 9. 响应字节数：服务器发送给客户端的字节数。
- 10. 请求处理时间：请求从开始到结束所花费的时间。

除了这些基本字段外，IIS日志还可以包含其他自定义字段，具体取决于IIS的配置和安装的扩展。

采集方法

使用ICAgent采集日志，接入前请参考[安装ICAgent（区域内主机）](#)完成ICAgent安装。然后再通过采集ECS-文本日志方式将日志上报至LTS服务，详细操作请参考[云主机ECS文本日志接入LTS](#)。

iis 日志示例以及字段含义

```
#Software: Information Services 10.0
#Version: 1.0
#Date: 2023-09-21 02:59:24
#Fields: date time s-ip cs-method cs-uri-stem cs-uri-query sc-status sc-substatus sc-win32-status time-taken
2023-09-21 02:59:24 192.168.1.1 GET /index.html - 301 0 0 123
2023-09-21 02:59:25 192.168.1.1 GET /index.html/ 403 0 2 56
```

表 4-168 iis 日志字段说明

字段	说明	示例
#Software	生成日志的软件信息。	Information Services 10.0
#Version	日志文件的版本信息。	1.0
#Date	日志开始记录的日期和时间。	2023-09-21 02:59:24
#Fields	定义日志条目的字段。	date time s-ip cs-method cs-uri-stem cs-uri-query sc-status sc-substatus sc-win32-status time-taken
date	请求发生的日期。	2023-09-21
time	请求发生的时间。	02:59:24
s-ip	源IP地址，即发起请求的客户端的IP地址。	192.168.1.1
cs-method	请求的HTTP方法。	GET
cs-uri-stem	请求的URL路径。	/index.html
cs-uri-query	请求的URI查询字符串。	在此示例中，第一个请求的查询字符串为空。

字段	说明	示例
sc-status	HTTP状态码，表示服务器的响应状态。	301
sc-substatus	子状态码，提供有关HTTP状态码的更多详细信息。	0
sc-win32-status	Win32状态码，表示与HTTP状态码相关的Windows系统错误代码。	在此示例中，Win32状态码为0或2，表示没有错误或特定的系统错误。
time-taken	请求从开始到结束所花费的时间（以毫秒为单位）。	123

日志位置

Internet Information Services (IIS) 日志文件的位置取决于IIS的配置以及您使用的是IIS的哪个版本。但是，通常情况下，IIS日志文件位于服务器的以下路径之一：

1. Windows Server 2008 R2 及更早版本：
%SystemDrive%\inetpub\logs\LogFiles\W3SVC<站点ID>
2. Windows Server 2012 R2 及更新版本（包括Windows 10上的IIS）：
%SystemDrive%\inetpub\logs\LogFiles\<站点名称>
3. 通过IIS管理器配置：
 - a. 打开IIS管理器：在Windows搜索栏中输入“Internet Information Services (IIS) 管理器”并打开它。或者您可以在“控制面板”->“管理工具”中找到IIS管理器。
 - b. 导航到站点：
 - 在IIS管理器左侧的导航窗格中，展开您的服务器名称。
 - 接着，您会看到“网站”或“应用程序池”等条目。单击“网站”来查看您的网站列表。
 - 在网站列表中，找到您要配置日志的特定网站，然后单击它。
 - c. 配置日志设置：
 - 在选定的网站下，双击“日志记录”以打开日志设置。
 - 在右侧窗格中，您会看到当前日志文件的存储位置和其他相关设置。
 - d. 修改日志文件目录：
 - 在右侧窗格中，找到“目录”部分，单击旁边的“浏览”按钮。
 - 这将打开一个文件浏览器窗口，允许您选择新的日志文件存储位置。
 - 选择您要放置日志文件的文件夹，然后单击“确定”。

4.12.8 服务器-DNS 日志

DNS（Domain Name System，域名系统）是互联网的一项服务，它作为将域名和IP地址相互映射的一个分布式数据库，能够使人更方便地访问互联网。当用户在浏览器

中输入域名时，DNS会将其解析为对应的IP地址，以便计算机能够找到正确的服务器并获取所需的信息。

DNS日志是记录域名系统相关活动的记录文件。它通常包含了域名解析的请求、响应以及可能出现的错误等信息。通过分析DNS日志，可以了解网络中域名解析的情况，有助于排查网络问题、监控网络活动以及进行安全分析等。

DNS记录类型：DNS系统支持多种记录类型，每种类型都用于存储不同类型的信息。最常见的记录类型包括：

- A记录：将域名映射到IPv4地址。
- AAAA记录：将域名映射到IPv6地址。
- CNAME记录：将一个域名映射到另一个域名。
- MX记录：指定邮件服务器的IP地址和优先级。
- NS记录：指定哪个DNS服务器负责解析某个域名的查询。
- SOA记录：包含关于DNS区域的信息，如区域管理员的电子邮件地址和区域的序列号。

采集方法

使用ICAgent采集日志，接入前请参考[安装ICAgent（区域内主机）](#)完成ICAgent安装。然后再通过采集ECS-文本日志方式将日志上报至LTS服务，详细操作请参考[云主机ECS文本日志接入LTS](#)。

DNS 日志示例以及字段含义

日志示例：2023-07-10 14:30:25 INFO client 192.168.2.50 query: example.org IN A

日志说明：2023年7月10日14时30分25秒，客户端IP地址为192.168.2.50发起了对example.org的A记录查询。

字段含义：

- 2023-07-10 14:30:25：表示事件发生的时间。
- client 192.168.2.50：表示发起查询的客户端的IP地址。
- query：表明这是一个查询操作。
- example.org：被查询的域名。
- IN A：表示查询的是A记录（IPv4地址记录）。

4.12.9 中间件-Flink 日志

Flink是一个大数据处理引擎的处理框架，是针对流进行的处理。它是Apache旗下的一个框架和分布式的处理引擎，用于对无界和有界的数据进行状态的计算。Flink的核心特点包括高吞吐、低延迟（每秒处理数百万个事件，毫秒级延迟），以及结果的准确性（Flink提供了时间事件（event-time）和处理事件（processing-time）语义）。

Flink 的系统日志通常包含了关于 Flink 运行时的各种信息，如任务执行情况、错误信息、资源使用情况等。

采集方法

使用ICAgent采集日志，接入前请参考[安装ICAgent（区域内主机）](#)完成ICAgent安装。然后再通过采集ECS-文本日志方式将日志上报至LTS服务，详细操作请参考[云主机ECS文本日志接入LTS](#)。

Flink 日志示例以及字段含义

系统日志示例：

```
2023-10-23 14:35:00,000 INFO org.apache.flink.runtime.taskmanager.Task - Source: Custom Source -> Map (1/1) (83529a6b73e4a3f7046f695a27f67f32) switched from RUNNING to FINISHED.
2023-10-23 14:35:01,500 WARN org.apache.flink.runtime.checkpoint.CheckpointCoordinator - Checkpoint 1 for job 83529a6b73e4a3f7046f695a27f67f32 timed out after 60000 milliseconds. This indicates a problem with the job, as the checkpoints were not completed in time.
2023-10-23 14:35:05,000 ERROR org.apache.flink.runtime.executiongraph.ExecutionGraph - Job 83529a6b73e4a3f7046f695a27f67f32 failed.
```

表 4-169 Flink 系统日志字段含义

字段	说明	示例
时间戳	该条日志的生成时间。	2023-10-23 14:35:00,000
日志级别	该条日志的重要性或紧急程度，常见的级别有 INFO、WARN、ERROR等。	INFO
日志来源	生成该条日志的组件或任务。	org.apache.flink.runtim e.taskmanager.Task
任务/作业信息	日志中可能包含任务或作业的相关信息，如任务名称、任务ID、作业ID等。在这个示例中，有任务名称 Source: Custom Source -> Map (1/1) 和作业ID 83529a6b73e4a3f7046f695a27f67f32。	-
状态变化	记录任务或作业的状态变化。	如switched from RUNNING to FINISHED表示任务从运行状态切换到完成状态。
错误信息	如果日志级别为ERROR，则可能包含有关错误的详细信息。	如 Job 83529a6b73e4a3f7046f695a27f67f32 failed. 表示作业失败。

日志位置

1. 在YARN上运行的Flink 作业：
- Flink-on-YARN 模式的作业日志输出路径通常是 `**$HADOOP_HOME/logs/userlogs/<APPLICATION_ID>/<CONTAINER_ID>/**`。这里的 `$HADOOP_HOME` 是Hadoop的安装目录，`<APPLICATION_ID>`和`<CONTAINER_ID>` 是YARN分配给Flink作业的标识符。可以在`yarn-site.xml`配置文件中修改YARN的日志目录，通过

`<property><name>yarn.nodemanager.log-dirs</name><value>/your/custom/log/path</value></property>`来指定。

2. 独立部署的Flink集群：如果在独立模式下部署Flink集群，那么日志文件通常位于Flink安装目录下的log或logs文件夹中。如果Flink安装在/usr/local/flink，则日志文件可能位于/usr/local/flink/log 或 /usr/local/flink/logs。

5 日志搜索与分析（管道符方式）

5.1 日志搜索与分析概述

日志搜索与分析是运维中不可或缺的一环。日志接入成功后，云日志服务LTS支持对采集成功的日志数据进行搜索与分析。通过合理的日志收集、高效的搜索方法和专业的分析工具，可以实现对系统或应用的全面监控和精细化管理。

了解更多日志搜索与分析的概念信息，请参考[基本概念](#)。

说明

管道符搜索方式仅支持白名单用户使用，如有需要，请提工单申请开通。详细操作请参考[提交工单](#)。

费用说明

搜索分析本身不产生费用，但由于其依赖日志上报至LTS并开启索引，会产生日志读写流量费用、索引流量费用、存储费用等，更多信息请参考[计费项](#)。

约束与限制

日志搜索与分析的限制请参考[搜索与分析限制](#)。

日志结构化

执行搜索与分析前，需要将上报的日志进行结构化配置和索引配置。结构化后的数据具有统一的长度和格式，能够显著提升搜索与分析的效率和准确性。

日志数据可分为结构化数据和非结构化数据。

- 结构化数据是指以表格、关系型数据库等数据模型组织的数据，具有严格的长度和格式，便于存储和分析。
- 非结构化数据是指不便于用数据库二维逻辑表来表现的数据，数据结构不规则或不完整，缺乏预定义的数据模型，难以直接进行分析。

日志结构化是以日志流为单位，通过特定的提取方式，将日志流中格式固定或相似度较高的日志提取出来，同时过滤掉不相关的日志。结构化后的日志可以方便的使用SQL语法进行查询与分析。

日志结构化解析是将非结构化或半结构化的日志数据转换为结构化格式的过程，以便于更好地存储、查询和分析，提高日志数据的可读性、可搜索性和查询效率。

云日志服务支持两种日志结构化解析方式：ICAgent结构化解析和云端结构化解析，且一个日志流只能配置一种结构化方式，例如选择ICAgent结构化解析后，不能再选择云端结构化解析，需要删除后，才能重新选择。更多信息请参考图5-1。

若用户在日志接入时已经配置ICAgent结构化解析，则无需再配置云端结构化解析。

- ICAgent结构化解析是在采集侧做结构化，支持插件组合解析，单个日志流的多个采集配置支持不同结构化解析规则，推荐使用ICAgent结构化解析的方式，更多内容请参考[配置ICAgent结构化解析](#)。
- 云端结构化解析是通过不同的日志提取方式将日志流中的日志进行结构化，云端结构化解析会消耗LTS系统端算力，未来会按照日志量大小收取日志加工流量费用。

图 5-1 不同解析方式



日志搜索与分析

结构化完成后，使用云日志服务LTS提供的[搜索语法](#)、[SQL函数](#)用于设置搜索条件，通过管道符联动搜索语句和SQL分析语句，可以更高效地提取和分析日志数据，将分析结果通过统计图表可视化展示。详细请参考[搜索与分析日志](#)。

5.2 设置云端结构化解析日志

5.2.1 设置日志云端结构化解析

云日志服务LTS目前支持五种日志结构化方式，分别是正则分析、JSON、分隔符、Nginx和结构化模板。您可以根据日志内容的实际场景进行选择。

- **正则分析**：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。使用正则表达式提取字段，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。
- **JSON**：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。
- **分隔符**：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（例如：冒号、空格或字符等）提取为多个key-value键值的日志解析模式。
- **Nginx**：适用于日志文本中每行内容为一条原始日志，每条日志符合Nginx格式，支持通过log_format指令来自定义访问日志的格式。
- **结构化模板**：适用于日志结构比较复杂或需要自定义提取key-value键值的场景，可以通过内置系统模板或者自定义模板提取字段。

结构化后的日志数据可理解为数据库中的二维表，结构化配置完成后就可以使用SQL语句对提取的字段进行查询与分析。

使用限制

- 若未配置索引配置，结构化字段默认分词符为空，最大长度限制为20KB，超过部分会被截断。
- 若已配置索引配置，结构化字段默认分词符参考[设置LTS日志内容分词](#)，此时最大长度限制为500KB。

注意事项

- 日志结构化是以日志流为单位。
- 日志流中的大部分日志需有一定的规则，否则结构化是无意义的。
- 结构化配置修改后，对新写入的日志数据生效，历史日志数据不会生效。

云端结构化解析

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 单击目标日志组或日志流名称，进入日志流详情页面。

步骤3 在“日志搜索”页面，单击右上角的日志设置，在弹出页面中，选择“云端结构化解析”，进行日志结构化配置。

- 开启“保留原始日志”后，原始日志将作为content字段的值存储到云日志服务。同时在资源统计和计费时content字段均会统计在内。
- 开启“上传解析失败日志”后，原始日志将作为_content_parse_fail字段的值上传到云日志服务。
- 关于开启或关闭“保留原始日志”和“上传解析失败日志”时，日志上报参考如下。

图 5-2 云端结构化解析



表 5-1 日志上报说明

参数	日志上报说明
- 开启“保留原始日志” - 开启“上传解析失败日志”	- 解析成功：上报解析后日志和原始日志content字段。 - 解析失败：为了避免日志冗余，只上报原始日志content字段，不重复上报_content_parse_fail_字段。
- 开启“保留原始日志” - 关闭“上传解析失败日志”	- 解析成功：上报解析后日志和原始日志content字段。 - 解析失败：上报原始日志content字段。
- 关闭“保留原始日志” - 开启“上传解析失败日志”	- 解析成功：上报解析后日志。 - 解析失败：上报_content_parse_fail_字段。
- 关闭“保留原始日志” - 关闭“上传解析失败日志”	- 解析成功：上报解析后日志。 - 解析失败：仅上报内置字段和label字段。

- 结构化不支持的系统字段包括：groupName、logStream、lineNum、content、logContent、logContentSize、collectTime、category、clusterId、clusterName、containerName、hostIP、hostId、hostName、nameSpace、pathFile、podName。
- **正则分析**：使用正则表达式提取字段。
- **JSON**：通过提取JSON字段将其拆分为键值对。
- **分隔符**：使用分隔符（例如：冒号、空格或字符等）提取字段。
- **Nginx**：通过log_format指令来自定义访问日志的格式。
- **结构化模板**：通过自定义模板或系统内置模板提取字段。

步骤4 云端结构化解析配置完成后，支持修改或删除结构化配置。

- 在云端结构化解析页面中，单击 ，修改结构化配置。
- 在云端结构化解析页面中，单击 ，删除结构化配置。

结构化配置删除后，将无法恢复，请谨慎操作。

----结束

正则分析

正则分析是使用正则表达式提取字段。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。

- **从已有日志中选择**：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。
 - 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
 - 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
 - 自定义：表示查询指定时间范围的日志数据。
- **从剪贴板中粘贴**：单击“从剪贴板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。

步骤2 字段提取。包括自动生成和手动输入两种方式，可将选择的日志提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

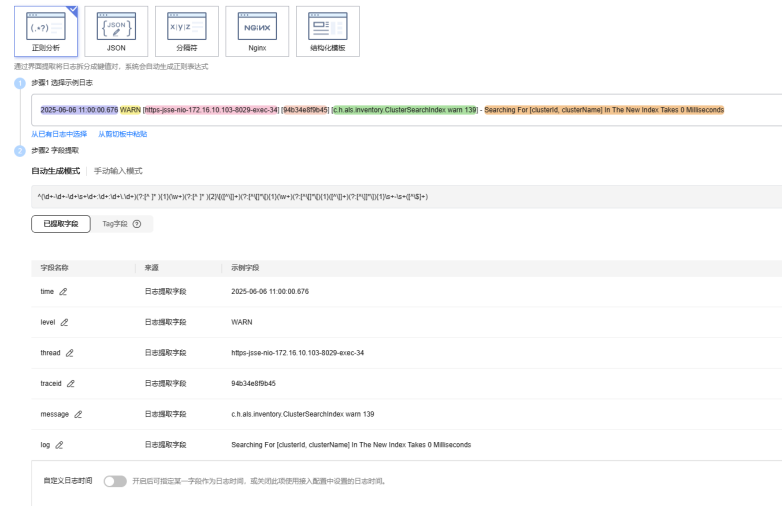
- **自动生成模式**：当用户选择自动生成时，可以用鼠标选中示例日志中待结构化的日志内容，在弹出的对话框中为选中内容设置一个名称，单击“添加”。字段名称只支持输入英文、数字、中划线、下划线及小数点，不能以小数点开头或结尾且不能以双下划线结尾。

例如在选择日志示例下方输入以下原始日志示例：

```
2025-06-06 11:00:00.676 WARN [https-jsse-nio-172.16.10.103-8029-exec-34] [94b34e8f9b45]
[c.h.als.inventory.ClusterSearchIndex warn 139] - Searching For [clusterId, clusterName] In The New
Index Takes 0 Milliseconds
```

根据原始日志示例，设置字段名称为time、level、thread、message等，自动生成字段参考如图5-3所示。

图 5-3 正则分析自动生成字段



- 手动输入模式：当用户选择手动输入时，可以在输入框中输入正则表达式，单击“生成字段”来进行字段提取。正则表达式通过分组来捕获字段，分组指用圆括号"()"括起来的正则表达式，匹配出的内容就表示一个分组，分组包含如下三种形式：
 - (exp)：把括号内的正则作为一个分组，系统自动分配组号，规则为从正则表达式的左边开始，第一个左括号“(”对应第一个分组，第二个“(”对应第二个分组，依次类推，组号从1开始，从左向右，依次累加。
 - (?<name>exp)：表示命名分组，分组的正则表达式为exp，分组名为name。分组名必须以字母开始，且仅包含字母和数字，可以通过分组名或分组号引用该分组。
 - (?exp)：表示不捕获分组，该分组只在当前位置匹配文本，在该分组之后，无法引用该分组，因为该分组没有分组名，没有分组号，也不会占用分组编号。
 - 分词符指将日志内容切分为多个单词的符号，默认分词符包括, ";=()[]{}@&<>/:\\?\\n\\t\\r，在日志搜索或者对日志进行结构化时，可以选取相邻两分词符之间的单词。
 - 在手工输入方式中，正则表达式的长度不能超过5000个字符，不强制要求用户在输入正则表达式时对分组进行命名，单击“生成字段”会以命名分组中的分组名作为字段名称，对于非命名分组会提取出对应的字段，并给字段名称默认命名field1、field2、field3……。
 - 由于手动输入正则表达式时不能填写双下划线，所以在日志提取字段下方可以将字段名称重命名为带有双下划线的字段。

步骤3 若需要指定某一字段作为日志时间，详细请参考[自定义日志时间](#)。

步骤4 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

JSON

JSON是通过提取JSON字段将其拆分为键值对。

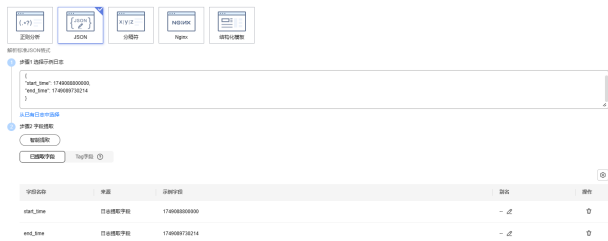
步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

步骤2 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

将以下参考示例原始日志输入待操作框中，在“步骤2 字段提取”下单击“智能提取”。自动生成字段参考如[图5-4](#)所示。

```
{
  "start_time": 1749088800000,
  "end_time": 1749089730214
}
```


图 5-4 JSON 字段提取结果



在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考[设置云端结构化字段](#)。

- 当日志提取字段的类型为float时，精度为16位有效数字。如果超过16位有效数字，则会导致提取字段内容不准确，从而影响快速分析，因此建议将字段类型修改为String。
- 当日志提取字段的类型为long时，日志内容超过16位有效数字，只会精确显示前16位有效数字，后面的数字会变为0。
- 当日志提取字段的类型为long时，日志内容超过21位有效数字，则会识别为float类型，建议将字段类型修改为String。

步骤3 若需要指定某一字段作为日志时间，详细请参考[自定义日志时间](#)。

步骤4 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

分隔符

分隔符是使用分隔符（例如：逗号、空格或字符）提取字段。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

步骤2 在“步骤2 指定分隔符”需要根据原始日志内容选择分隔符，或自定义其他需要的特殊字符作为分隔符。

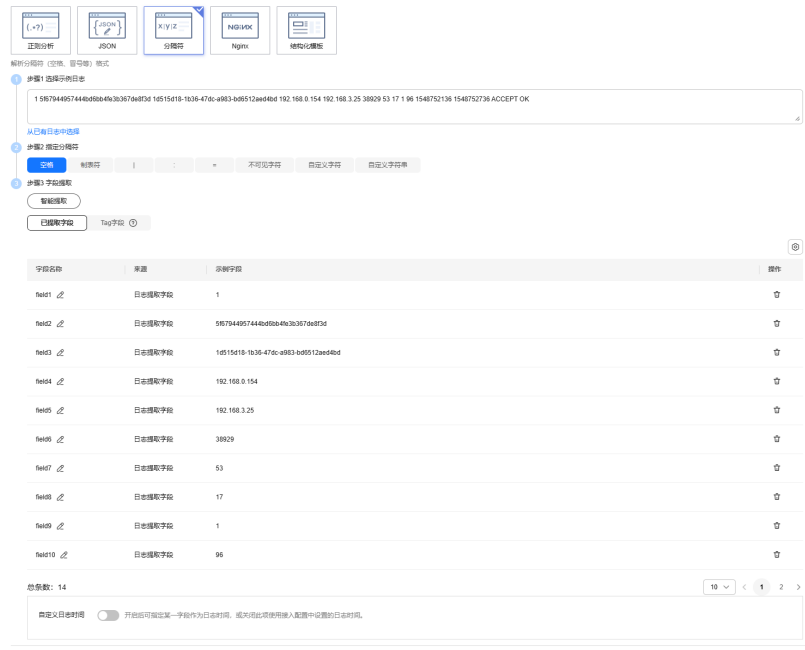
- 不可见字符需要输入长度为4个且以0x开头的16进制字符，例如0x01，最多可输入32个不可见字符。
- 自定义字符支持输入1-10个字符，每个字符都作为独立的分隔符。
- 自定义字符串支持输入1-30个字符，字符串整体作为一个分隔符。

步骤3 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

将以下原始日志示例输入待操作框中，在“步骤3字段提取”下单击“智能提取”。自动生成字段参考如图5-5所示。

```
1 5f67944957444bd6bb4fe3b367de8f3d 1d515d18-1b36-47dc-a983-bd6512aed4bd 192.168.0.154
192.168.3.25 38929 53 17 1 96 1548752136 1548752736 ACCEPT OK
```

图 5-5 分隔符字段提取结果



在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考[设置结构化字段](#)。

当日志提取字段的类型为float时，精确度为7位有效数字。

如果超过7位有效数字的话，则会导致提取字段内容不准确，从而影响快速分析，因此建议将字段类型修改为String。

步骤4 若需要指定某一字段作为日志时间，详细请参考[自定义日志时间](#)。

步骤5 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

Nginx

Nginx是通过log_format指令来自定义访问日志的格式。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

步骤2 在“步骤2 输入Nginx日志配置”中需要输入Nginx日志配置，根据输入或选择的日志进行配置。其中有默认配置可使用，单击“默认Nginx配置”即可。

标准Nginx配置文件中，日志配置的部分通常以log_format开头。

- 默认配置如下所示，参考表5-2查看字段说明。除了默认字段，请参考表5-3查看常用扩展字段说明。
- ```
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
 '$status $body_bytes_sent "$http_referer" '
 '"$http_user_agent" "$http_x_forwarded_for";'
```

表 5-2 默认配置字段参数

| 字段名称                 | 说明                    |
|----------------------|-----------------------|
| remote_addr          | 客户端IP地址。              |
| remote_user          | 客户端用户名。               |
| time_local           | 服务器时间，前后必须加上中括号（[]）。  |
| request              | 请求的URI和HTTP协议。        |
| status               | 请求状态。                 |
| body_bytes_sent      | 发送给客户端的字节数，不包括响应头的大小。 |
| http_referer         | URL跳转来源。              |
| http_user_agent      | 客户端浏览器等信息。            |
| http_x_forwarded_for | 通过代理服务器转发的客户端真实IP地址。  |

表 5-3 扩展字段参数

| 字段名称                   | 说明                   |
|------------------------|----------------------|
| request_time           | 整个请求的总时间（单位：秒，精度毫秒）。 |
| upstream_response_time | 后端服务器响应时间。           |
| http_cookie            | 客户端发送的Cookie数据。      |
| http_x_real_ip         | 客户端的真实IP地址。          |

| 字段名称            | 说明                                          |
|-----------------|---------------------------------------------|
| uri             | 请求的URI路径（如： /index.html ）。                  |
| args            | 请求行中的参数（如： ? param1=value1&param2=value2 ）。 |
| content_length  | 请求头中的Content-Length字段。                      |
| content_type    | 请求头中的Content-Type字段。                        |
| host            | 请求头中的Host字段，或处理请求的服务器名称。                    |
| server_addr     | 服务器地址。                                      |
| server_name     | 服务器名称。                                      |
| server_port     | 服务器端口。                                      |
| server_protocol | 服务器向客户端发送响应时的协议版本（如： HTTP/1.1 ）。            |
| scheme          | 请求协议（如： http或https ）。                       |

- 用户也可进行自定义配置，具体配置格式要求如下所示。
  - 使用Nginx配置，不可为空。
  - 以log\_format开头，并且包含（'）和字段名称。
  - 长度最大限制为5000。
  - 需要与示例日志内容匹配。
  - log\_format字段之间的间隔，除大小字母、数字、下划线及中划线外，可使用其他任意字符。
  - 以（'）或者（';）结尾。

**步骤3** 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

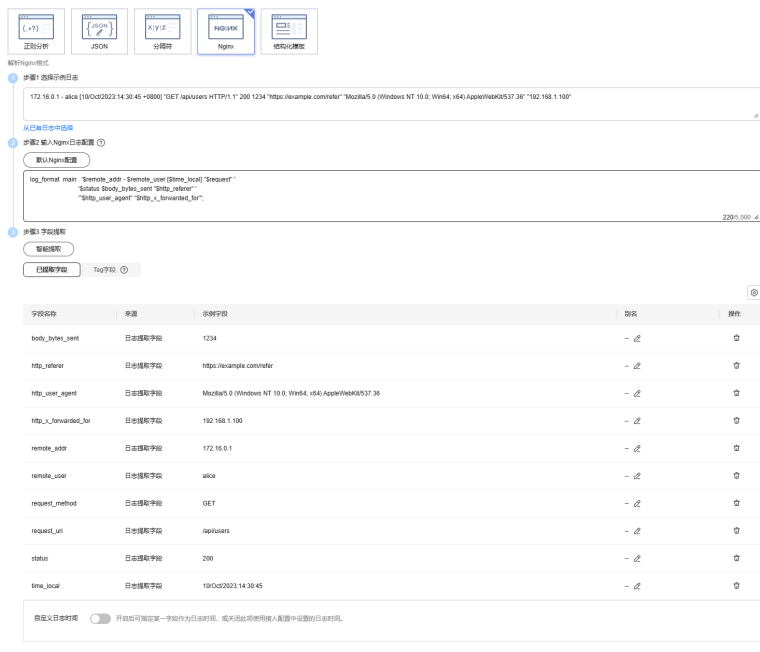
将以下原始日志示例输入待操作框中，在“步骤3 字段提取”下单击“智能提取”。自动生成字段参考如图5-6所示。

```
172.16.0.1 - alice [10/Oct/2023:14:30:45 +0800] "GET /api/users HTTP/1.1" 200 1234 "https://example.com/refer" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36" "192.168.1.100"
```

并使用如下Nginx日志配置。

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

图 5-6 Nginx 字段提取结果



在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考[设置结构化字段](#)。

- 当日志提取字段的类型为float时，精确度为7位有效数字。
- 如果超过7位有效数字的话，则会导致提取字段内容不准确，从而影响快速分析，因此建议将字段类型修改为String。

**步骤4** 若需要指定某一字段作为日志时间，详细请参考[自定义日志时间](#)。

**步骤5** 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

结构化模板

结构化模板是通过自定义模板或系统内置模板提取字段。

详情请参考[结构化模板](#)。

相关文档

LTS支持通过调用API接口实现创建结构化配置、查询结构化配置、查询结构化模板、删除结构化配置等，详细请参考[云端结构化配置](#)。

5.2.2 设置云端结构化字段和 tag 字段

设置云端结构化字段

在进行结构化配置字段提取之后，可对结构化字段进行设置，详细请参考[表5-4](#)。

表 5-4 结构化字段设置规则

| 日志提取方式     | 字段名称                                                                                                      | 字段类型是否可修改 | 字段是否可删除 |
|------------|-----------------------------------------------------------------------------------------------------------|-----------|---------|
| 正则分析（自动生成） | 用户自定义。<br>名称必须以字母开始，且仅包含字母和数字。                                                                            | 是         | 是       |
| 正则分析（手动输入） | <ul style="list-style-type: none"><li>支持在输入正则表达式时进行命名。</li><li>支持使用系统默认命名field1、field2、field3等。</li></ul> | 是         | 是       |
| JSON格式     | 智能提取字段名称，可定义别名。                                                                                           | 是         | 是       |
| 分隔符        | 默认名称field1、field2、field3……，可进行修改。                                                                         | 是         | 是       |
| Nginx      | 根据Nginx配置生成，可定义别名。                                                                                        | 是         | 是       |
| 自定义模板      | 用户自定义。                                                                                                    | 是         | 是       |

正则分析（手动输入）、JSON格式、分隔符、Nginx和自定义模板的字段名称需要满足如下要求：

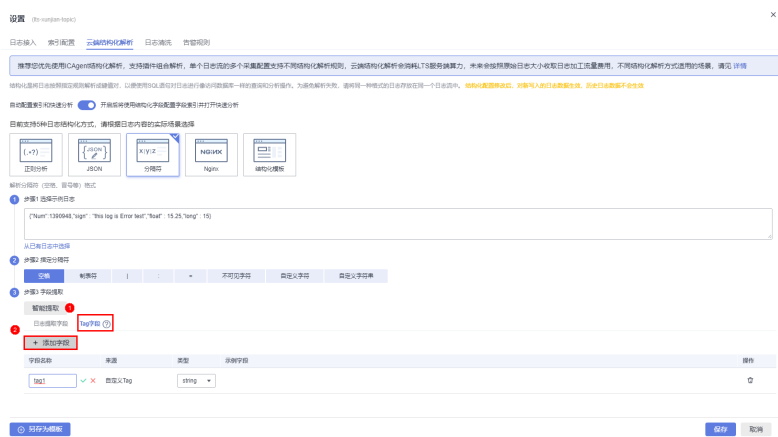
只支持输入英文、数字、中划线、下划线及小数点，不能包含双下划线、下划线加点且不能以小数点开头或结尾。长度为1-64个字符。

设置 tag 字段

设置结构化配置时，可以对日志维度信息进行tag字段设置。设置完成后可以在日志分析页面对设置字段进行SQL查询。

**步骤1** 在字段提取步骤中，选择“tag字段”页签，单击“添加字段”。

图 5-7 添加 tag 字段



**步骤2** 在tag字段列表中，输入需要设置tag“字段名称”。

tag字段功能上线前，已设置的结构化配置，在修改结构化配置进行tag字段设置时，系统tag不会带出示例字段。

**步骤3** 如需添加多个字段可单击“添加字段”，继续添加。

**步骤4** 设置完成后，单击“保存”。

----结束

### 5.2.3 设置云端结构化自定义日志时间

您可以开启“自定义日志时间”，使用自定义日志内容的时间作为日志搜索页面日志内容的时间。关闭“自定义日志时间”后，则使用日志采集时间作为日志搜索页面日志内容的时间。

#### 开启自定义日志时间

**步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。

**步骤2** 单击目标日志组或日志流名称，进入日志流详情页面。

**步骤3** 在“日志搜索”页面，单击右上角的日志设置，在弹出页面中，选择“云端结构化解析”，详细请参考[设置日志云端结构化解析](#)。

**步骤4** 配置完成后，开启自定义日志时间开关，配置如下参数。

切换自定义日志时间开关时，可能会导致日志搜索界面在切换时间点附近出现时间偏差，请勿频繁切换自定义日志时间开关。

表 5-5 参数配置表

| 参数      | 说明                                        | 示例                  |
|---------|-------------------------------------------|---------------------|
| 字段key   | 已提取字段的名称。单击下拉框选择已提取的字段，该字段为string或long类型。 | test                |
| 字段value | 已提取的字段value，选择字段key后，将自动填充。               | 2022-07-19 12:12:00 |
| 时间格式    | 请参考 <a href="#">常见日志时间格式</a> 。            | yyyy-MM-dd HH:mm:ss |
| 操作      | 单击“校验”，提示“时间格式和字段value匹配成功”则表示校验成功。       | -                   |

----结束

#### 常见日志时间格式

常见日志时间格式如下[表5-6](#)。

默认情况下，云日志服务LTS中的日志时间戳精确到秒，因此时间格式只需配置到秒，无需配置毫秒、微秒等信息。

表 5-6 时间格式

| 时间格式                        | 说明                         | 示例                          |
|-----------------------------|----------------------------|-----------------------------|
| EEE                         | 星期的缩写。                     | Fri                         |
| EEEE                        | 星期的全称。                     | Friday                      |
| MMM                         | 月份的缩写。                     | Jan                         |
| MMMM                        | 月份的全称。                     | January                     |
| dd                          | 每月第几天，十进制，范围为01~31。        | 07, 31                      |
| HH                          | 小时，24小时制。                  | 22                          |
| hh                          | 小时，12小时制。                  | 11                          |
| MM                          | 月份，十进制，范围为01~12。           | 08                          |
| mm                          | 分钟，十进制，范围为00~59。           | 59                          |
| a                           | AM或PM。                     | AM、PM                       |
| hh:mm:ss a                  | 12小时制的时间组合。                | 11:59:59 AM                 |
| HH:mm                       | 小时和分钟组合。                   | 23:59                       |
| ss                          | 秒数，十进制，范围为00~59。           | 59                          |
| yy                          | 年份，十进制，不带世纪，范围为00~99。      | 04、98                       |
| yyyy                        | 年份，十进制。                    | 2004、1998                   |
| d                           | 每月第几天，十进制，范围为1~31。         | 7、31                        |
| DDD                         | 一年中的天数，十进制，范围为001~366。     | 365                         |
| u                           | 星期几，十进制，范围为1~7，1表示周一。      | 2                           |
| w                           | 每年的第几周，星期天是一周的开始，范围为00~53。 | 23                          |
| W                           | 每月的第几周，范围为0~5。             | 2                           |
| U                           | 星期几，十进制，范围为0~6，0代表周日。      | 5                           |
| EEE MMM dd<br>HH:mm:ss yyyy | 标准的日期和时间。                  | Tue Nov 20 14:12:58<br>2020 |
| EEE MMM dd<br>yyyy          | 标准的日期，不带时间。                | Tue Nov 20 2020             |
| HH:mm:ss                    | 标准的时间，不带日期。                | 11:59:59                    |
| %s                          | Unix时间戳。                   | 147618725                   |

示例

常见的时间标准、示例及对应的时间表达式如下表5-7。

表 5-7 示例

| 示例                             | 时间表达式                       | 时间标准    |
|--------------------------------|-----------------------------|---------|
| 2022-07-14T19:57:36+08:00      | yyyy-MM-dd'T'HH:mm:ssXXX    | 自定义     |
| 1548752136                     | %s                          | 自定义     |
| 27/Jan/2022:15:56:44           | dd/MMM/yyyy:HH:mm:ss        | 自定义     |
| 2022-08-15 17:53:23+08         | yyyy-MM-dd HH:mm:ssX        | 自定义     |
| 2022-08-05T08:24:15.536+0000   | yyyy-MM-dd'T'HH:mm:ss.SSSZ  | 自定义     |
| 2022-08-20T10:04:03.204000Z    | yyyy-MM-dd'T'HH:mm:ss.SSSZ  | 自定义     |
| 2022-08-22T06:52:08Z           | yyyy-MM-dd'T'HH:mm:ssZ      | 自定义     |
| 2022-07-24T10:06:41.000        | yyyy-MM-dd'T'HH:mm:ss.SSS   | 自定义     |
| Monday, 02-Jan-06 15:04:05 MST | EEEE, dd-MMM-yy HH:mm:ss Z  | RFC850  |
| Mon, 02 Jan 2006 15:04:05 MST  | EEE, dd MMM yyyy HH:mm:ss Z | RFC1123 |
| 02 Jan 06 15:04 MST            | dd MMM yy HH:mm Z           | RFC822  |
| 02 Jan 06 15:04 -0700          | dd MMM yy HH:mm Z           | RFC822Z |
| 2023-01-02T15:04:05Z07:00      | yyyy-MM-dd'T'HH:mm:ss Z     | RFC3339 |
| 2022-12-11 15:05:07            | yyyy-MM-dd HH:mm:ss         | 自定义     |

5.2.4 设置云端结构化模板

云日志服务LTS目前支持的结构化模板有两种，分别为系统模板和自定义模板。

系统模板

支持多种系统模板，不支持修改系统模板的字段类型和删除字段，详情请参考表5-8。

- 步骤1 在云端结构化解析页面中，日志结构化方式选择“结构化模板”。
- 步骤2 在“选择模板”下，选择“系统模板”，选择对应的系统模板，模板日志从对应的云服务接入，可以直接应用模板的数据模型作为示例日志。

**步骤3** 选择模板后，“模板详情”中会自动显示对应的日志解析结果。

- 结构化配置时，如果使用系统模板，则系统模板中的时间为自定义日志时间。支持通过模板名称搜索模板，方便用户快速查询模板信息。
- string类型的字段不支持使用运算符（>=<）或 in 语法进行范围查询，建议使用星号（\*）或问号（?）进行模糊查询。需要重新配置结构化，将该字段修改为数字类型。

**表 5-8** 系统模板

| 模板名称        | 模板说明                           | 字段类型是否可修改 | 字段是否可删除 | 系统模板的请求参数               |
|-------------|--------------------------------|-----------|---------|-------------------------|
| ELB模板       | 根据ELB资料中提供的日志字段被定义。            | 否         | 否       | ELB                     |
| VPC模板       | 根据VPC资料中提供的日志字段被定义。            | 否         | 否       | VPC                     |
| CTS模板       | 字段名称为json日志中的key。              | 否         | 否       | CTS                     |
| APIG模板      | 根据APIG资料中提供的日志字段被定义。           | 否         | 否       | APIG                    |
| DCS审计日志     | 根据DCS资料中提供的日志字段被定义。            | 否         | 否       | DCS_AUDIT               |
| TOMCAT      | 根据TOMCAT官网提供的字段名称进行nginx解析的名称。 | 否         | 否       | TOMCAT                  |
| NGINX       | 根据NGINX资料中提供的日志字段被定义。          | 否         | 否       | NGINX                   |
| GAUSSV5审计日志 | 根据GAUSSV5资料中提供的日志字段被定义。        | 否         | 否       | GAUSSDB_OPENGAUSS_AUDIT |
| DDS审计日志     | 根据DDS资料中提供的日志字段被定义。            | 否         | 否       | DDS_AUDIT               |
| DDS错误日志     | 根据DDS资料中提供的日志字段被定义。            | 否         | 否       | MONGODB_ERROR           |
| DDS慢日志      | 根据DDS资料中提供的日志字段被定义。            | 否         | 否       | MONGODB_SLOW            |
| CFW访问控制日志   | 根据CFW资料中提供的日志字段被定义。            | 否         | 否       | CFW_ACCESS              |
| CFW攻击日志     | 根据CFW资料中提供的日志字段被定义。            | 否         | 否       | CFW_ATTACK              |
| CFW流量日志     | 根据CFW资料中提供的日志字段被定义。            | 否         | 否       | CFW_FLOW                |



| 模板名称                  | 模板说明                                  | 字段类型是否可修改 | 字段是否可删除 | 系统模板的请求参数              |
|-----------------------|---------------------------------------|-----------|---------|------------------------|
| MYSQL错误日志             | 根据MYSQL资料中提供的日志字段被定义。                 | 否         | 否       | MYSQL_ERROR            |
| MYSQL慢日志              | 根据MYSQL资料中提供的日志字段被定义。                 | 否         | 否       | MYSQL_SLOW             |
| POSTGRESQL慢日志         | 根据POSTGRESQL慢日志资料中提供的日志字段被定义。         | 否         | 否       | POSTGRESQL_SLOW        |
| POSTGRESQL错误日志        | 根据POSTGRESQL错误日志资料中提供的日志字段被定义。        | 否         | 否       | POSTGRESQL_ERROR       |
| SQLSERVER错误日志         | 根据SQLSERVER资料中提供的日志字段被定义。             | 否         | 否       | SQLSERVER_ERROR        |
| GeminiDB Redis慢日志     | 根据GeminiDB Redis资料中提供的日志字段被定义。        | 否         | 否       | GAUSSDB_REDIS_SLOW     |
| CDN                   | 根据CDN资料中提供的日志字段被定义。                   | 否         | 否       | CDN                    |
| SMN                   | 根据SMN资料中提供的日志字段被定义。                   | 否         | 否       | SMN                    |
| GAUSSDB_MYSQL错误日志     | 根据GAUSSDB_MYSQL资料中提供的日志字段被定义。         | 否         | 否       | GAUSSDB_MYSQL_ERROR    |
| GAUSSDB_MYSQL慢日志      | 根据GAUSSDB_MYSQL资料中提供的日志字段被定义。         | 否         | 否       | GAUSSDB_MYSQL_SLOW     |
| ER企业路由器               | 根据ER企业路由器资料中提供的日志字段被定义。               | 否         | 否       | ER                     |
| MYSQL审计日志             | 根据MYSQL审计日志资料中提供的日志字段被定义。             | 否         | 否       | MYSQL_AUDIT            |
| GeminiDB Cassandra慢日志 | 根据GeminiDB Cassandra慢日志资料中提供的日志字段被定义。 | 否         | 否       | GAUSSDB_CASSANDRA_SLOW |
| GeminiDB Mongo慢日志     | 根据GeminiDB Mongo慢日志资料中提供的日志字段被定义。     | 否         | 否       | GAUSSDB_MONGO_SLOW     |
| GeminiDB Mongo错误日志    | 根据GeminiDB Mongo错误日志资料中提供的日志字段被定义。    | 否         | 否       | GAUSSDB_MONGO_ERROR    |
| WAF访问日志               | 根据WAF访问日志资料中提供的日志字段被定义。               | 否         | 否       | WAF_ACCESS             |

| 模板名称                 | 模板说明                                | 字段类型是否可修改 | 字段是否可删除 | 系统模板的请求参数              |
|----------------------|-------------------------------------|-----------|---------|------------------------|
| WAF攻击日志              | 根据WAF攻击日志资料中提供的日志字段被定义。             | 否         | 否       | WAF_ATTACK             |
| DMS重平衡日志             | 根据DMS重平衡日志资料中提供的日志字段被定义。            | 否         | 否       | DMS_REBALANCED         |
| CCE审计日志              | 根据CCE审计日志资料中提供的日志字段被定义。             | 否         | 否       | CCE_AUDIT              |
| CCE事件日志              | 根据CCE事件日志资料中提供的日志字段被定义。             | 否         | 否       | CCE_EVENT              |
| CCE NGINX-INGRESS日志  | 根据CCE NGINX-INGRESS日志资料中提供的日志字段被定义。 | 否         | 否       | NGINX_INGRESS          |
| GeminiDB Redis审计日志   | 根据GeminiDB Redis审计日志资料中提供的日志字段被定义。  | 否         | 否       | GAUSSDB_REDIS_AUDIT    |
| influx慢日志            | 根据influx慢日志资料中提供的日志字段被定义。           | 否         | 否       | GAUSSDB_INFLUX_SLOW    |
| METRIC系统日志           | 根据日志生成指标任务过滤的监控日志字段被定义。             | 否         | 否       | SYS_LOG_FOR_LOG2METRIC |
| Microgateway         | 根据Microgateway应用网关提供的日志字段被定义。       | 否         | 否       | MICROGATEWAY           |
| GeminiDB Mongo接口审计日志 | 根据GeminiDB Mongo接口审计日志提供的日志字段被定义。   | 否         | 否       | GEMINIDB_MONGODB_AUDIT |
| CloudTable Doris审计日志 | 根据CloudTable Doris审计日志提供的日志字段被定义。   | 否         | 否       | CLOUD_TABLE_DORIS      |
| CoreDNS日志            | 根据CoreDNS日志提供的日志字段被定义。              | 否         | 否       | COREDNS_LOG            |
| TaurusDB审计日志         | 根据TaurusDB审计日志提供的日志字段被定义。           | 否         | 否       | TAURUS_DB_AUDIT        |
| Workspace用户操作日志      | 根据Workspace用户操作日志提供的日志字段被定义。        | 否         | 否       | WKS_USER_EVENT         |
| DNS解析请求日志            | 根据DNS解析请求日志提供的日志字段被定义。              | 否         | 否       | DNS_QUERY_LOG          |

**步骤4** 单击“保存”完成结构化配置。

----结束

## 自定义模板

在“选择模板”下，选择“自定义模板”，选择已有的结构化模板。自定义模板主要来源有以下两种：

- 在配置正则分析、JSON、分隔符或Nginx方式时，单击左下角的“另存为模板”，系统会弹出“另存为模板”页面，输入模板名称，单击“确定”，保存自定义模板。该模板会在“自定义模板”下的模板列表展示。
- 新增结构化模板，具体操作如下：
  - 在“选择模板”下，选择“自定义模板”，单击“新增结构化模板”。
  - 在“新增结构化模板”页面中，选择正则分析、JSON、分隔符或Nginx方式，进行配置。
  - 配置完成后输入模板名称，单击“确定”，完成自定义模板的保存，会在“自定义模板”下的模板列表展示。

## 5.3 创建 LTS 日志索引

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

### 索引类型

云日志服务LTS支持全文索引和字段索引，详细请参考[表5-9](#)。

表 5-9 索引类型

| 索引类型 | 说明                                                                                                                                                                                                                                         |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 全文索引 | <p>开启全文索引后，日志服务根据您的分词符将整条日志所有字段值拆分成多个词并构建索引。</p> <ul style="list-style-type: none"><li>用户上传的自定义标签（label）字段，不包含在全文索引中，如果您需要搜索自定义标签字段，请添加对应的字段索引。</li><li>LTS内置保留字段，不包含在全文索引中，您需要通过字段索引Key:Value的方式进行搜索，请参考<a href="#">内置保留字段</a>。</li></ul> |

| 索引类型 | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 字段索引 | <p>配置字段索引后，您可以指定字段名称和字段值（Key:Value）进行查询，缩小查询范围。</p> <ul style="list-style-type: none"><li>● 日志服务默认为部分内置保留字段创建字段索引，请参考<a href="#">内置保留字段</a>。</li><li>● 如果您的某个字段单独配置了字段索引，那么该字段值的分词符以字段索引配置为准。</li><li>● 结构化配置中的快速分析列已被移除，如果您要使用<a href="#">快速分析功能</a>，则必须配置字段索引且开启对应字段的快速分析按钮。</li><li>● 例如以下日志示例，如果配置了level和status两个字段索引，其中level是string类型，字段值是error，单独配置了分词符，status是long类型，不需要配置分词符；您可以使用level:error的方式精确搜索level字段值为error的所有日志。</li><li>● 例如以下日志示例，云日志服务LTS会默认为hostName、hostIP、pathFile这些内置保留字段创建字段索引。</li></ul> |

以下是一条典型日志示例，content字段值是日志原文，使用分隔符逗号将原始日志解析成3个字段level、status、message。

示例日志中的hostName、hostIP、pathFile是常见的内置保留字段，详细内置字段请参考[内置保留字段](#)。

```
{
 "hostName": "epstest-xx518",
 "hostIP": "192.168.0.31",
 "pathFile": "stdout.log",
 "content": "error,400,I Know XX",
 "level": "error",
 "status": 400,
 "message": "I Know XX"
}
```

费用说明

创建索引会产生索引流量，更多信息请参考[计费项](#)。

注意事项

- 全文索引属性和字段索引属性必须至少启用一种。
- 关闭索引后，历史索引的存储空间将在当前日志流的数据保存时间到期后，自动被清除。
- 云日志服务默认已为部分内置保留字段创建字段索引，请参见[内置保留字段](#)。
- 不同的索引配置，会产生不同的查询和分析结果，请根据您的需求，合理创建索引。全文索引和字段索引互不影响。
- 索引配置修改后，对新写入的日志数据生效，历史日志数据不会生效。
- 在字段索引功能上线前，SQL分析支持的字段来自于云端结构化解析；在字段索引功能上线后，只要用户配置了字段索引，SQL分析支持的字段将来自于字段索引，因此修改字段索引可能对现有的可视化图表、仪表盘、SQL告警、定时SQL、Grafana接入中的查询结果产生影响，请谨慎操作！

配置全文索引

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 单击目标日志组或日志流名称，进入日志流详情页面。
- 步骤3 在“日志搜索”页面，单击快速分析旁边的进入“索引配置”页面。

图 5-8 配置索引



- 步骤4 在索引配置页面中，默认开启“全文索引”按钮，参考[表5-10](#)配置各参数信息。

表 5-10 自定义全文索引配置参数

| 参数    | 说明                                                                                                                                                                                                                  | 示例 |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 全文索引  | 开启“全文索引”开关，表示创建全文索引。                                                                                                                                                                                                | 开启 |
| 大小写敏感 | 查询时是否区分英文字母的大小写。 <ul style="list-style-type: none"><li>开启“大小写敏感”开关，则查询时区分大小写。例如示例日志含有Know，那么您只能使用<b>Know</b>才能查询到该日志。</li><li>关闭“大小写敏感”开关，则查询时不区分大小写。例如示例日志含有Know，那么您使用关键字<b>KNOW</b>和<b>know</b>都能查到该日志。</li></ul> | 开启 |

| 参数   | 说明                                                                                                                                                                                                                                                                                                                                                                      | 示例                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 包含中文 | <p>查询时是否区分中英文。</p> <ul style="list-style-type: none"><li>• 开启“包含中文”开关后，如果日志中包含中文，默认按照一元分词法拆分中文内容，按照分词符的设置拆分英文内容。<ul style="list-style-type: none"><li>- 一元分词是指将中文字符串拆分为单个独立的中文字。</li><li>- 使用一元分词符的优点是对海量日志分词效率高，其他中文分词方法对写入速度影响大。</li><li>- 打开包含中文功能，会对中文使用一元分词（每个汉字单独分词），如果需要更精确的搜索结果，请用短语搜索，语法为：#"待搜索的短语"。</li></ul></li><li>• 关闭“包含中文”开关后，按照分词符的设置拆分所有内容。</li></ul> | <p>例如示例日志内容为：<br/><b>error,400,I Know 今天是星期一。</b></p> <ul style="list-style-type: none"><li>• 关闭“包含中文”开关后，按照分词符的设置拆分英文内容，日志会被拆分为<b>error、400、I、Know、今天是星期一</b>，您可以通过<b>error</b>或<b>今天是星期一</b>查找该日志。</li><li>• 打开“包含中文”开关后，日志服务后台分词器将日志拆分为<b>error、400、I、Know、今、天、是、星、期、一</b>，您通过<b>error</b>或<b>今天</b>等词都可以查找到该日志。</li></ul>                                                                                                                                                    |
| 分词符  | <p>根据指定分词符，将日志内容拆分成多个词。当默认设置不能满足您的需求时，您可以自定义设置分词符。</p> <p>如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整字符串或模糊查询查找对应的日志。</p> <p>单击“预览”，查看分词预览效果。</p>                                                                                                                                                                                                                                    | <p>例如示例日志内容为：<br/><b>error,400,I Know 今天是星期一。</b></p> <ul style="list-style-type: none"><li>• 如果不设置任何分词符，整条日志被作为一个词<b>error,400,I Know 今天是星期一</b>，您只能通过完整字符串<b>error,400,I Know 今天是星期一</b>或模糊查询<b>error,400,I K*查找该日志。</b></li><li>• 如果设置分词符为逗号（,），则原始日志被拆分为<b>error、400、I Know 今天是星期一</b>3个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如<b>error、400、I Kn*、今天是*</b>。</li><li>• 如果设置分词符为逗号（,）和空格，则原始日志被拆分为<b>error、400、I、Know、今天是星期一</b>5个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如<b>Know、今天是*</b>。</li></ul> |

| 参数    | 说明                                                 | 示例 |
|-------|----------------------------------------------------|----|
| 特殊分词符 | 单击“添加特殊分词符”，参考 <a href="#">ASCII码对照表</a> 输入ASCII值。 | -  |

- 步骤5** 完成后，单击“确定”。
- 如果没有开通管道符方式，需要使用SQL日志分析功能，请开启“日志分析”。

图 5-9 日志分析



- “日志分析”功能即将下线，建议直接使用管道符方式进行分析，详细请参考[日志搜索与分析（管道符方式）](#)。
- 开通管道符方式后，新建的日志流在索引配置页面不显示“日志分析”开关。

----结束

配置字段索引

创建字段索引时，最多支持添加500个字段。其中JSON类型字段，最多支持添加100个子字段。

- 步骤1** 设置快速分析采样条数，默认值10万条，最小值为10万条，最大值1000万条。通过采样快速统计字段值取值分布，并非对全量数据进行分析，采样条数越多分析数据越慢。
- 步骤2** 自动配置字段索引。在索引配置页面的字段索引下方，单击“自动配置”，云日志服务LTS会根据近15分钟的第一条日志内容或常见内置保留字段（例如hostIP、hostName、pathFile）自动生成字段索引，您可以根据自己的需要增加或者删除字段。
- 在索引配置页面单击“自动配置”时，默认获取最近15分钟的原始日志和内置字段的交集，LTS自动将原始日志和内置字段的交集、当前结构化字段、tag字段一起组成字段索引下方的表格数据。
  - 若15分钟内没有原始日志，则获取hostIP、hostName、pathFile、结构化字段、tag字段结合共同组成字段索引下方的表格数据。
  - ECS接入选择结构化配置时，进入索引配置页面，若没有单击“自动配置”，则会自动加上如下字段：category、hostName、hostId、hostIP、hostIPv6、pathFile，添加字段时，若某个字段已存在于索引配置，则不会重复添加。
  - CCF接入选择结构化配置时，进入索引配置页面，若没有单击“自动配置”，则会自动加上如下字段：category、clusterId、clusterName、nameSpace、podName、containerName、appName、hostName、hostId、hostIP、hostIPv6、pathFile，添加字段时，若某个字段已存在于索引配置，则不会重复添加。

**步骤3** 单个添加字段索引。在索引配置页面的字段索引下方，单击“添加字段”，配置字段索引。具体的参数配置请参考[表5-11](#)。

- 通过添加日志结构化中提取的字段，对日志数据进行统计与分析，如果您在结构化界面对字段名称进行了修改，请重新添加字段。
- 字段索引的参数配置仅对该字段生效。
- 当添加的字段在日志内容中不存在时，则配置的该索引字段无效。

**表 5-11** 自定义字段索引配置参数

| 参数   | 说明                                                                                                                                                                                                                                                                                                                                                                    |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 字段名称 | 日志字段名称，例如示例日志中的level。 <ul style="list-style-type: none"><li>• 字段名称只支持输入英文、数字、中划线、下划线及小数点，不能以小数点开头或结尾且不能以双下划线结尾。</li><li>• 日志服务默认会对部分内置保留字段开启字段索引，请参见<a href="#">内置保留字段</a>。</li><li>• 若是内置字段，在字段名称后会显示“内置”字眼，方便用户识别。</li></ul>                                                                                                                                        |
| 执行操作 | 显示字段的添加状态：新增、不修改、修改、删除。索引字段有变动后，单击“修改对比”，即可查看原配置内容与修改后配置内容的差异。 <ul style="list-style-type: none"><li>• 显示新增的字段不支持修改执行操作。</li><li>• 修改类型、别名、大小写敏感、自定义分词符、特殊分词符、包含中文、快速分析时，会与原索引配置中的字段进行对比，若任意一项不同，则执行操作变为“修改”。</li><li>• 索引配置单击确定后，不会保存执行操作为“删除”的字段。</li><li>• 在字段索引下方的搜索框，支持按字段名称、执行操作、类型搜索字段。</li><li>• 勾选多个字段后，单击“批量配置”，支持批量配置如下操作：执行操作、类型、大小写敏感、自定义分词符等。</li></ul> |
| 类型   | <ul style="list-style-type: none"><li>• 日志字段值（Value）的数据类型，可选值为string、long、float、json。</li><li>• long类型和float类型不支持设置大小写敏感、包含中文、分词符。</li></ul>                                                                                                                                                                                                                          |
| 别名   | 支持为字段名称设置别名，合理设置字段别名可以显著提高日志的可读性、可维护性和查询效率。<br>设置别名后，只支持使用别名进行SQL搜索分析，不支持使用别名进行关键字搜索。                                                                                                                                                                                                                                                                                 |



| 参数     | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 大小写敏感  | <p>查询时是否区分英文字母的大小写。</p> <ul style="list-style-type: none"><li>开启“大小写敏感”开关，则查询时区分大小写。例如示例日志message字段中含有Know，那么您只能使用<b>message:Know</b>才能查询到该日志。</li><li>关闭“大小写敏感”开关，则查询时不区分大小写。例如示例日志message字段中含有Know，那么您使用关键字<b>message:KNOW</b>和<b>message:know</b>都能查到该日志。</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 自定义分词符 | <p>根据指定分词符，将日志内容拆分成多个词。当默认设置不能满足您的需求时，您可以自定义设置分词符。</p> <p>如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整字符串或模糊查询查找对应的日志。</p> <p>例如示例日志message字段内容为：<b>I Know 今天是星期一。</b></p> <ul style="list-style-type: none"><li>如果不设置任何分词符，整条日志被作为一个词<b>I Know 今天是星期一</b>，您只能通过完整字符串<b>message:I Know 今天是星期一</b>或模糊查询<b>message:I Know 今天是*</b>查找该日志。</li><li>如果设置分词符为空格，则原始日志被拆分为<b>I</b>、<b>Know</b>、<b>今天是星期一</b>3个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如<b>message:Know</b>或<b>message:今天是星期一</b>。</li></ul>                                                                                                                                                                                                                                                                                         |
| 特殊分词符  | <p>单击“添加特殊分词符”，参考<a href="#">ASCII码对照表</a>输入ASCII值。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 包含中文   | <p>查询时是否区分中英文。</p> <ul style="list-style-type: none"><li>开启“包含中文”开关后，如果日志中包含中文，默认按照一元分词法拆分中文内容，按照分词符的设置拆分英文内容。<ul style="list-style-type: none"><li>一元分词是指将中文字符串拆分为单个独立的中文字。</li><li>使用一元分词符的优点是对海量日志分词效率高，其他中文分词方法对写入速度影响大。</li><li>打开包含中文功能，会对中文使用一元分词（每个汉字单独分词），如果需要更精确的搜索结果，请用短语搜索，语法为：<b>#"待搜索的短语"</b>。</li></ul></li><li>关闭“包含中文”开关后，按照分词符的设置拆分所有内容。</li></ul> <p>例如示例日志message字段内容为：<b>I Know 今天是星期一。</b></p> <ul style="list-style-type: none"><li>关闭“包含中文”开关后，按照分词符的设置拆分英文内容，日志会被拆分为<b>I</b>、<b>Know</b>、<b>今天是星期一</b>，您可以通过<b>message:Know</b>或<b>message:今天是星期一</b>查找该日志。</li><li>开启“包含中文”开关后，日志服务后台分词器将日志拆分为<b>I</b>、<b>Know</b>、<b>今</b>、<b>天</b>、<b>是</b>、<b>星</b>、<b>期</b>、<b>一</b>，您通过<b>message:Know</b>或<b>message:今天</b>等词都可以查找到该日志。</li></ul> |

| 参数   | 说明                                                                                                                                                                                                  |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 快速分析 | <p>默认为开启状态，开启后，可以对字段值做采样统计，请参见<a href="#">创建快速分析</a>。</p> <ul style="list-style-type: none"><li>快速分析的原理是对搜索命中的日志采样10万条进行数据统计，不是全量统计。</li><li>快速分析的字段长度最大为2000字节。</li><li>快速分析字段展示前100条数据。</li></ul> |
| 操作   | 单击  ，删除添加的自定义字段。                                                                                                   |

图 5-10 批量配置



步骤4 完成后，单击“确定”。

----结束

内置保留字段

在采集日志时，云日志服务会将采集时间、日志类型、主机IP等信息以Key-Value对的形式添加到日志中，这些字段是云日志服务的内置字段。

- 使用API写入日志数据或添加ICAgent配置时，请不要将字段名称设置为内置保留字段，否则可能会造成字段名称重复、查询不精确等问题。
- 云日志服务为日志数据增加的内置保留字段当前免费。更多信息请参见[价格计算器](#)。
- 用户自定义日志字段名称中不能使用双下划线\_\_，否则无法配置索引。

表 5-12 内置保留字段说明

| 内置保留字段      | 数据格式           | 索引与统计设置                                                                                   | 说明                                                                                                          |
|-------------|----------------|-------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| collectTime | 整型，Unix时间戳（毫秒） | <p>索引设置：开启索引后，日志服务默认为collectTime创建字段索引，索引数据类型为long类型。</p> <p>查询时输入 collectTime : xxx。</p> | <p>采集时间，指日志被采集器ICAgent采集时的时间。</p> <p>例如示例中的 "collectTime": "1681896081334"，转换成标准时间是 2023-04-19 17:21:21</p> |

| 内置保留字段      | 数据格式           | 索引与统计设置                                                                           | 说明                                                                                                                                |
|-------------|----------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| __time__    | 整型，Unix时间戳（毫秒） | 索引设置：开启索引后，日志服务默认为time创建字段索引，索引数据类型为long类型。该字段不支持查询。                              | 日志时间，指的是日志在控制台页面展示的日志时间。<br>例如示例中的"__time__":"1681896081334"，转换成标准时间是2023-04-19 17:21:21<br>日志时间默认使用采集时间，也支持自定义日志时间。              |
| lineNum     | 整型             | 索引设置：开启索引后，日志服务默认为lineNum创建字段索引，索引数据类型为long类型。                                    | 行号（偏移量），用来排序日志。<br>非高精度日志会根据collectTime生成，默认是collectTime * 1000000 + 1，高精度日志就是用户上报的纳秒值。<br>例如示例中的"lineNum":"1681896081333991900"。 |
| category    | 字符串            | 索引设置：开启索引后，日志服务默认为category创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入category: xxx。       | 日志类型，表示该日志的来源。<br>例如ICAgent采集的日志该字段为LTS，某云服务例如DCS上报的日志该字段为DCS。                                                                    |
| clusterName | 字符串            | 索引设置：开启索引后，日志服务默认为clusterName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入clusterName: xxx。 | 集群名称，k8s场景下集群名称。<br>例如示例中的"clusterName":"epstest"。                                                                                |
| clusterId   | 字符串            | 索引设置：开启索引后，日志服务默认为clusterId创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入clusterId: xxx。     | 集群ID，k8s场景下集群ID。例如示例中的"clusterId":"c7f3f4a5-xxxx-11ed-a4ec-0255ac100b07"。                                                         |

| 内置保留字段        | 数据格式 | 索引与统计设置                                                                               | 说明                                                                            |
|---------------|------|---------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| nameSpace     | 字符串  | 索引设置：开启索引后，日志服务默认为nameSpace创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入nameSpace: xxx。         | 命名空间，k8s场景下命名空间。<br>例如示例中的"nameSpace":"monitoring"。                           |
| appName       | 字符串  | 索引设置：开启索引后，日志服务默认为appName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入appName: xxx。             | 组件名称，k8s场景下工作负载的名称。<br>例如示例中的"appName":"alertmanager-alertmanager"。           |
| serviceID     | 字符串  | 索引设置：开启索引后，日志服务默认为serviceID创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入serviceID: xxx。         | 工作负载ID，k8s场景下工作负载ID。<br>例如示例中的"serviceID":"cf5b453xxxad61d4c483b50da3fad5ad"。 |
| podName       | 字符串  | 索引设置：开启索引后，日志服务默认为podName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入podName: xxx。             | POD名称，k8s场景下POD名称。<br>例如示例中的"podName":"alertmanager-alertmanager-0"。          |
| podIp         | 字符串  | 索引设置：开启索引后，日志服务默认为podIp创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入podIp: xxx。                 | pod的ip，k8s场景下pod的IP地址。<br>例如示例中的"podIp":"10.0.0.145"。                         |
| containerName | 字符串  | 索引设置：开启索引后，日志服务默认为containerName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入containerName: xxx。 | 容器名称，k8s场景下容器名称。<br>例如示例中的"containerName":"config-reloader"。                  |

| 内置保留字段   | 数据格式 | 索引与统计设置                                                                     | 说明                                                                                       |
|----------|------|-----------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| hostName | 字符串  | 索引设置：开启索引后，日志服务默认为hostName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostName: xxx。 | 主机名称，ICAgent所在主机的名称。<br>例如示例中的"hostName":"epstest-xx518"。                                |
| hostId   | 字符串  | 索引设置：开启索引后，日志服务默认为hostId创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostId: xxx。     | 主机ID，ICAgent所在主机的id，该id由ICAgent生成。例如示例中的"hostId":"318c02fe-xxxx-4c91-b5bb-6923513b6c34"。 |
| hostIP   | 字符串  | 索引设置：开启索引后，日志服务默认为hostIP创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostIP: xxx。     | 主机IP，日志采集器所在主机的ip（适用于ipv4场景）<br>例如示例中的"hostIP":"192.168.0.31"。                           |
| hostIPv6 | 字符串  | 索引设置：开启索引后，日志服务默认为hostIPv6创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostIPv6: xxx。 | 主机IP，日志采集器所在主机的ip（适用于ipv6场景）<br>例如示例中的"hostIPv6":""。                                     |
| pathFile | 字符串  | 索引设置：开启索引后，日志服务默认为pathFile创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入pathFile: xxx。 | 文件路径，采集的日志文件的路径。<br>例如示例中的"pathFile":"stdout.log"。                                       |
| content  | 字符串  | 索引设置：开启全文索引后，会使用全文索引定义的分词符对content字段的value进行分词；不支持将content字段配置到字段索引中。       | 日志原文，例如示例中的"content":"level=error<br>ts=2023-04-19T09:21:21.333895559Z"                  |

| 内置保留字段               | 数据格式           | 索引与统计设置                                                                                                | 说明                            |
|----------------------|----------------|--------------------------------------------------------------------------------------------------------|-------------------------------|
| __receive_time__     | 整型，Unix时间戳（毫秒） | 索引设置：开启索引后，日志服务默认为__receive_time__创建字段索引，索引数据类型为long类型。                                                | 上报日志的服务端时间，相当于LTS采集端接收到日志的时间。 |
| __client_time__      | 整型，Unix时间戳（毫秒） | 索引设置：开启索引后，日志服务默认为__client_time__创建字段索引，索引数据类型为long类型。                                                 | 端侧日志的客户端上报时间。                 |
| _content_parse_fail_ | 字符串            | 索引设置：开启索引后，日志服务默认为_content_parse_fail_创建字段索引，索引数据类型为string类型，分词字符为默认分词符。查询时输入_content_parse_fail_:xxx。 | 上报日志解析失败的日志内容。                |
| __time               | 整型，Unix时间戳（毫秒） | 不支持将__time字段配置到字段索引中。                                                                                  | 不涉及。                          |
| logContent           | 字符串            | 不支持将logContent字段配置到字段索引中。                                                                              | 不涉及。                          |
| logContentSize       | 整型             | 不支持将logContentSize字段配置到字段索引中。                                                                          | 不涉及。                          |
| logIndexSize         | 整型             | 不支持将logIndexSize字段配置到字段索引中。                                                                            | 不涉及。                          |
| groupName            | 字符串            | 不支持将groupName字段配置到字段索引中。                                                                               | 不涉及。                          |
| logStream            | 字符串            | 不支持将logStream字段配置到字段索引中。                                                                               | 不涉及。                          |

ASCII 码对照表

表 5-13 ASCII 码对照表

| AS<br>CII<br>值 | 控制字符                   | ASC<br>II值 | 控制字符 | AS<br>CII<br>值 | 控制字符 | AS<br>CII<br>值 | 控制字符 |
|----------------|------------------------|------------|------|----------------|------|----------------|------|
| 0              | NUL（空字<br>符）           | 32         | 空格   | 64             | @    | 96             | `    |
| 1              | SOH（标题开<br>始）          | 33         | !    | 65             | A    | 97             | a    |
| 2              | STX（正文开<br>始）          | 34         | "    | 66             | B    | 98             | b    |
| 3              | ETX（正文结<br>束）          | 35         | #    | 67             | C    | 99             | c    |
| 4              | EOT（传输结<br>束）          | 36         | \$   | 68             | D    | 100            | d    |
| 5              | ENQ（询问字<br>符）          | 37         | %    | 69             | E    | 101            | e    |
| 6              | ACK（确认回<br>应）          | 38         | &    | 70             | F    | 102            | f    |
| 7              | BEL（响铃）                | 39         | '    | 71             | G    | 103            | g    |
| 8              | BS（退格）                 | 40         | (    | 72             | H    | 104            | h    |
| 9              | HT（水平定位<br>符号，制表<br>符） | 41         | )    | 73             | I    | 105            | i    |
| 10             | LF（换行）                 | 42         | *    | 74             | J    | 106            | j    |
| 11             | VT（垂直定位<br>符号）         | 43         | +    | 75             | K    | 107            | k    |
| 12             | FF（换页键）                | 44         | ,    | 76             | L    | 108            | l    |
| 13             | CR（归位键）                | 45         | -    | 77             | M    | 109            | m    |
| 14             | SO（取消变<br>换）           | 46         | .    | 78             | N    | 110            | n    |
| 15             | SI（启用变<br>换）           | 47         | /    | 79             | O    | 111            | o    |
| 16             | DLE（跳出数<br>据通讯）        | 48         | 0    | 80             | P    | 112            | p    |
| 17             | DC1（设备控<br>制1）         | 49         | 1    | 81             | Q    | 113            | q    |

| AS CII 值 | 控制字符        | ASC II 值 | 控制字符 | AS CII 值 | 控制字符 | AS CII 值 | 控制字符    |
|----------|-------------|----------|------|----------|------|----------|---------|
| 18       | DC2（设备控制2）  | 50       | 2    | 82       | R    | 114      | r       |
| 19       | DC3（设备控制3）  | 51       | 3    | 83       | S    | 115      | s       |
| 20       | DC4（设备控制4）  | 52       | 4    | 84       | T    | 116      | t       |
| 21       | NAK（确认失败回应） | 53       | 5    | 85       | U    | 117      | u       |
| 22       | SYN（同步用暂停）  | 54       | 6    | 86       | V    | 118      | v       |
| 23       | ETB（区块传输结束） | 55       | 7    | 87       | W    | 119      | w       |
| 24       | CAN（取消）     | 56       | 8    | 88       | X    | 120      | x       |
| 25       | EM（连接介质中断）  | 57       | 9    | 89       | Y    | 121      | y       |
| 26       | SUB（替换）     | 58       | :    | 90       | Z    | 122      | z       |
| 27       | ESC（跳出）     | 59       | ;    | 91       | [    | 123      | {       |
| 28       | FS（文件分隔符）   | 60       | <    | 92       | \    | 124      |         |
| 29       | GS（组群分隔符）   | 61       | =    | 93       | ]    | 125      | }       |
| 30       | RS（记录分隔符）   | 62       | >    | 94       | ^    | 126      | ~       |
| 31       | US（单元分隔符）   | 63       | ?    | 95       | _    | 127      | DEL（删除） |

相关文档

- 创建LTS日志索引后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- LTS支持通过调用API接口创建日志流索引，详细请参考[创建日志流索引](#)。
- LTS支持通过调用API接口查询日志流索引，详细请参考[查询日志流索引](#)。

5.4 搜索与分析日志

云日志服务支持管道符特性在一个语句中同时进行搜索和分析。其语法结构主要由三部分构成：针对非结构化数据和半结构化数据的搜索语句、管道符“|”和针对结构化

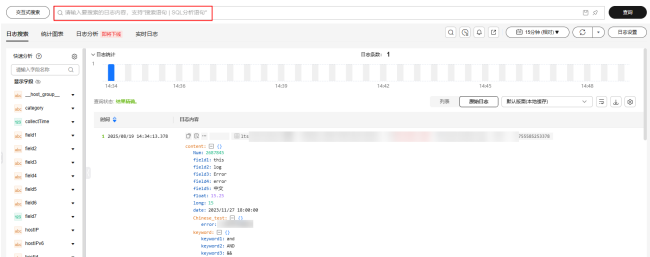


数据的查询的分析语句。语法示例结构：`* and msg:"hello world" | SELECT avg(value)`

日志搜索分析

- 步骤1 在云日志服务管理控制台，单击“日志管理”。
- 步骤2 在日志组列表中，单击日志组名称前对应的  按钮。
- 步骤3 在日志流列表中，单击日志流名称，进入日志详情页面。

图 5-11 日志搜索页面



- 步骤4 在日志内容下方，支持选择列表或原始日志方式展示日志内容。

日志高亮的实现原理是当查询条件命中日志后，再将日志与查询条件进行字符串匹配，相同日志部分增加高亮标签，该部分会在界面高亮展示。因此当查询条件比较复杂，尤其是存在或关系时，在界面高亮显示的内容可能会比实际查询结果显示的多。

在不上传原始日志的情况下，全文搜索中文时，可能会出现中文不高亮的情况，如果需要高亮字段（比如：field1）内容中的中文，请使用“field1: 中文关键词”形式。

- 步骤5 在“日志搜索”页签，查看快速分析下方的结构化字段和索引配置字段，可以直接单击对应字段后面的  图表，查看指标信息，详细操作请参考[创建快速分析](#)。

图 5-12 结构化字段



- 步骤6** 在日志搜索页面，当搜索数据量过大时，LTS开始执行自动迭代查询日志。
- 自动迭代查询日志过程中，搜索框不支持输入，您可以单击“暂停查询”中断自动查询过程，若需要继续查询日志，单击“继续查询”恢复自动查询。建议您等待当前自动迭代查询完成后再次发起新的查询任务。
  - 对于较长时间的迭代查询，建议您缩小时间范围或添加过滤条件减少迭代次数。
- 步骤7** 在“统计图表”页签，在右上角选择时间范围，可以查看原始日志和统计图表。在搜索框由搜索语句和SQL分析语句组成，两者通过管道符|联动。详细请参考[SQL函数](#)。

图 5-13 时间范围



- 时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。
- 相对时间：**表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
  - 整点时间：**表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
  - 自定义：**表示查询指定时间范围的日志数据。

- 步骤8** 支持设置日志数据的版面展示，主要设置字段是否显示或简化显示。
- 在下拉框单击编辑版面，进入版面设置页面，版面列表自带默认版面、纯净版面、容器日志默认版面，可以设置字段在版面是否显示。

图 5-14 编辑版面



- 云端：适用于有写权限的用户，版面配置信息保存在云端。支持设置账号级别或用户级别的版面配置。
  - 本地缓存：适用于只有读权限的用户，版面配置信息缓存在本地浏览器。
2. 在版本设置页面的版本列表下方，单击“+”新增自定义版面，设置版面名称和版面字段的可见性。

图 5-15 版面设置



新创建的版面可以进行如下操作：

- 编辑：单击  可以编辑版面的显示名称。
- 复制：鼠标悬浮  上，勾选复制到本地缓存，或者勾选云端-账号级别或云端-用户级别，单击“确定”即可复制版面信息。
- 删除：单击  ，单击“确定”即可删除版面信息。

 **警告**

删除版面信息后无法恢复，请谨慎操作。

3. 设置完成后，单击“确定”，返回下拉框显示新增的自定义版面。

**步骤9** 在“日志搜索”页面，显示列设置。

1. 选择列表展示日志时，鼠标悬浮在  按钮上，单击“显示列设置”，在弹出的日志折叠设置页面中，设置显示列的信息。
2. 时间列、content默认开启显示，在显示列下方，全量显示可见的字段名称，支持设置字段名称的别名，合理设置字段别名可以显著提高日志的可读性、可维护性和查询效率。设置完成后，单击“确定”。
3. 您可以通过[创建快速分析](#)或[步骤8](#)显示或隐藏字段。

----结束

## 交互模式

使用交互模式之前，请确保日志正常上报，且已经完成结构化分析和索引配置，详细请参考[使用ICAgent插件采集主机日志](#)和[创建LTS日志索引](#)。

若用户在日志接入LTS时没有配置ICAgent结构化解析，可以单独给目标日志流配置ICAgent结构化解析或云端结构化解析。推荐使用ICAgent结构化解析的方式，更多内容请参考[配置ICAgent结构化解析](#)。

交互式搜索分析适用于生成简单的分析语句，操作简单，通过在界面上设置搜索条件和指定日志查询时的过滤规则，从而筛选日志中满足条件的记录。如果您需要使用更多的函数或者嵌套查询，请手动输入SQL语句。详细请参考[SQL函数](#)。

**步骤1** 单击搜索框前面的“交互式搜索”，进入交互式搜索分析页面，支持设置日志搜索和日志分析。

图 5-16 交互模式



**步骤2** 在“日志搜索”下方的下拉框选择日志搜索的字段和条件，搜索框则会展示对应字段的值。根据业务需要，用户可以自定义设置搜索方式，添加关联关系或添加组进行搜索。

- 下拉框显示的字段为索引配置字段、结构化字段、[内置保留字段](#)。
- 且（AND）表示在多个条件中需要同时满足所有条件才能成立。
- 或（OR）表示在多个条件中只需满足其中一个条件即可。

例如在界面上选择content、hostId、pathFile字段，设置不同的条件，随时可以预览搜索语句，如有问题，可以修改搜索条件，操作简单。

- 步骤3** 在日志分析下方，设置分析规则。
- 单击添加指标，选择统计函数作用于选定的字段，计算出用户想要的指标。
  - 分组排序是先基于用户选择的字段分组(group by)，然后分组执行指标统计，最后对结果进行排序(order by)。
- 分别单击添加分组和添加排序，设置字段的分组信息和排序。
- 步骤4** 您可以根据需要设置不同的条件，随时可以预览搜索语句，如有问题，随时修改搜索条件，操作简单。设置完成后，单击“确定”，在“统计图表”页面，即可查看分析结果。更多信息请参考[使用统计图表将日志可视化](#)。
- 结束

日志搜索的常用操作

日志搜索的常用操作有分享日志、刷新等操作，具体参考[表5-14](#)。

表 5-14 常用操作

| 操作     | 说明                                                                                                                                                                                                                                                        |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 创建快速查询 | 单击  按钮，创建快速查询。                                                                                                                                                           |
| 查看仪表盘  | 单击  按钮，在弹出来的仪表盘页面中，可查看已创建的仪表盘。                                                                                                                                          |
| 添加告警   | 单击  按钮，在弹出的页面中，支持 <a href="#">新建告警规则</a> 。                                                                                                                             |
| 分享日志   | 单击  复制当前日志搜索页面的链接，用于分享搜索日志。                                                                                                                                            |
| 刷新日志   | 单击  对日志进行刷新，有两种方式刷新方式：手动刷新和自动刷新。 <ul style="list-style-type: none"><li>手动刷新：单击“手动刷新”，可直接对日志进行刷新。</li><li>自动刷新：选择自动刷新的间隔时间，将对日志进行自动刷新。间隔时间范围为15秒、30秒、1分钟和5分钟。</li></ul> |
| 复制     | 单击  复制日志内容。                                                                                                                                                            |
| 查看上下文  | 单击  查看日志上下文。<br>支持选择简洁模式查看日志上下文。支持下载上下文内容。                                                                                                                             |

| 操作      | 说明                                                                                                                                                                                                                                                                                                                          |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 更多操作    | <p>单击  进入该时间段的日志详情页，查看更多日志信息。</p> <ul style="list-style-type: none"><li>在“扩展字段”页签，查看字段名称、字段值，在对应字段操作列，支持对该字段进行添加到查询、从查询中排除、字段存在、字段不存在、隐藏的方式搜索日志。</li><li>在“json格式”页签，查看日志的json格式。</li><li>在“上下文日志”页签，支持设置查询行数、过滤字段、下载日志、简洁模式等操作。</li></ul> |
| 换行/取消换行 | <p>单击  按钮，搜索的日志内容将换行显示，默认开启换行按钮。若不需要换行，单击  按钮，取消换行。</p>                                                                                                  |

| 操作        | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 下载日志      | <p>支持将日志文件或日志查询分析结果下载到本地进行查看。<br/>请确保当前日志流上传了原始日志，否则下载后文件将没有内容。</p> <p>1. 单击  按钮。</p> <p>2. 单击“下载日志”，在弹出的下载日志页面中支持前端直接下载、后台离线下线和前往创建转储。开通一次性转储后才能使用后台离线下线和前往创建转储。</p> <p><b>说明</b><br/>一次性转储功能仅支持白名单用户使用，如有需要，请<a href="#">提交工单</a>申请开通。</p> <ul style="list-style-type: none"><li>前端直接下载：直接将前端查询的结果保存到本地，下载记录不会出现在日志下载历史中。单次下载支持最大5,000条日志。<br/>勾选“.csv”或“.txt”格式，单击“开始下载日志”，将日志导出至本地。</li><li>后台离线下线：通过后台任务将日志文件下载到临时OBS桶中，您的浏览器需要有公网访问权限才能在日志下载历史中下载结果文件。单次下载支持最大2,000万条日志。<br/>勾选“.csv”或“.txt”或“.json”，单击“开始下载日志”，将日志导出至本地。<br/>日志转储成功后，云日志服务根据您的日志转储量进行收费。转储费用详情请参见<a href="#">LTS价格详情</a>。</li><li>前往创建转储：通过OBS转储任务下载日志文件，单次下载支持最大2,000万条日志。<br/>单击“前往创建转储”，跳转至配置转储页面创建一次性转储任务，详细请参考<a href="#">配置日志转储（一次性）</a>。<br/>日志转储成功后，云日志服务根据您的日志转储量进行收费。转储费用详情请参见<a href="#">LTS价格详情</a>。</li></ul> <p>3. 单击“日志下载历史”，在日志下载历史页面，支持查看、下载、复制日志下载链接、删除日志下载记录。<br/>加载完成的日志文件只在“日志下载历史”页面保留1天，请及时下载到本地保存。</p> |
| 全部折叠/全部展开 | <p>单击  设置日志内容展示的行数。若不需要展示日志内容，再</p> <p>单击一次  按钮即可关闭展示的日志内容。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

| 操作                                                                                             | 说明                                                                                                                                                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| JSON设置                                                                                         | <p>鼠标悬浮在按钮上，单击“JSON设置”，在弹出的JSON设置页面中，设置格式化显示。</p> <p>默认开启格式化，JSON默认展开层级为2层。若日志包含多个反斜杠，当日志展示为json格式时，会丢失一个反斜杠，因为json解析会将第一个反斜杠作为转义符处理。</p> <ul style="list-style-type: none"><li>开启格式化按钮：设置JSON默认展开层级，最大设置为10层。</li><li>关闭格式化按钮：对于JSON格式的日志，将不会格式化层级显示。</li></ul> |
| 日志折叠设置                                                                                         | <p>鼠标悬浮在按钮上，单击“日志折叠设置”，在弹出的日志折叠设置页面中，设置长日志字符个数。</p> <p>日志超过设置的长日志字符个数时，超出字符将被隐藏，单击“展开”按钮可查看全部内容。</p> <p>默认开启自动折叠长日志，字符个数默认为400个。</p>                                                                                                                             |
| 日志时间展示                                                                                         | <p>鼠标悬浮在按钮上，单击“日志时间展示”，在弹出的日志折叠设置页面中，默认开启展示毫秒，设置是否展示毫秒、是否展示时区。</p>                                                                                                                                                                                               |
| 不可见字段列表<br> | <p>该列表展示版面设置中配置的不可见性字段。</p> <ul style="list-style-type: none"><li>当日志流未配置版面设置时，将不显示按钮。</li><li>当日志内容为“CONFIG_FILE”且未配置版面设置时，不可见字段默认有appName、clusterId、clusterName、containerName、hostIPv6、NameSpace、podName和serviceID。</li></ul>                              |

## 相关文档

- 日志接入LTS成功后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- 如果在搜索日志时遇到问题，您可以参考[日志搜索与分析常见问题](#)获取解决方案。
- 为您推荐几种在真实使用场景下日志搜索与分析的实践教程。更多信息请参考[日志搜索与分析最佳实践](#)。
  - 云日志服务LTS支持分页显示查询分析结果，可控制每次返回的日志数量。您可以参考[分页显示查询分析结果](#)。
  - WAF接入LTS后，支持通过LTS分析WAF日志。您可以参考[在LTS页面分析华为云WAF日志](#)。
  - LTS支持将日志查询界面嵌入到客户自建系统。您可以参考将[LTS日志查询页面嵌入用户自建系统](#)。



## 5.5 搜索语法介绍

云日志服务LTS提供一套搜索语法用于设置搜索条件，帮助您更有效地搜索日志。

查询语句用来指定日志查询时的过滤规则，返回符合条件的日志。根据索引配置方式可分为全文查询和字段查询，根据查询精确程度可分为精确查询和模糊查询。详细请参考[表5-15](#)。

SQL是用于访问和处理数据库的标准计算机语言。管道符提供了使用SQL语句进行数据分析的能力，具体请参考SQL92语法标准。

使用SQL语句进行查询时，建议SQL语句中的字段名或别名使用双引号，例如：select "filed1.a" as "别名" from log。

表 5-15 查询方式

| 查询方式 | 说明                                                                                                                                       | 示例                                                     |
|------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| 全文查询 | 配置全文索引后，日志服务根据您设置的分词符将整条日志拆分成多个词。您可以指定关键字（字段名、字段值）和查询规则进行查询。                                                                             | hello and world表示查询同时包含关键字hello和world的日志。              |
| 字段查询 | 配置字段索引后，您可以指定字段名称和字段值（Key:Value）进行查询。根据字段索引中设置的数据类型，您可以进行多种类型的基础查询和组合查询。                                                                 | time>60 and region:r*表示查询time字段值大于60且region字段值以r开头的日志。 |
| 精确查询 | 使用完整的词进行查询。日志服务查询采用的是分词法，精确查询时并不能完全匹配关键词。例如查询语句为abc def，查询结果将包含所有abc和def的日志，无法完全匹配目标短语。如果您要完全匹配短语abc def，可以使用短语查询。更多信息，请参见短语查询、如何精准查询日志。 | region:r1表示查询region字段值精确匹配r1的日志。                       |

| 查询方式 | 说明                                                                                                                                                                                                                                                                                                                                                                                                               | 示例                                    |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| 模糊查询 | <p>在查询语句中指定一个64个字符以内的词，在词的中间或者末尾加上模糊查询关键字，即星号（*）或问号（?），日志服务会在所有日志中为您查询到符合条件的100个词，返回包含这100个词并满足查询条件的所有日志。指定的词越精确，查询结果越精确。</p> <ul style="list-style-type: none"><li>星号（*）代表匹配多个字符，问号（?）代表匹配1个字符。</li><li>当星号（*）和问号（?）作为分词符时，不具备模糊搜索功能，其中问号（?）为默认的分词符，使用其模糊搜索功能前需将其从分词符中移除。</li><li>星号（*）或问号（?）不能用在词的开头。</li><li>long数据类型和float数据类型不支持使用星号（*）或问号（?）进行模糊搜索。</li><li>当模糊条件前缀很短且日志中符合条件的词超过100个时，查询结果会不精确。</li></ul> | my*表示在所有日志中查找以my开头的100个词，并返回包含这些词的日志。 |

- 分词符  
云日志服务LTS会根据分词符，将日志内容拆分成多个词。日志服务默认配置的分词符为, ";=() []{}@&<>/:\\?\\n\\t\\r。  
例如：日志2023-01-01 09:30:00，默认分词符会将其分为四部分：  
2023-01-01、09、30、00。  
此时搜索语句2023无法匹配到该条日志，可以通过2023-01\*或2023-01-01搜索到该条日志。  
如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整日志内容或模糊搜索查找对应的日志。
- 关键词顺序  
只有短语搜索#"abc def"才能保证关键词出现的顺序，其他搜索方式多个关键词默认AND连接。  
例如：request\_method:GET POST查询的是同时包含GET和POST的日志，不会保证GET和POST的顺序。如有需要推荐采用短语搜索。
- 中文搜索  
中文搜索时不需要采用模糊查询，如有需要推荐采用短语搜索，可以匹配到更精确的结果。  
云日志服务LTS的英文是以单词的形式进行拆分的，单词的长度不一致，因此可以通过模糊搜索匹配拥有相同前缀英文单词的日志。  
中文采用的是一元分词，每个字都是独立的，拆分后每部分的长度都是1。  
例如：搜索语句星期一，代表搜索同时包含星、期、一的日志；搜索语句#"星期一"，代表搜索包含目标短语星期一的日志。

- 语法关键词  
日志搜索语句的语法关键词包括：&& || AND OR and or NOT not in : > < = ( ) [ ]  
其中 **and AND or OR NOT not in** 作为语法关键词使用时，前后需要使用空格分隔；  
如果日志中本身包含语法关键词且需要搜索时，搜索语句需要用**双引号**包裹，否则可能会导致语法错误或搜索到错误的结果。  
例如：搜索语句**content:and**, 包含语法关键词 **and** ,需要修改为 **content:"and"**。

短语搜索

短语搜索用于准确匹配目标短语，例如搜索语句**abc def**，不区分先后顺序，将匹配所有同时包含**abc**和**def**的日志。短语搜索和关键词搜索的区别请参考表5-16。

- 短语搜索：在关键词搜索语法的基础上实现，短语搜索能够区分关键词的顺序，用于精准匹配目标短语，搜索结果更加精确。短语搜索适用于英文短语、中文短语的搜索，不支持模糊搜索。
- 关键词搜索：关键词搜索是基于分词实现，通过分词符先将搜索内容拆分为多个关键词，然后匹配日志。关键词搜索不会区分多个关键词在日志中出现的顺序，因此只要日志中按照搜索的与或非逻辑能命中关键词，该日志就会被搜索到。

表 5-16 搜索区别

| 搜索方式 | 短语搜索                                                                                                                            | 关键词搜索                    |
|------|---------------------------------------------------------------------------------------------------------------------------------|--------------------------|
| 搜索区别 | 区分关键词的顺序，用于精准匹配目标短语，搜索结果更加精确。                                                                                                   | 不区分关键词的顺序，按照搜索逻辑命中关键词即可。 |
| 举例说明 | 假设您的日志流中存在2条原始日志，如下： <ul style="list-style-type: none"><li>● 原始日志1：this service is lts</li><li>● 原始日志2：lts is service</li></ul> |                          |
|      | 短语搜索：#"is lts"，会命中1条日志。                                                                                                         | 关键词搜索：is lts，会命中2条日志。    |
|      | 短语搜索：#"lts is"，会命中1条日志。                                                                                                         | 关键词搜索：lts is，会命中2条日志。    |

表 5-17 搜索方式

| 搜索方式 | 说明                                                                                                                                                                  |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 全文搜索 | <ul style="list-style-type: none"><li>● #"abc def"</li><li>● content:#"abc def"</li></ul> content为日志原文对应的内置字段，#"abc def"等同于 <b>content:#"abc def"</b> ，默认匹配日志原文的内容。 |

| 搜索方式 | 说明                                                                                                                   |
|------|----------------------------------------------------------------------------------------------------------------------|
| 字段搜索 | key:"#abc def" <ul style="list-style-type: none"><li>value参数不可为空。</li><li>字段搜索和not运算符配合使用时，还会匹配到不包含该字段的日志。</li></ul> |

- 短语搜索不支持搭配模糊搜索。  
短语搜索中的星号（\*）和问号（?）会被视为普通字符，因此短语搜索不支持搭配模糊搜索，可以用来搜索日志中的星号（\*）和问号（?）。
- 短语搜索不支持对分词符进行搜索。  
例如搜索语句**#"var/log"**，其中 / 为分词符，搜索语句等同于**#"var log"**，会搜索包含目标短语**var log**的日志。同理，搜索语句**#"var:log"**、**#"var;log"**等搜索的也是包含目标短语**var log**的日志。
- 中文搜索推荐采用短语搜索。  
由于中文默认采用的是一元分词，每个汉字单独分词，搜索时会匹配同时包含搜索语句中每一个汉字的日志，本身便具有模糊搜索的特性，当需要更加精确的结果时，推荐采用短语搜索。

运算符

查询语句支持如下[运算符](#)。

表 5-18 运算符

| 运算符 | 说明               | 示例                                                                                                       |
|-----|------------------|----------------------------------------------------------------------------------------------------------|
| and | and运算符。          | request_method:GET and status:200                                                                        |
| AND | 与运算符，等同于and。     | request_method:GET AND status:200                                                                        |
| or  | or运算符。           | request_method:GET or status:200                                                                         |
| OR  | 或运算符，等同于or。      | request_method:GET OR status:200                                                                         |
| not | not运算符。          | <ul style="list-style-type: none"><li>request_method:GET not status:200</li><li>not status:200</li></ul> |
| ( ) | 用于提高括号内查询条件的优先级。 | (request_method:GET or request_method:POST) and status:200                                               |

| 运算符 | 说明                                                                                                                                                                                                                                                                        | 示例                                                                                                              |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| :   | 用于字段查询（Key:Value）。<br>如果字段名称（key）或者字段值（value）内有空格或冒号（:）等保留字符，请使用双引号（" "）包裹字段名称（key）或者字段值（value）。例如： <ul style="list-style-type: none"><li>• "request method":GET</li><li>• message:"This is a log"</li><li>• time:"09:00:00"</li><li>• ipv6:"2024:AC8:2ac::d09"</li></ul> | request_method:GET                                                                                              |
| " " | 使用双引号（" "）包裹一个语法关键词，可以将该语法关键词转换成普通字符。例如"and"表示查询包含and的日志，此处的and不代表运算符。在字段查询中双引号（" "）内的所有词被当成一个整体。                                                                                                                                                                         | request_method:"GET"                                                                                            |
| \   | 转义符号，用于转义双引号（" "），转义后的引号表示符号本身。<br>转义双引号\"在搜索语句中必须成对出现。例如支持使用fieldName : \"error\"或fieldName:\"\\\"，不支持使用fieldName : "error\"或者fieldName : "error\"                                                                                                                       | 例如日志内容为instance_id:nginx"01"，您可以使用instance_id:nginx\"01\"进行查询。                                                  |
| *   | 通配符查询，匹配零个、单个、多个字符。                                                                                                                                                                                                                                                       | host:ecs*c                                                                                                      |
| ?   | 通配符查询，匹配单个字符。                                                                                                                                                                                                                                                             | host:ecs?c                                                                                                      |
| >   | 查询某字段值大于某数值的日志。                                                                                                                                                                                                                                                           | request_time>100                                                                                                |
| >=  | 查询某字段值大于或等于某数值的日志。                                                                                                                                                                                                                                                        | request_time>=100                                                                                               |
| <   | 查询某字段值小于某数值的日志。                                                                                                                                                                                                                                                           | request_time<100                                                                                                |
| <=  | 查询某字段值小于或等于某数值的日志。                                                                                                                                                                                                                                                        | request_time<=100                                                                                               |
| =   | 查询某字段值等于某数值的日志。针对double、long类型的字段，等号（=）和冒号（:）作用相同。                                                                                                                                                                                                                        | 例如request_time=100等同于request_time:100。                                                                          |
| in  | 查询某字段值处于某数值范围内的日志，中括号表示闭区间，小括号表示开区间，两个数字之间使用空格分隔。                                                                                                                                                                                                                         | <ul style="list-style-type: none"><li>• request_time in [100 200]</li><li>• request_time in (100 200]</li></ul> |

| 运算符 | 说明                                                                                                | 示例                            |
|-----|---------------------------------------------------------------------------------------------------|-------------------------------|
| #"" | 用于搜索包含目标短语的日志，可以保证关键词出现的顺序。<br><br>短语搜索中的星号（*）和问号（?）会被视为普通字符，因此短语搜索不支持模糊搜索，可以用来搜索日志中的星号（*）和问号（?）。 | request_method:#"GET<br>POST" |

示例说明

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
content: {
 request_method: POST
 request_uri: /authui/login
 request_time: 56
 request_length: 3718
 status: 200
 x-language: zh-cn
 date: Mon, 17 Apr 2023 00:33:48 GMT
 content-type: application/json
 content-encoding: gzip
 scheme: https
 sec-ch-ua-mobile: ?0
 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
 week: 星期一
}
content-encoding: gzip
content-type: application/json
date: Mon, 17 Apr 2023 00:33:48 GMT
request_length: 3718
request_method: POST
request_time: 56
request_uri: /authui/login
scheme: https
sec-ch-ua-mobile: ?0
status: 200
week: 星期一
x-language: zh-cn
```

表 5-19 搜索说明

| 搜索需求                                    | 搜索语句                           |
|-----------------------------------------|--------------------------------|
| 搜索User-Agent字段值包含短语Mon, 17 Apr 2023的日志。 | User-Agent:#"Mon, 17 Apr 2023" |
| 搜索User-Agent字段值包含短语Mozilla/5.0的日志。      | User-Agent:#"Mozilla/5.0"      |
| 搜索week字段值包含短语星期一的日志。                    | week:#"星期一"                    |

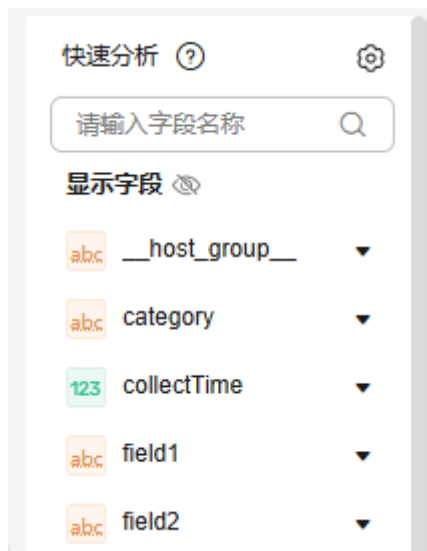
5.6 创建快速分析

日志包含了系统性能及业务等信息，例如关键词ERROR的多少反映了系统的健康度，关键词BUY的多少反映了业务的成交量等，当您需要了解这些信息时，可以通过快速分析功能，指定查询日志关键词，LTS能够针对您配置的关键词进行统计，并生成指标数据，以便您实时了解系统性能及业务等信息。

## 使用限制

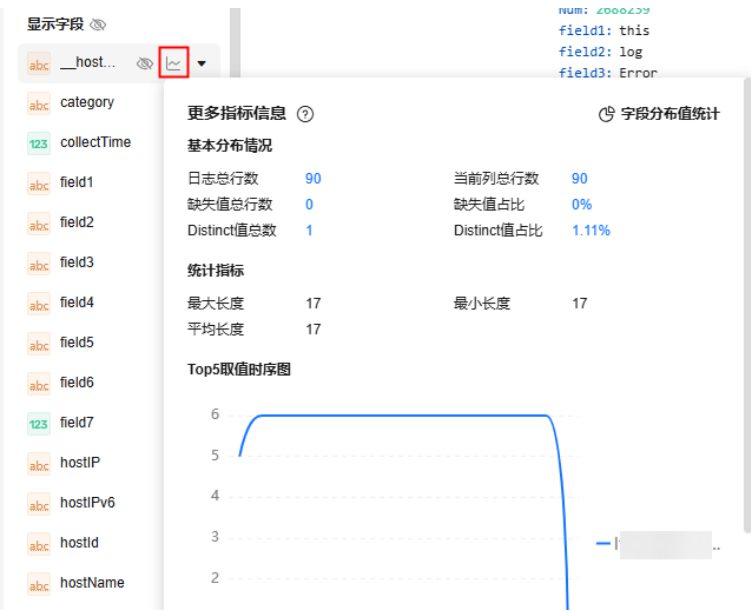
1. 在快速分析下方显示字段的相关限制说明，参考如下：

图 5-17 快速分析字段



- 支持对前100000条日志进行分析。  
快速分析的目的是快速返回字段值的分布情况和变化趋势，并没有对全量数据进行分析，是一种采样结果。
  - 支持通过查询时间和查询条件过滤日志进行分析。  
快速分析是对通过查询语句查询到的日志进行分析，当查询到的日志数目为零时，快速分析没有结果。
  - 支持将快速分析生成查询语句。  
单击快速分析的某一行分析结果，可自动生成查询语句，查询日志并生成新的快速分析。
  - 快速分析的字段长度最大为2000字节。
  - 快速分析字段分布统计展示前100条数据。
2. 在单个字段分析的“更多指标信息”页面。  
字段分析的数据范围，是当前查询页面的（满足时间范围和查询条件）的全部日志。当日志数目在1亿条以下时，会进行全量分析。当日志数目超过了1亿条，会进行采样分析（采样规则是将采样后的数据控制在1亿条左右），如果要避免这种采样，建议您可以缩小时间范围或者增加搜索过滤条件。

图 5-18 更多指标信息



前提条件

快速分析的对象为结构化日志中提取的关键字段，创建快速分析前请确保日志正常上报，且已经完成结构化分析和索引配置。详情请参考[使用ICAgent插件采集主机日志](#)和[创建LTS日志索引](#)。

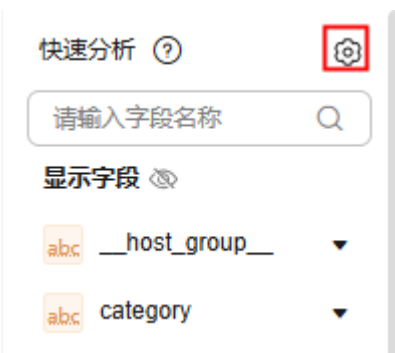
若您在日志接入LTS时没有配置ICAgent结构化解析，可以单独给目标日志流配置ICAgent结构化解析或云端结构化解析。推荐使用ICAgent结构化解析的方式，更多内容请参考[配置ICAgent结构化解析](#)。

创建快速分析

快速分析以日志流为单位，请参考如下步骤进行创建。

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 单击目标日志组或日志流名称，进入日志详情页面。
- 步骤3 在“日志搜索”页签，单击快速分析旁边的  进入“索引配置”页面，在索引配置页签的字段索引下方，单击“自动配置”，LTS自动配置字段索引，同时开启快速分析。

图 5-19 创建快速分析





- 步骤4** 单击“确定”，快速分析创建完成后，支持对字段进行统计分析。例如字段的基本分布情况、各种统计指标以及TOP5取值序列图，为您提供了深入的数据洞察和可视化工具，便于理解和挖掘。
- 在快速分析下方的搜索框输入字段名称，即可显示搜索结果，方便您快速定位目标字段。
  - 字段前面显示abc表示String类型字段。
  - 字段前面显示1.2表示float类型字段。
  - 字段前面显示123表示long类型字段。
  - 字段前面显示{...}表示json类型字段。

**步骤5** 设置隐藏或显示字段。

- 在显示字段下方的目标字段后面单击  即可隐藏单个字段。

图 5-20 隐藏单个字段

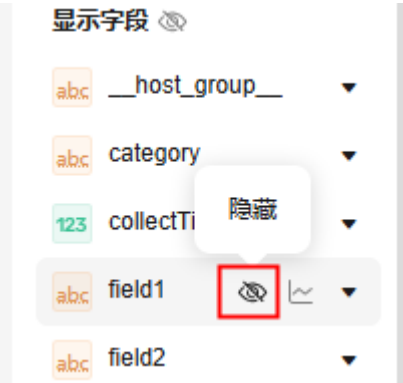
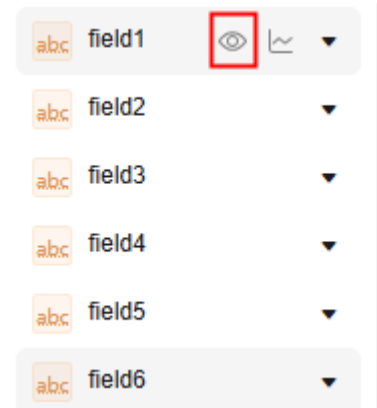


图 5-21 显示单个字段



**步骤6** 在快速分析显示字段下方，若该字段的统计数据大于或等于100条，单击“查看更多”可以跳转至统计图表页面，通过可视化图表，数据可以更直观地展示，便于分析和理解，提升数据分析的效率。

- 步骤7** 单击目标字段后面  图标进入“更多指标信息”页面。
- 数值类型的字段支持展示日志字段基本分布情况（日志总行数、当前列总行数等）、统计指标（最大值、最小值、平均值等）、数值分布直方图。

图 5-22 数值类型的更多指标信息



表 5-20 数值类型字段说明

| 分类     | 参数          | 说明                                 |
|--------|-------------|------------------------------------|
| 基本分布情况 | 日志总行数       | 当前查询页面设置时间和查询条件后统计出来的数据总行数。        |
|        | 当前列总行数      | 当前查询条件中包含该字段所在的行统计出来的总行数。          |
|        | 缺失值总行数      | 日志总行数当前列总行数。                       |
|        | Distinct值总数 | 使用approx_distinct函数计算该列中不重复的字段值数量。 |
| 统计指标   | 最大值         | 该字段的最大值。                           |
|        | 最小值         | 该字段的最小值。                           |
|        | 平均值         | 该字段的平均值。                           |
|        | 中位数         | 将数据正序排列后，返回位于中间位置的数据。              |
|        | 四分位数Q1      | 将数据正序排列后，返回位于25%位置的数据。             |

| 分类      | 参数                     | 说明                     |
|---------|------------------------|------------------------|
|         | 四分位数Q3                 | 将数据正序排列后，返回位于75%位置的数据。 |
|         | 样本标准差                  | 该字段的样本标准差。             |
|         | 总体标准差                  | 该字段的总体标准差。             |
|         | 峰度                     | 统计学概念，表示数据分布的集中程度。     |
|         | 偏度                     | 统计学概念，表示数据的倾斜程度。       |
| 数值分布直方图 | 将数值分布划分为10个区间，统计数据直方图。 |                        |

- 单击“更多指标信息”页面右上角的“数值分布值统计”进入统计图表页面，通过可视化图表，数据可以更直观地展示，便于分析和理解，提升数据分析的效率。
- 字符串类型的字段展示日志字段基本分布情况（日志总行数、当前列总行数等）、统计指标（最大长度、最小长度、平均长度）、Top5取值时序图。

图 5-23 字符串类型的更多指标信息



表 5-21 字符串类型字段说明

| 分类     | 参数     | 说明                          |
|--------|--------|-----------------------------|
| 基本分布情况 | 日志总行数  | 当前查询页面设置时间和查询条件后统计出来的数据总行数。 |
|        | 当前列总行数 | 当前查询条件中包含该字段所在的行统计出来的总行数。   |

| 分类        | 参数                                | 说明                                 |
|-----------|-----------------------------------|------------------------------------|
|           | 缺失值总行数                            | 日志总行数-当前列总行数。                      |
|           | 缺失值占比                             | 缺失值总行数/日志总行数。                      |
|           | Distinct值总数                       | 使用approx_distinct函数计算该列中不重复的字段值数量。 |
|           | Distinct值占比                       | Distinct值总数/日志总行数。                 |
| 统计指标      | 最大长度                              | 该字段值所占的最大字符长度。                     |
|           | 最小长度                              | 该字段值所占的最小字符长度。                     |
|           | 平均长度                              | 该字段值的平均字符长度。                       |
| Top5取值时序图 | 先计算出时间范围内总数前5的取值，然后绘制这些值随时间的变化趋势。 |                                    |

- 单击“更多指标信息”页面右上角的“字段分布值统计”进入统计图表页面，通过可视化图表，数据可以更直观地展示，便于分析和理解，提升数据分析的效率。

**步骤8** 在“更多指标信息”页面，单击显示蓝色的参数值会自动跳转到“统计图表”页面并生成对应的SQL查询语句进行查询，更加直观地展示字段值的分布和变化趋势。更多信息请参见[SQL函数](#)、[使用统计图表将日志可视化](#)。

----结束

## 5.7 保存搜索与分析语句

当您需要重复使用某一个搜索与分析语句时，可以将其设置为快速查询语句，通过保存的快速查询语句，可实现对日志的快速执行搜索与分析操作。

### 创建快速查询

- 在云日志服务控制台，单击“日志管理”。
- 在日志组列表中，单击日志组名称前对应的  按钮。
- 在日志流列表中，单击日志流名称，进入日志详情页面。
- 在“统计图表”页签，单击 ，输入“快速查询名称”和“快速查询语句”。默认开启快速查询和快速查询（保存本地）。

图 5-24 创建快速查询

创建快速查询

\* 快速查询名称

请输入快速查询名称

\* 快速查询语句

请输入快速查询语句

快速查询

开启后，添加到快速查询列表中

快速查询 (保存本地)

开启后，添加到快速查询 (保存本地) 列表中

取消 确定

- 快速查询名称，用于区分多个快速查询语句。名称自定义，需要满足如下要求：
    - 只支持输入英文、数字、中文、中划线、下划线及小数点。
    - 不能以小数点、下划线开头或以小数点结尾。
    - 长度为1-64个字符。
  - 快速查询语句，搜索日志时需要重复使用的关键字，例如“error\*”。
5. 单击“确定”，完成快速查询条件的创建。在左侧导航栏的快速查询页签，即可查看到保存成功的语句。
- 单击快速查询语句的名称，即可查看日志搜索结果。
  - 鼠标悬浮在快速查询语句上面，单击快速查询语句后面的，在打开的“编辑快速查询”页面，修改快速查询语句，单击“确定”。

图 5-25 编辑快速查询



- 鼠标悬浮在快速查询语句上面，单击快速查询语句后面的 ，在打开的“删除快速查询”页面，单击“确定”即可删除快速查询语句。删除后将无法再执行快速查询，请谨慎操作。

## 5.8 查看上下文

查看上下文是指定其中一条日志，将该日志在原始日志中的前若干条（上文）或后若干条日志（下文）也查找出来。您可以通过查找指定日志生成时间点前后的日志，在业务故障排查中快速查找相关故障信息，用于在运维过程中快速定位问题。

### 功能优势

- 免除登录主机的繁琐步骤，在LTS控制台的日志搜索页面，快速查看任意指定日志的上下文信息。
- 结合事件发生的时间范围，在LTS控制台的日志搜索页面，选择日志时间范围即可快速查找可疑日志，然后再查询该日志的上下文，快速定位问题。

### 查看上下文

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 单击目标日志组或日志流名称，进入日志详情页面。
- 步骤3 在“日志搜索”页签下方的原始日志页面，单击 .

图 5-26 查看上下文



- 步骤4 在弹出的上下文日志页面中，查看日志上下文，详细操作请参考[表5-22](#)。

图 5-27 上下文日志



表 5-22 查看上下文日志功能介绍

| 功能   | 说明                                                                                                             |
|------|----------------------------------------------------------------------------------------------------------------|
| 查询行数 | 根据需要选择查询日志的行数。                                                                                                 |
| 高亮显示 | 输入需要高亮的字符串，回车确认，在日志内容中高亮显示。                                                                                    |
| 过滤日志 | 输入需要过滤的字符串，回车确认，在日志内容中高亮显示。当高亮显示和过滤日志同时设置时，均可高亮显示。                                                             |
| 显示字段 | 查看上下文，默认字段为content，单击“显示字段”选择查看其他字段的上下文。                                                                       |
| 更早   | 从当前位置往前查看设置 <b>查询行数</b> 的二分之一。例如：当查询行数设置为100时，单击“更早”则从当前位置朝前显示50行，此时行号为-50；再次单击“更早”，依次叠加分别为-100、-150、-200..... |
| 当前位置 | 当前日志位置。当设置了更早或更新时，单击“当前位置”可回到查看上下文开始的位置，即行数为0时。                                                                |
| 更新   | 从当前位置往后查看设置 <b>查询行数</b> 的二分之一。例如：当查询行数设置为100时，单击“更新”则从当前位置朝后显示50行，此时行号为50；再次单击“更新”，依次叠加分别为100、150、200.....     |
| 简洁模式 | <ul style="list-style-type: none"><li>打开“简洁模式”，只显示行号和content内容。</li><li>关闭“简洁模式”，显示日志详情。</li></ul>             |
| 下载   | 目前仅支持下载content字段内容到本地查看。                                                                                       |

----结束

5.9 查看实时日志

日志接入云日志服务后，每隔大约1分钟上报一次。因此，在实时日志页面，您最多需要等待1分钟左右，即可查看实时上报的日志，实现对日志数据的快速检索与分析。

约束限制

- 若实时日志大于1MB且总数小于2000条时，LTS会清空前500KB的日志，保留最新的500KB日志。
- 若实时日志未超过1MB且总数超过2000条时，LTS会清空前1000条日志，保留最新的1000条日志。

前提条件

切换到实时日志页面后，有日志上报。

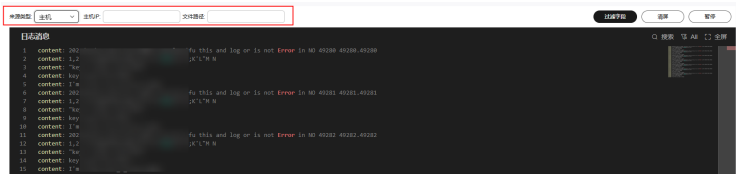
查看实时日志

如果您正在使用实时查看功能，请停留在实时查看页面，请勿切换页面。如果离开实时查看页面，实时查看功能将会停止，重新开启后上一次查看的实时日志将不会显示。

正常情况下，每隔5秒加载一次。如果这5秒内没有产生日志，则不显示；5秒后会继续调用接口，刷新出产生的日志数据。即如果每5秒都有日志数据产生，则加载数据时延为5秒。

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 单击目标日志组或日志流名称，进入日志详情页面。
- 步骤3 选择“实时日志”页签，查看实时日志。

图 5-28 实时日志



日志每隔大约1分钟上报一次，在日志消息区域，您最多需要等待1分钟左右，即可查看实时上报的日志。

有实时日志上报后，您可以进行如下操作：

- 来源类型选择主机时，通过设置主机IP和文件路径筛选日志。
- 来源类型选择K8S时，通过设置实例名称、容器名称或文件路径筛选日志。
- 字段过滤：从索引配置、结构化配置、最新日志获取字段信息，勾选需要过滤的字段查看日志。
- 清屏：清除日志消息区域已经显示出来的日志。
- 暂停：暂停日志消息的实时显示，页面定格在当前已显示的日志。  
暂停后，“暂停”会变成“继续”，再次单击“继续”，日志消息继续实时显示。

----结束

5.10 SQL 函数

5.10.1 聚合函数

本文介绍聚合函数的语法规则，包括参数解释、函数示例等。



函数列表

表 5-23 聚合函数

| 函数                                  | 描述                              |
|-------------------------------------|---------------------------------|
| <a href="#">approx_distinct函数</a>   | 用于估算x中不重复值的个数。                  |
| <a href="#">approx_percentile函数</a> | 用于对x进行正序排列，返回处于percentage位置的数值。 |
| <a href="#">arbitrary函数</a>         | 用于返回x中任意一个非空的值。                 |
| <a href="#">max_by函数</a>            | 查询y为最大值时对应的x值，或查询最大的n个y值对应的x值。  |
| <a href="#">min_by函数</a>            | 查询y为最小值时对应的x值，或查询最小的n个y值对应的x值。  |
| <a href="#">count函数</a>             | 用于计数。                           |
| <a href="#">max函数</a>               | 用于查询x中最大的值。                     |
| <a href="#">min函数</a>               | 用于查询x中最小值。                      |
| <a href="#">avg函数</a>               | 用于计算x的算术平均值。                    |

approx\_distinct 函数

用于估算x中不重复值的个数。

- 估算x中不重复值的个数，默认存在2.3%的标准误差  
语法：approx\_distinct(x)
- 估算x中不重复值的个数，支持自定义标准误差  
语法：approx\_distinct(x, e)

表 5-24 参数说明

| 参数名称 | 描述      | 类型                             | 是否必选 |
|------|---------|--------------------------------|------|
| x    | 原始字段    | 任意                             | 是    |
| e    | 自定义标准误差 | double类型，<br>取值为[0.0115, 0.26] | 否    |

返回值类型：bigint类型

- 示例1：使用approx\_distinct函数估算不重复的clientIp字段值，标准误差为2.3%。

表 5-25 查询分析结果

| 类型 | 查询语句                             | 返回结果 |
|----|----------------------------------|------|
| 场景 | SELECT approx_distinct(clientIp) | 1    |

- 示例2：使用approx\_distinct函数估算不重复的clientIp字段值， 自定义标准误差为10%。

表 5-26 查询分析结果

| 类型 | 查询语句                                  | 返回结果 |
|----|---------------------------------------|------|
| 场景 | SELECT approx_distinct(clientIp, 0.1) | 1    |

approx\_percentile 函数

用于对x进行正序排列，返回处于percentage位置的数值。

- 对于x进行正序排列，返回处于percentage位置的x，返回结果为double类型。  
语法：approx\_percentile(x, percentage)
- 对x进行正序排列，返回处于percentage01、percentage02位置的x，返回结果为array(double,double)类型。  
语法：approx\_percentile(x, array[percentage01, percentage02])
- 对x和权重的乘积进行正序排列，返回大约处于percentage位置的x，返回结果为double类型。  
语法：approx\_percentile(x, weight, percentage)
- 对x和权重的乘积进行正序排列，返回处于percentage01、percentage02位置的x，返回结果为array(double,double)类型。  
语法：approx\_percentile(x, weight, array[percentage01, percentage02...])
- 对x和权重的乘积进行正序排列，返回大约处于percentage位置的x，返回结果为double类型。支持设置返回结果的准确度。  
语法：approx\_percentile(x, weight, percentage, accuracy)

表 5-27 参数说明

| 参数名称       | 描述                               | 类型     | 是否必选 |
|------------|----------------------------------|--------|------|
| x          | 原始字段                             | double | 是    |
| percentage | 百分比值，取值范围为[0,1]。                 | double | 是    |
| weight     | 权重，大于1的整数。设置权重后，系统根据x与权重的乘积进行排序。 | int    | 否    |
| accuracy   | 准确度，取值范围为(0,1)。                  | double | 否    |

返回值类型：double类型或array(double,...,double)类型

- 示例1：对request\_time列进行排序后，返回大约处于50%位置的request\_time字段的值。

表 5-28 查询分析结果

| 类型 | 查询语句                                        | 返回结果 |
|----|---------------------------------------------|------|
| 场景 | SELECT approx_percentile(request_time, 0.5) | 45.0 |

- 示例2：对request\_time列进行排序后，返回处于10%、20%及70%位置的request\_time字段的值。

表 5-29 查询分析结果

| 类型 | 查询语句                                                      | 返回结果               |
|----|-----------------------------------------------------------|--------------------|
| 场景 | SELECT approx_percentile(request_time,array[0.1,0.2,0.7]) | [17.0, 24.0, 59.0] |

- 示例3：根据request\_time与权重的乘积对request\_time列进行排序后，返回大约处于50%位置的request\_time字段的值，权重值为60。

表 5-30 查询分析结果

| 类型 | 查询语句                                            | 返回结果 |
|----|-------------------------------------------------|------|
| 场景 | SELECT approx_percentile(request_time, 60, 0.5) | 45.0 |

- 示例4：根据request\_time与权重的乘积对request\_time列进行排序后，返回大约处于80%和90%位置的request\_time字段的值，权重值为60。

表 5-31 查询分析结果

| 类型 | 查询语句                                                        | 返回结果        |
|----|-------------------------------------------------------------|-------------|
| 场景 | SELECT approx_percentile(request_time, 60, array[0.8, 0.9]) | [66.0,73.0] |

- 示例5：根据request\_time与权重的乘积对request\_time列进行排序后，返回大约处于50%位置的request\_time字段的值，权重值为60，准确度为0.2。

表 5-32 查询分析结果

| 类型 | 查询语句                                                       | 返回结果 |
|----|------------------------------------------------------------|------|
| 场景 | SELECT<br>approx_percentile(request_time, 60,<br>0.5, 0.2) | 45.0 |

arbitrary 函数

用于返回x中任意一个非空的值。

语法：arbitrary(x)

表 5-33 参数说明

| 参数名称 | 描述    | 类型 | 是否必选 |
|------|-------|----|------|
| x    | 原始字段。 | 任意 | 是    |

返回值类型：与参数值的数据类型一致。

示例：select arbitrary(region)

表 5-34 查询分析结果

| 类型 | 查询语句              | 返回结果 |
|----|-------------------|------|
| 场景 | arbitrary(region) | r2   |

max\_by 函数

查询y为最大值时对应的x值，或查询最大的n个y值对应的x值。

- 查询y为最大值时对应的x值。  
语法：max\_by(x, y)
- 查询最大的n个y值对应的x值。  
语法：max\_by(x, y, n)

表 5-35 参数说明

| 参数名称 | 描述    | 类型     | 是否必选 |
|------|-------|--------|------|
| x    | 原始字段。 | 任意数据类型 | 是    |
| y    | 原始字段。 | 任意数据类型 | 是    |

| 参数名称 | 描述      | 类型  | 是否必选 |
|------|---------|-----|------|
| n    | 大于0的整数。 | int | 否    |

返回值类型：与参数值的数据类型一致。

- 示例1：统计请求时长最大时对应的请求方法。

表 5-36 查询分析结果

| 类型 | 查询语句                                        | 返回结果 |
|----|---------------------------------------------|------|
| 场景 | SELECT max_by(request_method, request_time) | GET  |

- 示例2：统计请求时长最大的3个请求对应的请求方法。

表 5-37 查询分析结果

| 类型 | 查询语句                                           | 返回结果                |
|----|------------------------------------------------|---------------------|
| 场景 | SELECT max_by(request_method, request_time, 3) | ["GET","GET","GET"] |

min\_by 函数

查询y为最小值时对应的x值，或查询最小的n个y值对应的x值。

- 查询y为最小值时对应的x值。  
语法：min\_by(x, y)
- 查询最小的n个y值对应的x值。  
语法：min\_by(x, y, n)

表 5-38 参数说明

| 参数名称 | 描述      | 类型     | 是否必选 |
|------|---------|--------|------|
| x    | 原始字段。   | 任意数据类型 | 是    |
| y    | 原始字段。   | 任意数据类型 | 是    |
| n    | 大于0的整数。 | int    | 否    |

返回值类型：与参数值的数据类型一致

- 示例1：统计请求时长最小时对应的请求方法。

表 5-39 查询分析结果

| 类型 | 查询语句                                        | 返回结果 |
|----|---------------------------------------------|------|
| 场景 | SELECT min_by(request_method, request_time) | POST |

- 示例2：统计请求时长最小的3个请求对应的请求方法。

表 5-40 查询分析结果

| 类型 | 查询语句                                           | 返回结果                   |
|----|------------------------------------------------|------------------------|
| 场景 | SELECT min_by(request_method, request_time, 3) | ["POST","POST","POST"] |

count 函数

用于计数。

- 统计所有的日志条数。  
语法：COUNT(\*)
- 统计所有的日志条数。等同于count(\*)。  
语法：COUNT(1)
- 统计x中值不为NULL的日志条数。  
语法：COUNT(x)

表 5-41 参数说明

| 参数名称 | 描述    | 类型 | 是否必选 |
|------|-------|----|------|
| x    | 原始字段。 | 任意 | 是    |

返回值类型：int

示例：select COUNT(\*)

表 5-42 查询分析结果

| 类型 | 查询语句     | 返回结果 |
|----|----------|------|
| 场景 | COUNT(*) | 1    |

max 函数

- 用于查询x中最大的值。
- 查询x中最大的值。  
语法：max(x)
  - 查询x中最大的n个值，结果返回为数组。  
语法：max(x, n)

表 5-43 参数说明

| 参数名称 | 描述       | 类型   | 是否必选 |
|------|----------|------|------|
| x    | 原始字段。    | 任意数据 | 是    |
| n    | 返回最大值的个数 | 正整数  | 否    |

返回值类型：与参数值的数据类型一致。

- 示例1：查询x中的最大值

表 5-44 查询分析结果

| 类型 | 查询语句          | 返回结果 |
|----|---------------|------|
| 场景 | SELECT max(x) | 99.0 |

- 示例2：查询x中的最大的2个值

表 5-45 查询分析结果

| 类型 | 查询语句             | 返回结果         |
|----|------------------|--------------|
| 场景 | SELECT max(x, 2) | [99.0, 99.0] |

min 函数

- 用于查询x中最小值。
- 查询x中最小的值。  
语法：min(x)
  - 查询x中最小的n个值，结果返回为数组。  
语法：min(x, n)

表 5-46 参数说明

| 参数名称 | 描述       | 类型   | 是否必选 |
|------|----------|------|------|
| x    | 原始字段。    | 任意数据 | 是    |
| n    | 返回最小值的个数 | 正整数  | 否    |

返回值类型：与参数值的数据类型一致。

- 示例1：查询x中的最小值

表 5-47 查询分析结果

| 类型 | 查询语句          | 返回结果 |
|----|---------------|------|
| 场景 | SELECT min(x) | 10.0 |

- 示例2：查询x中的最小的2个值

表 5-48 查询分析结果

| 类型 | 查询语句             | 返回结果         |
|----|------------------|--------------|
| 场景 | SELECT min(x, 2) | [10.0, 10.0] |

avg 函数

用于计算x的算术平均值。

语法：AVG(x)

表 5-49 参数说明

| 参数名称 | 描述    | 类型     | 是否必选 |
|------|-------|--------|------|
| x    | 原始字段。 | number | 是    |

返回值类型：double

示例：select AVG(value)

表 5-50 查询分析结果

| 类型 | 查询语句       | 返回结果 |
|----|------------|------|
| 场景 | AVG(value) | 1    |



### 5.10.2 字符串函数

本文介绍字符串函数的语法规则，包括参数解释、函数示例等。

#### 函数列表

表 5-51 字符串函数

| 函数                                 | 描述                                                      |
|------------------------------------|---------------------------------------------------------|
| <a href="#">lpad函数</a>             | 在字符串的开头填充指定字符，直到指定长度后返回结果字符串。如果未指定填充字符，则使用“ ”进行填充。      |
| <a href="#">rpad函数</a>             | 在字符串的结尾填充指定字符，直到指定长度后返回结果字符串。如果未指定填充字符，则使用“ ”进行填充。      |
| <a href="#">replace函数</a>          | 将字符串中的匹配字符串替换成指定的字符串，如果未指定替换字符串，则将匹配的字符串从原来的字符串中删除。     |
| <a href="#">reverse函数</a>          | 将字符串转换成反向顺序的字符串。                                        |
| <a href="#">repeat函数</a>           | 将字符串重复指定次数。                                             |
| <a href="#">contains_string函数</a>  | 判断字符串是否包含指定字符串。                                         |
| <a href="#">icontains_string函数</a> | 判断字符串是否包含指定字符串，字符串不区分大小写。                               |
| <a href="#">textcat函数</a>          | 将两个字符串拼接起来，返回拼接后的结果。                                    |
| <a href="#">btrim函数</a>            | 删除字符串左右两侧的空格。                                           |
| <a href="#">parse_long函数</a>       | 将字符串转换成基于指定基数的Long型，如果未指定基数，则基于十进制对原始字符串进行转换。           |
| <a href="#">split函数</a>            | 使用指定的分隔符拆分字符串，并返回子字符串的集合。如果设置了limit，则使用limit限制拆分字符串的数量。 |
| <a href="#">split_part函数</a>       | 使用指定的分隔符拆分字符串，并返回指定位置的字符串。如果拆分后指定的位置超过数组的长度，则返回“ ”。     |
| <a href="#">split_to_map函数</a>     | 使用指定的第一个分隔符拆分字符串，然后使用指定的第二个分隔符第二次拆分字符串，返回第二次拆分后的结果。     |
| <a href="#">string_format函数</a>    | 返回以Java的String.format方式格式化的字符串。                         |
| <a href="#">strpos函数</a>           | 查询目标子字符串初次出现在字符串中的位置，如果目标子字符串未出现在字符串中，则返回0。             |
| <a href="#">substr函数</a>           | 使用起始位置和长度返回子字符串。如果没有长度参数，则返回从起始位置开始的整个字符串，同substring。   |
| <a href="#">substring函数</a>        | 使用起始位置和长度返回子字符串。如果没有长度参数，则返回从起始位置开始的整个字符串。              |

| 函数                                     | 描述                                                                        |
|----------------------------------------|---------------------------------------------------------------------------|
| <a href="#">length函数</a>               | 如果字符串表达式是字符数据类型，则返回输入的字符长度；否则，返回字符串表达式的字节长度（不小于位数除以8的最小整数）。               |
| <a href="#">char_length函数</a>          | 与length函数功能一致，如果字符串表达式是字符数据类型，则返回输入的字符长度；否则，返回字符串表达式的字节长度（不小于位数除以8的最小整数）。 |
| <a href="#">character_length函数</a>     | 与length函数功能一致。                                                            |
| <a href="#">strlen函数</a>               | 与length函数功能一致。                                                            |
| <a href="#">levenshtein_distance函数</a> | 计算两个字符串str1和str2之间的最小编辑距离。                                                |
| <a href="#">normalize函数</a>            | 格式化字符串。                                                                   |
| <a href="#">to_utf8函数</a>              | 将字符串转换为UTF-8编码格式。                                                         |
| <a href="#">chr函数</a>                  | 将ascii值转换成字符。                                                             |
| <a href="#">concat函数</a>               | 用于将多个参数拼接成一个字符串。                                                          |

## lpad 函数

在字符串的开头填充指定字符，直到指定长度后返回结果字符串。如果未指定填充字符，则使用“ ”进行填充。

语法：lpad(str, len, lpadStr)

表 5-52 参数说明

| 参数名称    | 描述                                                                                                                                   | 类型     | 是否必选 |
|---------|--------------------------------------------------------------------------------------------------------------------------------------|--------|------|
| str     | 原始字符串。                                                                                                                               | String | 是    |
| len     | 结果字符串的长度。 <ul style="list-style-type: none"><li>如果原始字符串的长度小于len，那么在字符串的开头填充指定的字符。</li><li>如果原始字符串的长度大于len，那么只返回字符串的len个字符。</li></ul> | int    | 是    |
| lpadStr | 填充字符。                                                                                                                                | String | 否    |

返回值类型：String类型

示例：SELECT LPAD('hello world', 10), LPAD('hello', 10, 'e')

表 5-53 查询分析结果

| 类型  | 查询语句                    | 返回结果       |
|-----|-------------------------|------------|
| 场景1 | LPAD('hello world', 10) | hello worl |
| 场景2 | LPAD('hello', 10, 'e')  | eeeeehello |

rpad 函数

在字符串的结尾填充指定字符，直到指定长度后返回结果字符串。如果未指定填充字符，则使用 “ ” 进行填充。

语法：rpad(str, len, rpadStr)

表 5-54 参数说明

| 参数名称    | 描述                                                                                                                                   | 类型      | 是否必选 |
|---------|--------------------------------------------------------------------------------------------------------------------------------------|---------|------|
| str     | 原始字符串。                                                                                                                               | String  | 是    |
| len     | 结果字符串的长度。 <ul style="list-style-type: none"><li>如果原始字符串的长度小于len，那么在字符串的结尾填充指定的字符。</li><li>如果原始字符串的长度大于len，那么只返回字符串的len个字符。</li></ul> | Integer | 是    |
| rpadStr | 填充字符。                                                                                                                                | String  | 否    |

返回值类型：String类型

示例：SELECT RPAD('hello world', 10), RPAD('hello', 10, 'e')

表 5-55 查询分析结果

| 类型  | 查询语句                    | 返回结果        |
|-----|-------------------------|-------------|
| 场景1 | RPAD('hello world', 10) | hello worl  |
| 场景2 | RPAD('hello', 10, 'e')  | helloeeeeee |

replace 函数

将字符串中的匹配字符串替换成指定的字符串，如果未指定替换字符串，则将匹配的字符串从原来的字符串中删除。

语法：replace(str, subStr, replaceStr)

表 5-56 参数说明

| 参数名称       | 描述        | 类型     | 是否必选 |
|------------|-----------|--------|------|
| str        | 原始字符串。    | String | 是    |
| subStr     | 目标子字符串。   | String | 是    |
| replaceStr | 用于替换的字符串。 | String | 否    |

返回值类型：String类型

示例：SELECT REPLACE('hello world', 'o'), REPLACE('hello world', 'w', 'new w')

表 5-57 查询分析结果

| 类型  | 查询语句                                 | 返回结果            |
|-----|--------------------------------------|-----------------|
| 场景1 | REPLACE('hello world', 'o')          | hell wrld       |
| 场景2 | REPLACE('hello world', 'w', 'new w') | hello new world |

reverse 函数

将字符串转换成反向顺序的字符串。

语法：reverse(str)

表 5-58 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| str  | 原始字符串。 | String | 是    |

返回值类型：String类型

示例：SELECT REVERSE('hello world')

表 5-59 查询分析结果

| 类型 | 查询语句                   | 返回结果        |
|----|------------------------|-------------|
| 场景 | REVERSE('hello world') | dlrow olleh |

repeat 函数

将字符串重复指定次数。

语法：repeat(str, num)

表 5-60 参数说明

| 参数名称 | 描述     | 类型      | 是否必选 |
|------|--------|---------|------|
| str  | 原始字符串。 | String  | 是    |
| num  | 重复次数   | Integer | 是    |

返回值类型：String类型

示例：SELECT REPEAT('hello world', 2), REPEAT('hello world', 0)

表 5-61 查询分析结果

| 类型  | 查询语句                     | 返回结果                   |
|-----|--------------------------|------------------------|
| 场景1 | REPEAT('hello world', 2) | hello worldhello world |
| 场景2 | REPEAT('hello world', 0) |                        |

contains\_string 函数

判断字符串是否包含指定字符串。

语法：contains\_string(str, containsStr)

表 5-62 参数说明

| 参数名称        | 描述     | 类型     | 是否必选 |
|-------------|--------|--------|------|
| str         | 原始字符串。 | String | 是    |
| containsStr | 指定字符串。 | String | 是    |

返回值类型：Boolean类型

示例：SELECT CONTAINS\_STRING('hello world', 'llo')

表 5-63 查询分析结果

| 类型 | 查询语句                                  | 返回结果 |
|----|---------------------------------------|------|
| 场景 | CONTAINS_STRING('hello world', 'llo') | true |

icontains\_string 函数

判断字符串是否包含指定字符串，字符串不区分大小写。

语法: icontains\_string(str, containsStr)

表 5-64 参数说明

| 参数名称        | 描述     | 类型     | 是否必选 |
|-------------|--------|--------|------|
| str         | 原始字符串。 | String | 是    |
| containsStr | 指定字符串。 | String | 是    |

返回值类型: Boolean类型

示例: SELECT ICONTAINS\_STRING('hello world', 'llo')

表 5-65 查询分析结果

| 类型 | 查询语句                                   | 返回结果 |
|----|----------------------------------------|------|
| 场景 | ICONTAINS_STRING('hello world', 'llo') | true |

textcat 函数

将两个字符串拼接起来，返回拼接后的结果。

语法: textcat(str1, str2)

表 5-66 参数说明

| 参数名称 | 描述       | 类型     | 是否必选 |
|------|----------|--------|------|
| str1 | 待拼接字符串1。 | String | 是    |
| str2 | 待拼接字符串2。 | String | 是    |

返回值类型: String类型

示例: SELECT TEXTCAT('hello ', 'world')

表 5-67 查询分析结果

| 类型 | 查询语句                       | 返回结果        |
|----|----------------------------|-------------|
| 场景 | TEXTCAT('hello ', 'world') | hello world |

btrim 函数

删除字符串左右两侧的空格。

语法: btrim(str)

表 5-68 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| str  | 原始字符串。 | String | 是    |

返回值类型：String类型  
示例：SELECT BTRIM(' hello world ')

表 5-69 查询分析结果

| 类型 | 查询语句                   | 返回结果        |
|----|------------------------|-------------|
| 场景 | BTRIM(' hello world ') | hello world |

parse\_long 函数

将字符串转换成基于指定基数的Long型，如果未指定基数，则基于十进制对原始字符串进行转换。  
语法：parse\_long(str, radix)

表 5-70 参数说明

| 参数名称  | 描述     | 类型      | 是否必选 |
|-------|--------|---------|------|
| str   | 原始字符串。 | String  | 是    |
| radix | 基数。    | Integer | 否    |

返回值类型：Long类型  
示例：SELECT PARSE\_LONG('-1234', 8), PARSE\_LONG('1234')

表 5-71 查询分析结果

| 类型  | 查询语句                   | 返回结果 |
|-----|------------------------|------|
| 场景1 | PARSE_LONG('-1234', 8) | -668 |
| 场景2 | PARSE_LONG('1234')     | 1234 |

split 函数

使用指定的分隔符拆分字符串，并返回子字符串的集合。如果设置了limit，则使用limit限制拆分字符串的数量。  
语法：split(str, splitStr, limit)

表 5-72 参数说明

| 参数名称     | 描述     | 类型      | 是否必选 |
|----------|--------|---------|------|
| str      | 原始字符串。 | String  | 是    |
| splitStr | 分隔符。   | String  | 是    |
| limit    | 基数。    | Integer | 否    |

返回值类型：ARRAYS<STRING>类型

示例：SELECT SPLIT('helloworld', 'o'), SPLIT('helloworld', 'o', 2)

表 5-73 查询分析结果

| 类型  | 查询语句                        | 返回结果               |
|-----|-----------------------------|--------------------|
| 场景1 | SPLIT('helloworld','o' )    | ["hell","w","rld"] |
| 场景2 | SPLIT('helloworld', 'o', 2) | ["hell","world"]   |

split\_part 函数

使用指定的分隔符拆分字符串，并返回指定位置的字符串。如果拆分后指定的位置超过数组的长度，则返回 “ ”。

语法：split\_part(str, splitStr, position)

表 5-74 参数说明

| 参数名称     | 描述           | 类型      | 是否必选 |
|----------|--------------|---------|------|
| str      | 原始字符串。       | String  | 是    |
| splitStr | 分隔符。         | String  | 是    |
| position | 需要返回的字符串的位置。 | Integer | 是    |

返回值类型：STRING类型

示例：SELECT SPLIT\_PART('helloworld', 'o', 2), SPLIT\_PART('helloworld', 'o', 4)

表 5-75 查询分析结果

| 类型  | 查询语句                             | 返回结果 |
|-----|----------------------------------|------|
| 场景1 | SPLIT_PART('helloworld','o', 2)  | w    |
| 场景2 | SPLIT_PART('helloworld', 'o', 4) |      |



### split\_to\_map 函数

使用指定的第一个分隔符拆分字符串，然后使用指定的第二个分隔符第二次拆分字符串。返回第二次拆分后的结果。

语法：split\_to\_map(str, splitStr1, splitStr2)

表 5-76 参数说明

| 参数名称      | 描述     | 类型     | 是否必选 |
|-----------|--------|--------|------|
| str       | 原始字符串。 | String | 是    |
| splitStr1 | 分隔符1。  | String | 是    |
| splitStr2 | 分隔符2。  | String | 是    |

返回值类型：ARRAY类型

示例：SELECT SPLIT\_TO\_MAP('upstream\_response\_time:123, request\_time:456', ',', ':')

表 5-77 查询分析结果

| 类型 | 查询语句                                                                   | 返回结果                                                  |
|----|------------------------------------------------------------------------|-------------------------------------------------------|
| 场景 | SPLIT_TO_MAP('upstream_response_time:123, request_time:456', ',', ':') | {"request_time":"456","upstream_response_time":"123"} |

### string\_format 函数

返回以Java的String.format方式格式化的字符串。

语法：string\_format(str, Object... args)

表 5-78 参数说明

| 参数名称 | 描述       | 类型                                 | 是否必选 |
|------|----------|------------------------------------|------|
| str  | 原始字符串。   | String                             | 是    |
| args | 输出的时间格式。 | String/Integer/Long/Boolean/Double | 是    |

返回值类型：String类型

示例：SELECT STRING\_FORMAT('My name is %s and I am %d years old.', 'name', age)

表 5-79 查询分析结果

| 类型 | 查询语句                                                             | 返回结果                                  |
|----|------------------------------------------------------------------|---------------------------------------|
| 场景 | STRING_FORMAT('My name is %s and I am %d years old.', 'Tom', 25) | My name is Tom and I am 25 years old. |

- %d: Output integer.
- %f: Output floating point number.
- %s: Output character string.
- %n: Output newline character.
- %c: Output Characters.
- %b: Output bool type.
- %e: Output Exponential Type.
- %tc: Output Date and time information.
- %tF: Output format is YYYY-MM-DD.
- %tr: Output in HH:MM:SS PM format.

strpos 函数

查询目标子字符串初次出现在字符串中的位置，如果目标子字符串未出现在字符串中，则返回0。

语法：strpos(str, subStr)

表 5-80 参数说明

| 参数名称   | 描述      | 类型     | 是否必选 |
|--------|---------|--------|------|
| str    | 原始字符串。  | String | 是    |
| subStr | 目标子字符串。 | String | 是    |

返回值类型：Integer类型

示例：SELECT STRPOS('hello world', 'llo'), STRPOS('llo', 'hello world')

表 5-81 查询分析结果

| 类型  | 查询语句                         | 返回结果 |
|-----|------------------------------|------|
| 场景1 | STRPOS('hello world', 'llo') | 3    |
| 场景2 | STRPOS('llo', 'hello world') | 0    |

substr 函数

使用起始位置和长度返回子字符串。如果没有长度参数，则返回从起始位置开始的整个字符串，同substring。

语法：substr(str, start, length)

表 5-82 参数说明

| 参数名称   | 描述                  | 类型      | 是否必选 |
|--------|---------------------|---------|------|
| str    | 原始字符串               | String  | 是    |
| start  | 函数用于返回字符串中指定位置的子字符串 | Integer | 是    |
| length | 子字符串的长度             | Integer | 否    |

返回值类型：String类型

示例：SELECT SUBSTR('helloworld', 5), SUBSTR('helloworld', 5, 3)

表 5-83 查询分析结果

| 类型  | 查询语句                        | 返回结果   |
|-----|-----------------------------|--------|
| 场景1 | SUBSTR ('helloworld', 5)    | oworld |
| 场景2 | SUBSTR ('helloworld', 5, 3) | owo    |

substring 函数

使用起始位置和长度返回子字符串。如果没有长度参数，则返回从起始位置开始的整个字符串。

语法：substring(str, start, length)

表 5-84 参数说明

| 参数名称   | 描述                | 类型      | 是否必选 |
|--------|-------------------|---------|------|
| str    | 原始字符串。            | String  | 是    |
| start  | 开始提取子字符串的位置，从1开始。 | Integer | 是    |
| length | 子字符串的长度。          | Integer | 否    |

返回值类型：String类型

示例：SELECT SUBSTRING('helloworld', 5), SUBSTRING('helloworld', 5, 3)

表 5-85 查询分析结果

| 类型  | 查询语句                          | 返回结果   |
|-----|-------------------------------|--------|
| 场景1 | SUBSTRING('helloworld', 5)    | oworld |
| 场景2 | SUBSTRING('helloworld', 5, 3) | owo    |

length 函数

如果字符串表达式是字符数据类型，则返回输入的字符长度；否则，返回字符串表达式的字节长度（不小于位数除以8的最小整数）。

语法：length(str)

表 5-86 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| str  | 原始字符串。 | String | 是    |

返回值类型：Integer类型  
示例：SELECT LENGTH('HELLO WORLD')

表 5-87 查询分析结果

| 类型 | 查询语句                  | 返回结果 |
|----|-----------------------|------|
| 场景 | LENGTH('HELLO WORLD') | 11   |

char\_length 函数

与length函数功能一致，如果字符串表达式是字符数据类型，则返回输入的字符长度；否则，返回字符串表达式的字节长度（不小于位数除以8的最小整数）。

语法：char\_length(str)

表 5-88 参数说明

| 参数名称 | 描述    | 类型     | 是否必选 |
|------|-------|--------|------|
| str  | 原始字符串 | String | 是    |

返回值类型：Integer类型  
示例：SELECT CHAR\_LENGTH('HELLO WORLD')

表 5-89 查询分析结果

| 类型 | 查询语句                       | 返回结果 |
|----|----------------------------|------|
| 场景 | CHAR_LENGTH('HELLO WORLD') | 11   |

character\_length 函数

与length函数功能一致。

语法：character\_length(str)

表 5-90 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| str  | 原始字符串。 | String | 是    |

返回值类型：Integer类型

示例：SELECT CHARACTER\_LENGTH('HELLO WORLD')

表 5-91 查询分析结果

| 类型 | 查询语句                            | 返回结果 |
|----|---------------------------------|------|
| 场景 | CHARACTER_LENGTH('HELLO WORLD') | 11   |

strlen 函数

与length函数功能一致。

语法：strlen(str)

表 5-92 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| str  | 原始字符串。 | String | 是    |

返回值类型：Integer类型

示例：SELECT STRLEN('HELLO WORLD')

表 5-93 查询分析结果

| 类型 | 查询语句                  | 返回结果 |
|----|-----------------------|------|
| 场景 | STRLEN('HELLO WORLD') | 11   |

levenshtein\_distance 函数

计算两个字符串str1和str2之间的最小编辑距离。

语法: levenshtein\_distance(str1, str2)

表 5-94 参数说明

| 参数名称 | 描述      | 类型     | 是否必选 |
|------|---------|--------|------|
| str1 | 原始字符串1。 | String | 是    |
| str2 | 原始字符串2。 | String | 是    |

返回值类型: Integer类型

示例: SELECT LEVENSHTEIN\_DISTANCE('horse', 'ros')

表 5-95 查询分析结果

| 类型 | 查询语句                                 | 返回结果 |
|----|--------------------------------------|------|
| 场景 | LEVENSHTEIN_DISTANCE('horse', 'ros') | 3    |

normalize 函数

格式化字符串str。

- 使用NFC格式来格式化字符串str。  
语法: normalize(str)
- 使用给定格式来格式化字符串str，支持的格式有NFC，NFD，NFKC，NFKD。  
语法: normalize(str, form)

表 5-96 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| str  | 原始字符串。 | String | 是    |
| form | 格式化类型。 | String | 否    |

返回值类型：String类型  
示例：SELECT NORMALIZE('schön'), NORMALIZE('Henry \u2163', 'nfd')

表 5-97 查询分析结果

| 类型  | 查询语句                             | 返回结果     |
|-----|----------------------------------|----------|
| 场景1 | NORMALIZE('schön')               | schön    |
| 场景2 | NORMALIZE('Henry \u2163', 'nfd') | Henry IV |

to\_utf8 函数

将字符串转换为UTF-8编码格式。  
语法：to\_utf8(str)

表 5-98 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| str1 | 原始字符串1 | String | 是    |
| str2 | 原始字符串2 | String | 是    |

返回值类型：Integer类型  
示例：SELECT TO\_UTF8('%')

表 5-99 查询分析结果

| 类型 | 查询语句         | 返回结果 |
|----|--------------|------|
| 场景 | TO_UTF8('%') | JQ== |

chr 函数

将ascii值转换成字符。  
语法：chr(ascii)

表 5-100 参数说明

| 参数名称  | 描述     | 类型      | 是否必选 |
|-------|--------|---------|------|
| ascii | ascii码 | Integer | 是    |

返回值类型：String类型

示例：SELECT CHR(99)

表 5-101 查询分析结果

| 类型 | 查询语句    | 返回结果 |
|----|---------|------|
| 场景 | CHR(99) | c    |

concat 函数

用于将多个参数拼接成一个字符串。

语法：concat(x, y...)

表 5-102 参数说明

| 参数名称   | 描述    | 类型     | 是否必选 |
|--------|-------|--------|------|
| x,y... | 原始字段。 | string | 是    |

返回值类型：String类型

示例：select concat('1','1')

表 5-103 查询分析结果

| 类型 | 查询语句            | 返回结果 |
|----|-----------------|------|
| 场景 | concat('1','1') | 1+1  |

5.10.3 日期和时间函数

本文介绍日期、时间函数的语法规则，包括参数解释、函数示例等。管道符特性处理的日期和时间timestamp类型，返回值格式表现为yyyy-MM-dd HH:mm:ss.SSS TimeZone。

函数列表

表 5-104 日期和时间函数

| 函数                                   | 描述                                            |
|--------------------------------------|-----------------------------------------------|
| <a href="#">current_timestamp 函数</a> | 返回当前日期和时间，格式为yyyy-MM-dd HH:mm:ss.SSS          |
| <a href="#">from_iso8601_date 函数</a> | 将ISO8601格式的日期表达式转换为date类型的日期表达式，格式为yyyy-MM-dd |



| 函数                                       | 描述                                                                                                                                         |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">from_iso8601_timestamp函数</a> | 将ISO8601格式的日期表达式转换为timestamp类型的日期表达式，格式为yyyy-MM-dd HH:mm:ss.SSS                                                                            |
| <a href="#">mills_to_timestamp函数</a>     | 将毫秒级的时间戳转换为timestamp类型的时间和日期表达式。                                                                                                           |
| <a href="#">from_unixtime函数</a>          | 将秒级或毫秒级时间戳转换为日期和时间表达式。                                                                                                                     |
| <a href="#">to_iso8601函数</a>             | 将日期类型或时间戳类型的日期时间表达式转换为ISO8601格式的日期时间表达式。                                                                                                   |
| <a href="#">timestamp_to_mills函数</a>     | 将timestamp类型的日期和时间表达式转换为毫秒级的时间戳。                                                                                                           |
| <a href="#">to_unixtime函数</a>            | 将timestamp类型的日期和时间表达式转换毫秒级的时间戳。                                                                                                            |
| <a href="#">time_ceil函数</a>              | 将timestamp或long类型的时间戳按固定周期以向上舍入的方式取整，将其作为新的时间戳返回。Period可以是任何ISO8601的周期，如P3M（季度）或PT12H（半天）。                                                 |
| <a href="#">time_floor函数</a>             | 将timestamp或long类型的时间戳按时间戳返回。Period可以是任何ISO8601的周期，如P3M（季度）或PT12H（半天）。                                                                      |
| <a href="#">ceil函数</a>                   | 使用时间单位对时间戳进行向上舍入，单位可以是SECOND、MINUTE、HOUR、DAY、WEEK、MONTH、QUARTER或YEAR。                                                                      |
| <a href="#">floor函数</a>                  | 使用时间单位对时间戳进行向下舍入，单位可以是SECOND、MINUTE、HOUR、DAY、WEEK、MONTH、QUARTER或YEAR。                                                                      |
| <a href="#">time_shift函数</a>             | 将时间戳expr移动一个Period（步长时间）。Period可以是任何ISO8601的Period。                                                                                        |
| <a href="#">timezone_hour函数</a>          | 计算时区的小时偏移量。                                                                                                                                |
| <a href="#">timezone_minute函数</a>        | 计算时区的分钟偏移量。                                                                                                                                |
| <a href="#">time_format函数</a>            | 将毫秒级的时间戳或timestamp类型的日期和时间表达式expr，依据用户自定义的format参数以 <a href="#">Joda DateTimeFormat</a> 模式转换为指定日期和时间格式的字符串。                                |
| <a href="#">time_parse函数</a>             | time_parse将给定的字符串expr，依据用户自定义的format参数以 <a href="#">Joda DateTimeFormat</a> 模式解析为timestamp类型的日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TimeZone。 |
| <a href="#">date_format函数</a>            | timestamp类型的日期和时间表达式转换为指定日期和时间格式的字符串。                                                                                                      |

| 函数                                 | 描述                                              |
|------------------------------------|-------------------------------------------------|
| <a href="#">date_parse函数</a>       | 将日期和时间字符串转换为指定格式的时间戳类型的日期和时间表达式。                |
| <a href="#">time_extract函数</a>     | 通过指定字段提取日期时间表达式的日期或时间部分。                        |
| <a href="#">date_trunc函数</a>       | 根据您的指定的时间单位截断日期和时间表达式，指定的时间单位有毫秒、秒、分钟、小时、天、月或年。 |
| <a href="#">date_diff函数</a>        | 返回时间戳expr1和时间戳expr2之间的单位数（有符号）。                 |
| <a href="#">current_date函数</a>     | 返回当前日期，格式是“yyyy-MM-dd”。                         |
| <a href="#">now函数</a>              | 返回当前日期和时间。                                      |
| <a href="#">date_add函数</a>         | 给时间加上给定的时间间隔。                                   |
| <a href="#">current_time函数</a>     | 返回当前时间，格式为HH:mm:ss.SSSSSS。                      |
| <a href="#">current_timezone函数</a> | 返回当前时区。                                         |
| <a href="#">localtime函数</a>        | 返回本地时间。                                         |
| <a href="#">localtimestamp函数</a>   | 返回本地的日期和时间。                                     |
| <a href="#">year_of_week函数</a>     | 返回目标日期在ISO周日历中的年份。year_of_week函数等同于yow函数。       |
| <a href="#">yow函数</a>              | 返回目标日期在ISO周日历中的年份。year_of_week函数等同于yow函数。       |
| <a href="#">time_series函数</a>      | 用于补全您查询时间窗口内缺失的数据。                              |

**current\_timestamp 函数**

返回当前日期和时间，格式为yyyy-MM-dd HH:mm:ss.SSS。

语法：current\_timestamp()

返回值类型：timestamp类型

示例：SELECT CURRENT\_TIMESTAMP()

表 5-105 查询分析结果

| 类型 | 查询语句                | 返回结果                                     |
|----|---------------------|------------------------------------------|
| 场景 | CURRENT_TIMESTAMP() | 2025-05-29 20:09:17.656<br>Asia/Shanghai |

from\_iso8601\_date 函数

将ISO8601格式的日期表达式转化为date类型的日期表达式，格式为YYYY-MM-DD。

语法：from\_iso8601\_date(expr)

表 5-106 参数说明

| 参数名称 | 描述              | 类型     | 是否必选 |
|------|-----------------|--------|------|
| expr | ISO8601格式的日期表达式 | String | 是    |

返回值类型：String类型

示例：SELECT FROM\_ISO8601\_DATE('2018-05-14')

表 5-107 查询分析结果

| 类型 | 查询语句                            | 返回结果       |
|----|---------------------------------|------------|
| 场景 | FROM_ISO8601_DATE('2018-05-14') | 2018-05-14 |

from\_iso8601\_timestamp 函数

将ISO8601格式的日期和时间表达式转化为timestamp类型的日期和时间表达式，格式为YYYY-MM-DD HH:MM:SS.Ms Time\_zone。

语法：from\_iso8601\_timestamp(expr)

表 5-108 参数说明

| 参数名称 | 描述                 | 类型     | 是否必选 |
|------|--------------------|--------|------|
| expr | ISO8601格式的日期和时间表达式 | String | 是    |

返回值类型：timestamp类型。

示例：SELECT FROM\_ISO8601\_TIMESTAMP('2018-05-14T11:51:50.153+08:00')

表 5-109 查询分析结果

| 类型 | 查询语句                                                    | 返回结果                          |
|----|---------------------------------------------------------|-------------------------------|
| 场景 | FROM_ISO8601_TIMESTAMP('2018-05-14T11:51:50.153+08:00') | 2018-05-14 11:51:50.153+08:00 |

### mills\_to\_timestamp 函数

将毫秒级的时间戳转换为timestamp类型的时间和日期表达式。

- 将毫秒级的时间戳转化为timestamp类型的日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS。  
语法：mills\_to\_timestamp(expr)

表 5-110 参数说明

| 参数名称 | 描述      | 类型   | 是否必选 |
|------|---------|------|------|
| expr | UNIX时间戳 | Long | 是    |

返回值类型：timestamp类型  
示例：SELECT MILLS\_TO\_TIMESTAMP(1626774758000)

表 5-111 查询分析结果

| 类型 | 查询语句                              | 返回结果                                  |
|----|-----------------------------------|---------------------------------------|
| 场景 | MILLS_TO_TIMESTAMP(1626774758000) | 2021-07-20 17:52:38.000 Asia/Shanghai |

### from\_unixtime 函数

将秒级或毫秒级时间戳转换为日期和时间表达式。当expr取值小于315360000000视为秒级时间戳。

- 将秒级或毫秒级时间戳转化为时间戳类型的日期和时间表达式。  
语法：from\_unixtime(expr)

表 5-112 参数说明

| 参数名称 | 描述        | 类型   | 是否必选 |
|------|-----------|------|------|
| expr | 秒级或毫秒级时间戳 | Long | 是    |

返回值类型：timestamp  
示例：SELECT FROM\_UNIXTIME(1626774758000)

表 5-113 查询分析结果

| 类型 | 查询语句                         | 返回结果                                  |
|----|------------------------------|---------------------------------------|
| 场景 | FROM_UNIXTIME(1626774758000) | 2021-07-20 17:52:38.000 Asia/shanghai |

示例：SELECT FROM\_UNIXTIME(1731916857)

表 5-114 查询分析结果

| 类型 | 查询语句                      | 返回结果                                     |
|----|---------------------------|------------------------------------------|
| 场景 | FROM_UNIXTIME(1731916857) | 2024-11-18 16:00:57.000<br>Asia/shanghai |

- 将秒级或毫秒级时间戳转化为带特定时区的timestamp类型的日期和时间表达式。  
语法：from\_unixtime(expr, timezone)

表 5-115 参数说明

| 参数名称         | 描述      | 类型     | 是否必选 |
|--------------|---------|--------|------|
| expr         | UNIX时间戳 | Long   | 是    |
| timezon<br>e | 时区      | String | 是    |

返回值类型：String类型

示例：SELECT FROM\_UNIXTIME(1626774758000, 'Asia/Shanghai')

表 5-116 查询分析结果

| 类型 | 查询语句                                          | 返回结果                                     |
|----|-----------------------------------------------|------------------------------------------|
| 场景 | FROM_UNIXTIME(1626774758000, 'Asia/Shanghai') | 2021-07-20 17:52:38.000<br>Asia/Shanghai |

- 将秒级或毫秒级时间戳转换为具有特定时区的日期和时间表达式。表达式中，hour和minute表示时区偏移量。  
语法：from\_unixtime(expr, hour, minutes)

表 5-117 参数说明

| 参数名称   | 描述      | 类型      | 是否必选 |
|--------|---------|---------|------|
| expr   | UNIX时间戳 | Long    | 是    |
| hour   | 小时      | Integer | 是    |
| minute | 分钟      | Integer | 是    |

返回值类型：String

示例：SELECT FROM\_UNIXTIME(1626774758000, -2, 0)

表 5-118 查询分析结果

| 类型 | 查询语句                                | 返回结果                          |
|----|-------------------------------------|-------------------------------|
| 场景 | FROM_UNIXTIME(1626774758000, -2, 0) | 2021-07-20 07:52:38.000-02:00 |

## to\_iso8601 函数

将日期类型或时间戳类型的日期时间表达式转换为ISO8601格式的日期时间表达式。

语法：to\_iso8601(expr)

表 5-119 参数说明

| 参数名称 | 描述                   | 类型          | 是否必选 |
|------|----------------------|-------------|------|
| expr | timestamp类型的日期和时间表达式 | timestamp类型 | 是    |

返回值类型：String类型

示例：SELECT TO\_ISO8601(from\_unixtime(1626774758000))

表 5-120 查询分析结果

| 类型 | 查询语句                                     | 返回结果                          |
|----|------------------------------------------|-------------------------------|
| 场景 | TO_ISO8601(FROM_UNIXTIME(1626774758000)) | 2021-07-20T17:52:38.000+08:00 |

## timestamp\_to\_mills 函数

将timestamp类型的日期和时间表达式转换为毫秒级的时间戳。

语法：timestamp\_to\_mills(expr)

表 5-121 参数说明

| 参数名称 | 描述                   | 类型          | 是否必选 |
|------|----------------------|-------------|------|
| expr | timestamp类型的日期和时间表达式 | timestamp类型 | 是    |

返回值类型：Long类型

示例：select timestamp\_to\_mills(from\_unixtime(1626774758000))

表 5-122 查询分析结果

| 类型 | 查询语句                                             | 返回结果          |
|----|--------------------------------------------------|---------------|
| 场景 | timestamp_to_mills(from_unixtime(1626774758000)) | 1626774758000 |

to\_unixtime 函数

将timestamp类型的日期和时间表达式转换毫秒级的时间戳。

语法：to\_unixtime(expr)

表 5-123 参数说明

| 参数名称 | 描述                   | 类型        | 是否必选 |
|------|----------------------|-----------|------|
| expr | timestamp类型的日期和时间表达式 | timestamp | 是    |

返回值类型：Long类型

示例：SELECT to\_unixtime(from\_unixtime(1626774758000))

表 5-124 查询分析结果

| 类型 | 查询语句                                      | 返回结果       |
|----|-------------------------------------------|------------|
| 场景 | to_unixtime(from_unixtime(1626774758000)) | 1626774758 |

time\_ceil 函数

将timestamp或long类型的时间戳按固定周期以向上舍入的方式取整，将其作为新的时间戳返回。Period可以是任何ISO8601的周期，如P3M（季度）或PT12H（半天）。

time\_ceil(expr, period)

表 5-125 参数说明

| 参数名称   | 描述                                           | 类型             | 是否必选 |
|--------|----------------------------------------------|----------------|------|
| expr   | 日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TimeZone | timestamp或long | 是    |
| period | ISO8601的周期                                   | String         | 是    |

返回值类型：timestamp类型

示例1：SELECT time\_ceil(from\_unixtime(\_\_time),'PT1M')

表 5-126 查询分析结果

| 类型 | 查询语句                                                  | 返回结果                                     |
|----|-------------------------------------------------------|------------------------------------------|
| 场景 | SELECT<br>time_ceil(from_unixtime(__time),'P<br>T1M') | 2025-05-30 14:58:00.000<br>Asia/Shanghai |

Period描述：period的格式应为P[n]Y[n]M[n]DT[n]H[n]M[n]S或P[n]W。在这些表示中，【n】是日期和时间元素的数量。不需要前导零，但每个元素的最大位数应由通信各方商定。大写字母P、Y、M、W、D、T、H、M和S是每个日期和时间元素的指示符，不被替换。

- P是放置在持续时间表示的开始处的持续时间指示符（对于期间）。
- Y是日历年数值之后的年指示符。
- M是日历月数值之后的月份指示符。
- W是周数值之后的周指示符。
- D是日历天数数值之后的日期指示符。
- T是表示的时间分量之前的时间指示符。
- H是小时数值之后的小时指示符。
- M是分钟指示符，紧随分钟数的值。
- S是秒数值之后的第二个指示符。

例如，“P3Y6M4DT12H30M5S”表示“三年六个月四天十二小时三十分五秒”的持续时间。

更多信息，请访问[https://en.wikipedia.org/wiki/ISO\\_8601](https://en.wikipedia.org/wiki/ISO_8601)

time\_floor 函数

将timestamp或long类型的时间戳按时间戳返回。Period可以是任何ISO8601的周期，如P3M（季度）或PT12H（半天）。

语法：time\_floor(expr, period)

表 5-127 参数说明

| 参数名称   | 描述                                                       | 类型             | 是否必选 |
|--------|----------------------------------------------------------|----------------|------|
| expr   | timestamp类型的日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TimeZone | timestamp或long | 是    |
| period | ISO8601的周期                                               | String         | 是    |

返回值类型：timestamp类型

示例1：SELECT TIME\_FLOOR(from\_unixtime(1626774758000), 'PT2H')



表 5-128 查询分析结果

| 类型 | 查询语句                                             | 返回结果                                     |
|----|--------------------------------------------------|------------------------------------------|
| 场景 | TIME_FLOOR(from_unixtime(1626774758000), 'PT2H') | 2021-07-20 16:00:00.000<br>Asia/Shanghai |

ceil 函数

使用时间单位对时间戳进行向上舍入，单位可以是SECOND、MINUTE、 HOUR、 DAY、 WEEK、 MONTH、 QUARTER或 YEAR。

语法：ceil(expr, unit)

表 5-129 参数说明

| 参数名称 | 描述                                  | 类型        | 是否必选 |
|------|-------------------------------------|-----------|------|
| expr | 日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS | timestamp | 是    |
| unit | 时间单位                                | String    | 是    |

返回值类型：String类型

示例：SELECT CEIL('2023-04-20 11:28:31.770', 'DAY')

表 5-130 查询分析结果

| 类型 | 查询语句                                    | 返回结果                    |
|----|-----------------------------------------|-------------------------|
| 场景 | CEIL ('2023-04-20 11:28:31.770', 'DAY') | 2023-04-21 00:00:00.000 |

floor 函数

使用时间单位对时间戳进行向下舍入，单位可以是SECOND、MINUTE、 HOUR、 DAY、 WEEK、 MONTH、 QUARTER或 YEAR。

语法：floor(expr, unit)

表 5-131 参数说明

| 参数名称 | 描述                                  | 类型        | 是否必选 |
|------|-------------------------------------|-----------|------|
| expr | 日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS | timestamp | 是    |
| unit | 时间单位                                | String    | 是    |

返回值类型：String类型

示例：SELECT FLOOR('2023-04-20 11:28:31.770', 'DAY')

表 5-132 查询分析结果

| 类型 | 查询语句                                    | 返回结果                    |
|----|-----------------------------------------|-------------------------|
| 场景 | FLOOR('2023-04-20 11:28:31.770', 'DAY') | 2023-04-20 00:00:00.000 |

time\_shift 函数

将时间戳expr移动一个Period（步长时间）。Period可以是任何ISO8601的Period。

语法：time\_shift(expr, period, step, [timezone])

表 5-133 参数说明

| 参数名称     | 描述                                                        | 类型        | 是否必选 |
|----------|-----------------------------------------------------------|-----------|------|
| expr     | timestamp类型的日期和时间表达式，格式为 yyyy-MM-dd HH:mm:ss.SSS TimeZone | timestamp | 是    |
| period   | ISO8601的周期                                                | String    | 是    |
| step     | 步长                                                        | Integer   | 是    |
| timezone | 时区                                                        | String    | 否    |

返回值类型：String类型

示例：SELECT TIME\_SHIFT( from\_unixtime(\_\_time), 'P1D', 5),  
TIME\_SHIFT( from\_unixtime(\_\_time), 'P1D', 5,'+08:00')

表 5-134 查询分析结果

| 类型  | 查询语句                                                  | 返回结果                                  |
|-----|-------------------------------------------------------|---------------------------------------|
| 场景1 | TIME_SHIFT( from_unixtime(__time), 'P1D', 5)          | 2024-11-23 21:58:59.907 Asia/Shanghai |
| 场景2 | TIME_SHIFT( from_unixtime(__time), 'P1D', 5,'+08:00') | 2024-11-23 21:58:59.907 +08:00        |

timezone\_hour 函数

计算时区的小时偏移量。

语法：timezone\_hour(expr)

表 5-135 参数说明

| 参数名称 | 描述                                           | 类型        | 是否必选 |
|------|----------------------------------------------|-----------|------|
| expr | 日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TIMEZONE | timestamp | 否    |

返回值类型：Integer类型

示例：SELECT TIMEZONE\_HOUR(from\_unixtime(\_\_time))

表 5-136 查询分析结果

| 类型 | 查询语句                                           | 返回结果 |
|----|------------------------------------------------|------|
| 场景 | SELECT<br>TIMEZONE_HOUR(from_unixtime(__time)) | 8    |

timezone\_minute 函数

计算时区的分钟偏移量。

语法：timezone\_minute(expr)

表 5-137 参数说明

| 参数名称 | 描述                                           | 类型        | 是否必选 |
|------|----------------------------------------------|-----------|------|
| expr | 日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TIMEZONE | timestamp | 否    |

返回值类型：Integer类型

示例：SELECT TIMEZONE\_MINUTE(from\_unixtime(\_\_time))

表 5-138 查询分析结果

| 类型 | 查询语句                                   | 返回结果 |
|----|----------------------------------------|------|
| 场景 | TIMEZONE_MINUTE(from_unixtime(__time)) | 0    |

time\_format 函数

将毫秒级的时间戳或timestamp类型的日期和时间表达式expr，依据用户自定义的format参数以 **Joda DateTimeFormat**模式转换为指定日期和时间格式的字符串。

语法：time\_format(expr, format)

表 5-139 参数说明

| 参数名称   | 描述                           | 类型             | 是否必选 |
|--------|------------------------------|----------------|------|
| expr   | timestamp类型的日期和时间表达式或毫秒级的时间戳 | timestamp或Long | 是    |
| format | 时间格式。                        | String         | 是    |

返回值类型：String类型

示例：SELECT TIME\_FORMAT(\_\_time, 'yyyy-MM-dd HH:mm:ss'),  
TIME\_FORMAT(from\_unixtime(\_\_time), 'yyyy-MM-dd HH:mm:ss')

表 5-140 查询分析结果

| 类型  | 查询语句                                                      | 返回结果                |
|-----|-----------------------------------------------------------|---------------------|
| 场景1 | TIME_FORMAT(__time, 'yyyy-MM-dd HH:mm:ss')                | 2024-11-18 22:15:16 |
| 场景2 | TIME_FORMAT(from_unixtime(__time), 'yyyy-MM-dd HH:mm:ss') | 2024-11-18 22:15:16 |

time\_parse 函数

time\_parse将给定的字符串expr，依据用户自定义的format参数以 [Joda DateTimeFormat](#)模式解析为timestamp类型的日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TimeZone。若<format>未填写则以默认的ISO8601将字符串解析。

语法：time\_parse(expr, format)

表 5-141 参数说明

| 参数名称   | 描述          | 类型     | 是否必选 |
|--------|-------------|--------|------|
| expr   | 某种时间格式的字符串。 | String | 是    |
| format | 时间格式。       | String | 是    |

返回值类型：timestamp类型的时间和日志表达式

示例：SELECT TIME\_PARSE('2024-11-18 22:15:11', 'yyyy-MM-dd HH:mm:ss')

表 5-142 查询分析结果

| 类型 | 查询语句                                                     | 返回结果                                  |
|----|----------------------------------------------------------|---------------------------------------|
| 场景 | TIME_PARSE('2024-11-18 22:15:11', 'yyyy-MM-dd HH:mm:ss') | 2024-11-18 22:15:11.000 Asia/Shanghai |

format描述:

Symbol Meaning Presentation Examples

-----

G era text AD

C century of era (>=0) number 20

Y year of era (>=0) year 1996

x weekyear year 1996

w week of weekyear number 27

e day of week number 2

E day of week text Tuesday; Tue

y year year 1996

D day of year number 189

M month of year month July; Jul; 07

d day of month number 10

a halfday of day text PM

K hour of halfday (0~11) number 0

h clockhour of halfday (1~12) number 12

H hour of day (0~23) number 0

k clockhour of day (1~24) number 24

m minute of hour number 30

s second of minute number 55

S fraction of second millis 978

z time zone text Pacific Standard Time; PST

Z time zone offset/id zone -0800; -08:00; America/Los\_Angeles

' escape for text delimiter

" single quote literal '

date\_format 函数

timestamp类型的日期和时间表达式转换为指定日期和时间格式的字符串。

语法：date\_format(expr, format)

表 5-143 参数说明

| 参数名称   | 描述                                                        | 类型        | 是否必选 |
|--------|-----------------------------------------------------------|-----------|------|
| expr   | timestamp类型的日期和时间表达式，格式为 yyyy-MM-dd HH:mm:ss.SSS TimeZone | timestamp | 是    |
| format | 时间格式                                                      | String    | 是    |

返回值类型：String类型

示例：SELECT DATE\_FORMAT(current\_timestamp(),'%H%i')

表 5-144 查询分析结果

| 类型 | 查询语句                                    | 返回结果 |
|----|-----------------------------------------|------|
| 场景 | DATE_FORMAT(current_timestamp(),'%H%i') | 1432 |

format描述：

- %a Abbreviation for the week. For example, Sun and Sat.
- %b Abbreviation of the month. For example, Jan and Dec.
- %c Month. Numeral type. Range: 1-12.
- %D Day of the month. The value must be suffixed, for example, 0th, 1st, 2nd, and 3rd.
- %d Day of the month. The value ranges from 01 to 31 in decimal notation.
- %e Day of the month. The value ranges from 1 to 31 in decimal notation.
- %H Hour, 24-hour system.
- %h Hour, 12-hour system.
- %i Minute. Numeral type. Range: 00-59.
- %j Day of the year. The value ranges from 001 to 366.
- %k Hour. The value ranges from 0 to 23.
- %l Hour. The value ranges from 1 to 12.
- %M The English expression of the month, for example, January, December.

- %m Month. Numeral format. The value ranges from 01 to 12.
- %p AM and PM.
- %r Time in the 12-hour format. The format is hh:mm:ss AM/PM.
- %S Indicates the second. The value ranges from 00 to 59.
- %s Indicates the second. The value ranges from 00 to 59.
- %T Time, in the 24-hour format of hh:mm:ss.
- %v The first week of the year, Monday is the first day of the week. The value ranges from 01 to 53.
- %W The name of the day of the week. For example, Sunday and Saturday.
- %w Day of the week. Sunday is day 0.
- %Y A 4-digit year, for example, 2020.
- %y A 2-digit year, for example, 20.
- %% Escape character for%.

date\_parse 函数

将日期和时间字符串转换为指定格式的时间戳类型的日期和时间表达式。

语法：date\_parse(expr, pattern)

表 5-145 参数说明

| 参数名称    | 描述            | 类型     | 是否必选 |
|---------|---------------|--------|------|
| expr    | 日期和时间表达式。     | String | 是    |
| pattern | 日期和时间表达式的转换格式 | String | 是    |

返回值类型：timestamp类型的日期和时间表达式。

示例：SELECT DATE\_PARSE('2024-11-18 10:53','%Y-%m-%d %h:%i')

表 5-146 查询分析结果

| 类型 | 查询语句                                            | 返回结果                    |
|----|-------------------------------------------------|-------------------------|
| 场景 | DATE_PARSE('2024-11-18 10:53','%Y-%m-%d %h:%i') | 2024-11-18 18:53:00.000 |

pattern在设置时，如果设置了分钟，那么就必须同时配置分钟之前的年、月、日和时。

time\_extract 函数

通过指定字段提取日期时间表达式的日期或时间部分。

单位可以是纪元、秒、分钟、小时、天（月中的某天）、DOW（星期几）、DOY（一年中的某一天）、WEEK（一周中的一周）、MONTH（1 到 12）、QUARTER（1 到 4）或 YEAR。如果提供时区应为时区名称（如 “America/Los\_Angeles”）或偏移量（如 “-08: 00”）。

语法：time\_extract(expr, unit)

表 5-147 参数说明

| 参数名称 | 描述                                  | 类型     | 是否必选 |
|------|-------------------------------------|--------|------|
| expr | 日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS | String | 是    |
| unit | 时间单位                                | String | 是    |

返回值类型：Long类型

示例：SELECT TIME\_EXTRACT('2023-05-05','YEAR')

表 5-148 查询分析结果

| 类型 | 查询语句                              | 返回结果 |
|----|-----------------------------------|------|
| 场景 | TIME_EXTRACT('2023-05-05','YEAR') | 2023 |

date\_trunc 函数

根据您指定的时间单位截断日期和时间表达式，指定的时间单位有毫秒、秒、分钟、小时、天、月或年。

语法：date\_trunc(unit, expr)

表 5-149 参数说明

| 参数名称 | 描述                                                        | 类型        | 是否必选 |
|------|-----------------------------------------------------------|-----------|------|
| expr | timestamp类型的日期和时间表达式，格式为 yyyy-MM-dd HH:mm:ss.SSS TimeZone | timestamp | 是    |
| unit | 时间单位                                                      | String    | 是    |

返回值类型：Long类型

示例：SELECT DATE\_TRUNC('year', from\_unixtime(\_\_time))



表 5-150 查询分析结果

| 类型 | 查询语句                                         | 返回结果                                     |
|----|----------------------------------------------|------------------------------------------|
| 场景 | DATE_TRUNC('year',<br>from_unixtime(__time)) | 2024-01-01 00:00:00.000<br>Asia/Shanghai |

date\_diff 函数

返回时间戳expr1和时间戳expr2之间的单位数（有符号）。

语法：date\_diff(unit, expr1, expr2)

表 5-151 参数说明

| 参数名称  | 描述                                                           | 类型        | 是否必选 |
|-------|--------------------------------------------------------------|-----------|------|
| expr1 | timestamp类型的日期和时间表达式，格式为<br>yyyy-MM-dd HH:mm:ss.SSS TimeZone | timestamp | 是    |
| expr2 | timestamp类型的日期和时间表达式，格式为<br>yyyy-MM-dd HH:mm:ss.SSS TimeZone | timestamp |      |
| unit  | 时间单位                                                         | String    | 是    |

返回值类型：Long类型

示例：SELECT DATE\_DIFF('SECOND',  
CURRENT\_TIMESTAMP() ,from\_unixtime(\_\_time))

表 5-152 查询分析结果

| 类型 | 查询语句                                                                   | 返回结果    |
|----|------------------------------------------------------------------------|---------|
| 场景 | DATE_DIFF('SECOND',<br>CURRENT_TIMESTAMP() ,from_unixtime(__time)<br>) | -899.00 |

current\_date 函数

返回当前日期，格式是“yyyy-MM-dd”。

语法：current\_date()

返回值类型：String类型

示例：SELECT CURRENT\_DATE()

表 5-153 查询分析结果

| 类型 | 查询语句           | 返回结果       |
|----|----------------|------------|
| 场景 | CURRENT_DATE() | 2023-04-17 |

now 函数

返回当前日期和时间。功能与current\_timestamp一致

语法：now()

返回值类型：timestamp类型的日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TimeZone

示例：SELECT NOW()

表 5-154 查询分析结果

| 类型 | 查询语句  | 返回结果                                  |
|----|-------|---------------------------------------|
| 场景 | NOW() | 2024-11-18 23:29:18.434 Asia/Shanghai |

date\_add 函数

给时间加上给定的时间间隔。

语法：date\_add(unit, n, expr)

表 5-155 参数说明

| 参数名称 | 描述                                                                 | 类型        | 是否必选 |
|------|--------------------------------------------------------------------|-----------|------|
| unit | 时间单位，取值为millisecond、second、minute、hour、day、week、month、quarter、year | String    | 是    |
| n    | 时间间隔                                                               | Long      | 是    |
| expr | timestamp类型的日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TimeZone           | timestamp | 是    |

返回值类型：timestamp

示例：SELECT DATE\_ADD('MINUTE', 7, from\_unixtime(\_\_time))

表 5-156 查询分析结果

| 类型 | 查询语句                                         | 返回结果                                  |
|----|----------------------------------------------|---------------------------------------|
| 场景 | DATE_ADD('MINUTE', 7, from_unixtime(__time)) | 2024-11-18 23:23:47.407 Asia/Shanghai |

current\_time 函数

返回当前时间，格式为HH:mm:ss.SSSSSS。

语法：current\_time()

返回值类型：String类型

示例：SELECT CURRENT\_TIME()

表 5-157 查询分析结果

| 类型 | 查询语句            | 返回结果               |
|----|-----------------|--------------------|
| 场景 | CURRENT_TIME () | 23:38:18.491+08:00 |

current\_timezone 函数

返回当前时区。

语法：current\_timezone()

返回值类型：String类型

示例：SELECT CURRENT\_TIMEZONE()

表 5-158 查询分析结果

| 类型 | 查询语句                | 返回结果          |
|----|---------------------|---------------|
| 场景 | CURRENT_TIMEZONE () | Asia/Shanghai |

localtime 函数

返回本地时间。

语法：localtime()

返回值类型：String类型

示例：SELECT LOCALTIME()

表 5-159 查询分析结果

| 类型 | 查询语句        | 返回结果         |
|----|-------------|--------------|
| 场景 | LOCALTIME() | 23:38:51.346 |

localtimestamp 函数

返回本地的日期和时间。

语法：localtimestamp

返回值类型：String类型

示例：SELECT LOCALTIMESTAMP

表 5-160 查询分析结果

| 类型 | 查询语句           | 返回结果                    |
|----|----------------|-------------------------|
| 场景 | LOCALTIMESTAMP | 2024-11-18 23:39:17.070 |

year\_of\_week 函数

返回目标日期在ISO周日历中的年份。year\_of\_week函数等同于yow函数。

语法：year\_of\_week(expr)

表 5-161 参数说明

| 参数名称 | 描述                                          | 类型     | 是否必选 |
|------|---------------------------------------------|--------|------|
| expr | 时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS或yyyy-MM-dd | String | 是    |

返回值类型：Integer类型

示例：SELECT YEAR\_OF\_WEEK('2023-09-22 14:47:59.325'),  
YEAR\_OF\_WEEK('2023-09-22')

表 5-162 查询分析结果

| 类型  | 查询语句                                    | 返回结果 |
|-----|-----------------------------------------|------|
| 场景1 | YEAR_OF_WEEK('2023-09-22 14:47:59.325') | 2023 |
| 场景2 | YEAR_OF_WEEK('2023-09-22')              | 2023 |

表 5-163 参数说明

| 参数名称 | 描述                                                       | 类型        | 是否必选 |
|------|----------------------------------------------------------|-----------|------|
| expr | timestamp类型的日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TimeZone | timestamp | 是    |

返回值类型：Integer

示例：SELECT YEAR\_OF\_WEEK(from\_unixtime(\_\_time))

表 5-164 查询分析结果

| 类型 | 查询语句                                | 返回结果 |
|----|-------------------------------------|------|
| 场景 | YEAR_OF_WEEK(from_unixtime(__time)) | 2024 |

yow 函数

返回目标日期在ISO周日历中的年份。year\_of\_week函数等同于yow函数。

语法：yow(expr)

表 5-165 参数说明

| 参数名称 | 描述                                          | 类型     | 是否必选 |
|------|---------------------------------------------|--------|------|
| expr | 时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS或yyyy-MM-dd | String | 是    |

返回值类型：Integer类型

示例：SELECT YOW('2023-09-22 14:47:59.325'), YOW('2023-09-22')

表 5-166 查询分析结果

| 类型  | 查询语句                           | 返回结果 |
|-----|--------------------------------|------|
| 场景1 | YOW('2023-09-22 14:47:59.325') | 2023 |
| 场景2 | YOW('2023-09-22')              | 2023 |

表 5-167 参数说明

| 参数名称 | 描述                                          | 类型     | 是否必选 |
|------|---------------------------------------------|--------|------|
| expr | 时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS或yyyy-MM-dd | String | 是    |

返回值类型：Integer类型

示例：SELECT YOW(from\_unixtime(\_\_time))

表 5-168 查询分析结果

| 类型 | 查询语句                       | 返回结果 |
|----|----------------------------|------|
| 场景 | YOW(from_unixtime(__time)) | 2024 |

time\_series 函数

用于补全您查询时间窗口内缺失的数据。time\_series函数必须搭配GROUP BY语法和ORDER BY语法使用。

语法：time\_series(x, window\_time, format, padding\_data)

表 5-169 参数说明

| 参数名称         | 描述                                                                                                              | 类型             | 是否必选 |
|--------------|-----------------------------------------------------------------------------------------------------------------|----------------|------|
| x            | 原始字段。                                                                                                           | long或timestamp | 是    |
| window_time  | 窗口大小，单位为s（秒）、m（分）、h（小时）、d（天）。例如2h、5m、3d。                                                                        | String         | 是    |
| format       | 返回结果的时间格式。                                                                                                      | String         | 是    |
| padding_data | 补全的内容。包括：0：将缺失的值设置为0。null：将缺失的值设置为null。last：将缺失的值设置了上一个时间点对应的值。next：将缺失的值设置了下一个时间点对应的值。avg：将缺失的值设置为前后两个时间点的平均值。 | String         | 是    |

返回值类型：String

示例：select time\_series(\_\_time, '1d', 'HH:mm:ss','0') as time, count(\*) as count group by time order by time asc

表 5-170 查询分析结果

| 类型 | 查询语句                                                                                                        | 返回结果                                                                                                                                                                              |
|----|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 场景 | select time_series(__time, '1m', 'HH:mm:ss','0') as time, count(*) as count group by time order by time asc | <b>time count</b><br>7:04:00 5.00<br>7:05:00 6.00<br>7:06:00 6.00<br>7:07:00 6.00<br>7:08:00 6.00<br>7:09:00 6.00<br>7:10:00 6.00<br>7:11:00 6.00<br>7:12:00 6.00<br>7:13:00 6.00 |

5.10.4 正则式函数

本文介绍正则式函数的语法规则，包括参数解释、函数示例等。

函数列表

表 5-171 正则式函数

| 函数                                   | 描述                         |
|--------------------------------------|----------------------------|
| <a href="#">regexp_extract函数</a>     | 提取目标字符串中符合正则表达式的子字符串。      |
| <a href="#">regexp_like函数</a>        | 判断目标字符串是否符合正则表达式。          |
| <a href="#">regexp_extract_all函数</a> | 从目标字符串中提取符合正则表达式的子字符串。     |
| <a href="#">regexp_replace函数</a>     | 删除或替换目标字符串中符合正则表达式的全部子字符串。 |
| <a href="#">regexp_split函数</a>       | 分割目标字符串，返回被分割后的子字符串合集。     |

regexp\_extract 函数

提取目标字符串中符合正则表达式的子字符串。

- 提取并返回目标字符串中符合正则表达式的第一个子字符串。  
语法：regexp\_extract(expr, regularExpr)
- 提取并返回目标字符串中符合正则表达式的第n个子字符串。  
语法：regexp\_extract(expr, regularExpr, n)

表 5-172 参数说明

| 参数名称        | 描述                           | 类型      | 是否必选 |
|-------------|------------------------------|---------|------|
| expr        | 目标字符串。                       | String  | 是    |
| regularExpr | 包含捕获组的正则表达式，(\d)(\d)表示两个捕获组。 | String  | 是    |
| n           | 第n个符合正则表达式的子字符串。             | Integer | 否    |

返回值类型：String类型

示例：SELECT REGEXP\_EXTRACT('HTTP/2.0', '\d+')

表 5-173 查询分析结果

| 类型  | 查询语句                                  | 返回结果 |
|-----|---------------------------------------|------|
| 场景1 | REGEXP_EXTRACT('HTTP/2.0', '\d+')     | 2    |
| 场景2 | REGEXP_EXTRACT ('HTTP/2.0', '\d+', 1) | 2    |

regexp\_like 函数

判断目标字符串是否符合正则表达式。

语法：regexp\_like(expr, regularExpr)

表 5-174 参数说明

| 参数名称        | 描述                          | 类型     | 是否必选 |
|-------------|-----------------------------|--------|------|
| expr        | 目标字符串                       | String | 是    |
| regularExpr | 包含捕获组的正则表达式，(\d)(\d)表示两个捕获组 | String | 是    |

返回值类型：Boolean类型

示例：SELECT REGEXP\_LIKE('HTTP/2.0', '\d+')

表 5-175 查询分析结果

| 类型 | 查询语句                           | 返回结果 |
|----|--------------------------------|------|
| 场景 | REGEXP_LIKE('HTTP/2.0', '\d+') | true |



regexp\_extract\_all 函数

从目标字符串中提取符合正则表达式的子字符串。

语法：regexp\_extract\_all(expr, regularExpr)

表 5-176 参数说明

| 参数名称        | 描述                           | 类型     | 是否必选 |
|-------------|------------------------------|--------|------|
| expr        | 目标字符串                        | String | 是    |
| regularExpr | 包含捕获组的正则表达式，(\d) (\d)表示两个捕获组 | String | 是    |

返回值类型：Array类型

示例：SELECT REGEXP\_EXTRACT\_ALL('HTTP/2.0', '\d+')

表 5-177 查询分析结果

| 类型 | 查询语句                                   | 返回结果      |
|----|----------------------------------------|-----------|
| 场景 | REGEXP_EXTRACT_ALL ('HTTP/2.0', '\d+') | ["2","0"] |

regexp\_replace 函数

删除或替换目标字符串中符合正则表达式的全部子字符串。

- 删除目标字符串中符合正则表达式的子字符串，返回未被删除的子字符串。  
语法：regexp\_replace(expr, regularExpr)
- 替换目标字符串中符合正则表达式的子字符串，返回被替换后的字符串。  
语法：regexp\_replace(expr, regularExpr, replaceStr)

表 5-178 参数说明

| 参数名称        | 描述                           | 类型     | 是否必选 |
|-------------|------------------------------|--------|------|
| expr        | 目标字符串                        | String | 是    |
| regularExpr | 包含捕获组的正则表达式，(\d) (\d)表示两个捕获组 | String | 是    |
| replaceStr  | 替换的字符串                       | String | 否    |

返回值类型：String类型

示例：SELECT REGEXP\_REPLACE('ab12cd34', '\d+'), REGEXP\_REPLACE('ab12cd34', '\d+', '00')

表 5-179 查询分析结果

| 类型  | 查询语句                                    | 返回结果     |
|-----|-----------------------------------------|----------|
| 场景1 | REGEXP_REPLACE('ab12cd34', '\d+')       | abcd     |
| 场景2 | REGEXP_REPLACE('ab12cd34', '\d+', '00') | ab00cd00 |

regexp\_split 函数

分割目标字符串，返回被分割后的子字符串合集。

语法：regexp\_split(expr, regularExpr)

表 5-180 参数说明

| 参数名称        | 描述                           | 类型     | 是否必选 |
|-------------|------------------------------|--------|------|
| expr        | 目标字符串                        | String | 是    |
| regularExpr | 包含捕获组的正则表达式，(\d) (\d)表示两个捕获组 | String | 是    |

返回值类型：Array类型

示例：SELECT REGEXP\_SPLIT('request\_uri:/request/path-0/file-7','/')

表 5-181 查询分析结果

| 类型 | 查询语句                                                   | 返回结果                                            |
|----|--------------------------------------------------------|-------------------------------------------------|
| 场景 | REGEXP_SPLIT('request_uri:/request/path-0/file-7','/') | ["request_uri:", "request", "path-0", "file-7"] |

5.10.5 同比和环比函数

本文介绍同比和环比函数的语法规则，包括参数解释、函数示例等。

## 函数列表

表 5-182 同比和环比函数

| 函数                           | 描述                                                   |
|------------------------------|------------------------------------------------------|
| <a href="#">compare函数</a>    | 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。                     |
| <a href="#">ts_compare函数</a> | 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。该函数在结果数组中额外返回毫秒级时间戳。 |

### 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。  
`compare(x, n)`
- 对比当前时间周期内的计算结果与n1、n2、n3秒之前时间周期内的计算结果。  
`compare(x, n1, n2, n3...)`

表 5-183 参数说明

| 参数名称 | 描述                                                       | 类型          | 是否必选 |
|------|----------------------------------------------------------|-------------|------|
| x    | 需要对比的聚合函数结果字段。                                           | double或long | 是    |
| n?   | 时间窗口，单位为秒。例如3600（1小时）、86400（1天）、604800（1周）、31622400（1年）。 | long        | 是    |

### 返回类型

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]。

### 示例说明

计算当前1小时和昨天同时段的访问量比值。

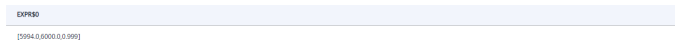
- 选择查询和分析的时间范围为**1小时（整点时间）**，并执行如下查询和分析语句。其中**86400**表示当前时间减去86400秒(1天)。

```
SELECT
 compare(PV, 86400)
FROM (SELECT count(*) AS PV)
```

- 查询和分析结果。

- 5994.0**表示当前1小时（例如2021-01-02 00:00:00~2021-01-02 01:00:00）的网站访问量。
- 6000.0**表示昨天同时段（例如2021-01-01 00:00:00~2021-01-01 01:00:00）的网站访问量。
- 0.999**表示当前1小时与昨天同时段的网站访问量比值。

### 图 5-29 查询和分析结果



3. 分列显示查询和分析结果。

```
SELECT
diff[1] as "today",
diff[2] as "yesterday",
diff[3] as "ratio"
FROM(SELECT compare(pv, 86400) AS diff FROM (SELECT count(*) AS pv))
```

### 图 5-30 查询和分析结果



## ts\_compare 函数

ts\_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。ts\_compare函数必须按照时间列进行分组（GROUP BY）。

## 语法格式

- 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。  
ts\_compare(x, n)
- 对比当前时间周期内的计算结果与n1、n2、n3秒之前时间周期内的计算结果。  
ts\_compare(x, n1, n2, n3...)

### 表 5-184 环比函数参数说明

| 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）。 |

### 返回类型

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, n秒前的UNIX时间戳]。

### 示例说明

环比今天3小时与昨天3小时的网站访问量。

选择查询和分析的时间范围为今天某3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），date\_trunc('hour',\_\_time)表示使用date\_trunc函数将时间对齐到小时。

- 查询和分析语句

```
* | SELECT
 t_time,
 ts_compare(PV, 86400) AS data
FROM(
 SELECT
 to_unixtime(date_trunc('hour', __time)) AS t_time,
 count(*) AS PV
 GROUP BY
```

```
 t_time
 ORDER BY
 t_time
)
GROUP BY
 t_time
```

- 查询和分析结果

| t_time                   | data                                         |
|--------------------------|----------------------------------------------|
| 2021-10-26T06:00:00.000Z | [159.0,224.0,0.7098214285714286,1.6351416E9] |
| 2021-10-26T07:00:00.000Z | [100.0,148.0,0.6756756756756757,1.6351452E9] |
| 2021-10-26T08:00:00.000Z | [100.0,100.0,1.0, 1.6016544E9, 1.6351488E9]  |

### 5.10.6 窗口函数

本文介绍窗口函数的语法规则，包括参数解释、函数示例等。

#### 函数列表

表 5-185 窗口函数

| 函数                      | 描述                  |
|-------------------------|---------------------|
| <a href="#">ntile函数</a> | 用于将窗口分区内数据按照顺序分成N组。 |

#### ntile 函数

用于将窗口分区内数据按照顺序分成N组。

语法：

```
ntile(n) over (
 [partition by partition_expression]
 [order by order_expression]
)
```

表 5-186 参数说明

| 参数名称                                 | 描述                             | 类型  | 是否必选 |
|--------------------------------------|--------------------------------|-----|------|
| n                                    | 组数。                            | int | 是    |
| partition by<br>partition_expression | 窗口分区，根据分区表达式将数据划分成不同的分区。       | 任意  | 是    |
| order by<br>order_expression         | 窗口排序，order by子句用于对窗口分区内的行进行排序。 | 任意  | 是    |

返回值类型：integer

示例：status,host,ntile(5) over (partition by status order by host) as n

表 5-187 查询分析结果

| 类型     | 查询语句                                                               | 返回结果                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
|--------|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|------|---|-----|------------------|---|-----|------------------|---|-----|------------------|---|-----|------------------|---|-----|------------------|---|-----|------------------|---|-----|------------------|---|-----|------------------|---|-----|------------------|---|-----|------------------|---|
| 场景     | status,host,ntile(5) over (partition by status order by host) as n | <table><thead><tr><th>status</th><th>host</th><th>n</th></tr></thead><tbody><tr><td>200</td><td>www.jp-mock1.com</td><td>1</td></tr><tr><td>200</td><td>www.jp-mock1.com</td><td>1</td></tr><tr><td>200</td><td>www.jp-mock1.com</td><td>2</td></tr><tr><td>200</td><td>www.jp-mock1.com</td><td>2</td></tr><tr><td>200</td><td>www.jp-mock1.com</td><td>3</td></tr><tr><td>200</td><td>www.jp-mock1.com</td><td>3</td></tr><tr><td>200</td><td>www.jp-mock1.com</td><td>4</td></tr><tr><td>200</td><td>www.jp-mock1.com</td><td>4</td></tr><tr><td>200</td><td>www.jp-mock1.com</td><td>5</td></tr><tr><td>200</td><td>www.jp-mock1.com</td><td>5</td></tr></tbody></table> | status | host | n | 200 | www.jp-mock1.com | 1 | 200 | www.jp-mock1.com | 1 | 200 | www.jp-mock1.com | 2 | 200 | www.jp-mock1.com | 2 | 200 | www.jp-mock1.com | 3 | 200 | www.jp-mock1.com | 3 | 200 | www.jp-mock1.com | 4 | 200 | www.jp-mock1.com | 4 | 200 | www.jp-mock1.com | 5 | 200 | www.jp-mock1.com | 5 |
| status | host                                                               | n                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |
| 200    | www.jp-mock1.com                                                   | 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |      |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |     |                  |   |

5.10.7 类型转换函数

本文介绍类型转换函数的语法规则，包括参数解释、函数示例等。

函数列表

表 5-188 类型转换函数

| 函数     | 描述                                 |
|--------|------------------------------------|
| cast函数 | 将值从一种数据类型转换为另一种数据类型，并将数据类型提供给动态参数。 |

cast 函数

将值从一种数据类型转换为另一种数据类型，并将数据类型提供给动态参数。

在允许表达式的任何地方都允许使用铸造表达式。

语法：CAST(Expression as Datatype)

表 5-189 参数说明

| 参数名称       | 描述       | 类型                                                                                              | 是否必选 |
|------------|----------|-------------------------------------------------------------------------------------------------|------|
| expression | 源类型的表达式。 | 任意                                                                                              | 是    |
| Datatype   | 目标类型。    | SQL数据类型，可选值为bigint、varchar、double、double precision、boolean、timestamp、decimal、integer、char、date。 | 是    |

表 5-190 索引数据类型和 SQL 数据类型的对应关系

| 索引的数据类型 | SQL的数据类型                |
|---------|-------------------------|
| long    | bigint                  |
| string  | varchar                 |
| float   | double或double precision |

返回值类型：目标数据类型。

示例：select cast(time as date)

表 5-191 查询分析结果

| 类型 | 查询语句               | 返回结果                    |
|----|--------------------|-------------------------|
| 场景 | cast(time as date) | 2023-08-31 23:11:17.000 |

5.10.8 数学计算函数

本文介绍数学计算函数的语法规则，包括参数解释、函数示例等。

函数列表

表 5-192 数学计算函数

| 函数                      | 描述                                            |
|-------------------------|-----------------------------------------------|
| <a href="#">round函数</a> | 用于对x进行四舍五入。如果n存在，则保留n位小数；如果n不存在，则对x进行四舍五入取整数。 |

round 函数

用于对x进行四舍五入。如果n存在，则保留n位小数；如果n不存在，则对x进行四舍五入取整数。

- 对x进行四舍五入取整数。  
语法：round(x)
- 对x进行四舍五入且保留n位小数。  
语法：round(x, n)

表 5-193 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 原始字段。      | double | 是    |
| n    | n位小数(int)。 | int    | 是    |

返回值类型：double

示例：select round(num,4)

示例数据num：3.1415926

表 5-194 查询分析结果

| 类型 | 查询语句                | 返回结果   |
|----|---------------------|--------|
| 场景 | select round(num,4) | 3.1416 |

5.10.9 URL 函数

本文介绍URL函数的语法规则，包括参数解释、函数示例等。

函数列表

表 5-195 URL 函数

| 函数                                      | 描述                 |
|-----------------------------------------|--------------------|
| <a href="#">url_encode函数</a>            | 对URL进行编码。          |
| <a href="#">url_decode函数</a>            | 对URL进行解码。          |
| <a href="#">url_extract_fragment函数</a>  | 提取URL中的fragment信息。 |
| <a href="#">url_extract_host函数</a>      | 提取URL中的host信息。     |
| <a href="#">url_extract_parameter函数</a> | 提取URL中的参数信息。       |
| <a href="#">url_extract_path函数</a>      | 提取URL中的path信息。     |
| <a href="#">url_extract_port函数</a>      | 提取URL中的端口信息。       |
| <a href="#">url_extract_protocol函数</a>  | 提取URL中的协议信息。       |
| <a href="#">url_extract_query函数</a>     | 提取URL中查询部分的信息。     |



url\_encode 函数

对URL进行编码。

语法：url\_encode(expr)

表 5-196 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| expr | URL字符串 | String | 是    |

返回值类型：String类型

示例：SELECT URL\_ENCODE('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 5-197 查询分析结果

| 类型 | 查询语句                                                                                         | 返回结果                                                                                               |
|----|----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| 场景 | URL_ENCODE('http://username:password@host:8080/index?parameterName=parameterValue#fragment') | http%3A%2F%2Fusername%3Apassword%40host%3A8080%2Findex%3FparameterName%3DparameterValue%23fragment |

url\_decode 函数

对URL进行解码。

语法：url\_decode(expr)

表 5-198 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| expr | URL字符串 | String | 是    |

返回值类型：String类型

示例：SELECT URL\_DECODE('http%3A%2F%2Fusername%3Apassword%40host%3A8080%2Findex%3FparameterName%3DparameterValue%23fragment')

表 5-199 查询分析结果

| 类型 | 查询语句                                                                                                             | 返回结果                                                                           |
|----|------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| 场景 | URL_DECODE('http%3A%2F%2Fusername%3Apassword%40host%3A8080%2Findex%3FparameterName%3DparameterValue%23fragment') | http://username:password@host:8080/index?parameterName=parameterValue#fragment |

url\_extract\_fragment 函数

提取URL中的fragment信息。

语法：url\_extract\_fragment(expr)

表 5-200 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| expr | URL字符串 | String | 是    |

返回值类型：String类型

示例：SELECT URL\_EXTRACT\_FRAGMENT('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 5-201 查询分析结果

| 类型 | 查询语句                                                                                                   | 返回结果     |
|----|--------------------------------------------------------------------------------------------------------|----------|
| 场景 | URL_EXTRACT_FRAGMENT('http://username:password@host:8080/index?parameterName=parameterValue#fragment') | fragment |

url\_extract\_host 函数

提取URL中的host信息。

语法：url\_extract\_host(expr)

表 5-202 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| expr | URL字符串 | String | 是    |

返回值类型：String类型

示例：SELECT URL\_EXTRACT\_HOST('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 5-203 查询分析结果

| 类型 | 查询语句                                                                                               | 返回结果 |
|----|----------------------------------------------------------------------------------------------------|------|
| 场景 | URL_EXTRACT_HOST('http://username:password@host:8080/index?parameterName=parameterValue#fragment') | host |

url\_extract\_parameter 函数

提取URL中的参数信息。

语法：url\_extract\_parameter(expr, paramName)

表 5-204 参数说明

| 参数名称      | 描述     | 类型     | 是否必选 |
|-----------|--------|--------|------|
| expr      | URL字符串 | String | 是    |
| paramName | 参数名称   | String | 是    |

返回值类型：String类型

示例：SELECT URL\_EXTRACT\_PARAMETER('http://username:password@host:8080/index?parameterName=parameterValue#fragment', 'parameterName')

表 5-205 查询分析结果

| 类型 | 查询语句                                                                                                                     | 返回结果           |
|----|--------------------------------------------------------------------------------------------------------------------------|----------------|
| 场景 | URL_EXTRACT_PARAMETER('http://username:password@host:8080/index?parameterName=parameterValue#fragment', 'parameterName') | parameterValue |

url\_extract\_path 函数

提取URL中的path信息。

语法：url\_extract\_path(expr)

表 5-206 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| expr | URL字符串 | String | 是    |

返回值类型：

String类型

示例：SELECT URL\_EXTRACT\_PATH('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 5-207 查询分析结果

| 类型 | 查询语句                                                                                               | 返回结果   |
|----|----------------------------------------------------------------------------------------------------|--------|
| 场景 | URL_EXTRACT_PATH('http://username:password@host:8080/index?parameterName=parameterValue#fragment') | /index |

## url\_extract\_port 函数

提取URL中的端口信息。

语法：url\_extract\_port(expr)

表 5-208 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| expr | URL字符串 | String | 是    |

返回值类型：String类型

示例：SELECT URL\_EXTRACT\_PORT('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 5-209 查询分析结果

| 类型 | 查询语句                                                                                               | 返回结果 |
|----|----------------------------------------------------------------------------------------------------|------|
| 场景 | URL_EXTRACT_PORT('http://username:password@host:8080/index?parameterName=parameterValue#fragment') | 8080 |

url\_extract\_protocol 函数

提取URL中的协议信息。

语法：url\_extract\_protocol(expr)

表 5-210 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| expr | URL字符串 | String | 是    |

返回值类型：String类型

示例：SELECT URL\_EXTRACT\_PROTOCOL('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 5-211 查询分析结果

| 类型 | 查询语句                                                                                                   | 返回结果 |
|----|--------------------------------------------------------------------------------------------------------|------|
| 场景 | URL_EXTRACT_PROTOCOL('http://username:password@host:8080/index?parameterName=parameterValue#fragment') | http |

url\_extract\_query 函数

提取URL中的查询部分的信息。

语法：url\_extract\_query(expr)

表 5-212 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| expr | URL字符串 | String | 是    |

返回值类型：String类型

示例：SELECT URL\_EXTRACT\_QUERY('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 5-213 查询分析结果

| 类型 | 查询语句                                                                                                | 返回结果                         |
|----|-----------------------------------------------------------------------------------------------------|------------------------------|
| 场景 | URL_EXTRACT_QUERY('http://username:password@host:8080/index?parameterName=parameterValue#fragment') | parameterName=parameterValue |

5.10.10 IP 地址函数

本文介绍IP地址函数的语法规则，包括参数解释、函数示例等。

函数列表

表 5-214 IP 地址函数

| 函数                                     | 描述                     |
|----------------------------------------|------------------------|
| <a href="#">ip_to_domain函数</a>         | 判断目标IPv4地址是内网地址还是外网地址。 |
| <a href="#">ip_prefix函数</a>            | 获取目标IPv4地址的前缀。         |
| <a href="#">is_prefix_subnet_of函数</a>  | 判断目标IPv4网段是否为某网段的子网。   |
| <a href="#">is_subnet_of函数</a>         | 判断目标IPv4地址是否在某网段内。     |
| <a href="#">ip_subnet_max函数</a>        | 获取IPv4网段中的最大IP地址。      |
| <a href="#">ip_subnet_min函数</a>        | 获取IPv4网段中的最小IP地址。      |
| <a href="#">ip_subnet_range函数</a>      | 获取IPv4网段范围。            |
| <a href="#">ip_to_city函数</a>           | 返回IPv4地址所在城市的名称。       |
| <a href="#">ip_to_provider函数</a>       | 返回IPv4地址的网络运营商的名称。     |
| <a href="#">ipv6_to_country函数</a>      | 返回IPv6地址的网络运营商的名称。     |
| <a href="#">ipv6_to_country_code函数</a> | 返回IPv6地址所在国家编码。        |
| <a href="#">ipv6_to_province函数</a>     | 返回IPv6地址所在省级行政区名称。     |
| <a href="#">ipv6_to_city函数</a>         | 返回IPv6地址所在城市的名称。       |
| <a href="#">ipv6_to_provider函数</a>     | 返回IPv6地址对应的网络服务供应商。    |
| <a href="#">ipv6_to_city_geo函数</a>     | 返回IPv6地址对应城市的经纬度信息。    |

| 函数                               | 描述                  |
|----------------------------------|---------------------|
| <a href="#">ipv6_to_geo函数</a>    | 返回IPv6地址对应城市的经纬度信息。 |
| <a href="#">ipv6_to_domain函数</a> | 判断是否为内网地址。          |

## ip\_to\_domain 函数

判断目标IPv4地址是内网地址还是外网地址。

语法：ip\_to\_domain(expr)

表 5-215 参数说明

| 参数名称 | 描述   | 类型     | 是否必选 |
|------|------|--------|------|
| expr | IP地址 | String | 是    |

返回值类型：String类型

示例：SELECT IP\_TO\_DOMAIN('10.110.10.210'), IP\_TO\_DOMAIN('192.175.4.1')

表 5-216 查询分析结果

| 类型  | 查询语句                          | 返回结果     |
|-----|-------------------------------|----------|
| 场景1 | IP_TO_DOMAIN('10.110.10.210') | intranet |
| 场景2 | IP_TO_DOMAIN('192.175.4.1')   | internet |

## ip\_prefix 函数

获取目标IPv4地址的前缀。

语法：ip\_prefix(expr, prefixBit)

表 5-217 参数说明

| 参数名称      | 描述     | 类型     | 是否必选 |
|-----------|--------|--------|------|
| expr      | IPv4地址 | String | 是    |
| prefixBit | 前缀位数   | String | 是    |

返回值类型：String类型

示例：SELECT IP\_PREFIX('10.110.10.210', 8), IP\_PREFIX('144.101.32.5', 12)

表 5-218 查询分析结果

| 类型  | 查询语句                          | 返回结果          |
|-----|-------------------------------|---------------|
| 场景1 | IP_PREFIX('10.110.10.210', 8) | 10.0.0.0/8    |
| 场景2 | IP_PREFIX('144.101.32.5', 12) | 144.96.0.0/12 |

is\_prefix\_subnet\_of 函数

判断目标IPv4网段是否为某网段的子网。

语法：is\_prefix\_subnet\_of(expr1, expr2)

表 5-219 参数说明

| 参数名称  | 描述        | 类型     | 是否必选 |
|-------|-----------|--------|------|
| expr1 | 给定的IPv4网段 | String | 是    |
| expr2 | 目标IPv4网段  | String | 是    |

返回值类型：Boolean类型

示例：SELECT IS\_PREFIX\_SUBNET\_OF('192.168.0.1/24', '192.168.1.1/24')

表 5-220 查询分析结果

| 类型 | 查询语句                                                    | 返回结果  |
|----|---------------------------------------------------------|-------|
| 场景 | IS_PREFIX_SUBNET_OF('192.168.0.1/24', '192.168.1.1/24') | false |

is\_subnet\_of 函数

判断目标IPv4地址是否在某网段内。

语法：is\_subnet\_of(expr1, expr2)

表 5-221 参数说明

| 参数名称  | 描述        | 类型     | 是否必选 |
|-------|-----------|--------|------|
| expr1 | 给定的IPv4网段 | String | 是    |
| expr2 | 目标IPv4地址  | String | 是    |

返回值类型：Boolean类型



示例：SELECT IS\_SUBNET\_OF('192.168.0.1/24', '192.168.1.1')

表 5-222 查询分析结果

| 类型 | 查询语句                                          | 返回结果  |
|----|-----------------------------------------------|-------|
| 场景 | IS_SUBNET_OF('192.168.0.1/24', '192.168.1.1') | false |

ip\_subnet\_max 函数

获取IPv4网段中的最大IP地址。

语法：ip\_subnet\_max(expr)

表 5-223 参数说明

| 参数名称 | 描述     | 类型     | 是否必选 |
|------|--------|--------|------|
| expr | IPv4网段 | String | 是    |

返回值类型：String类型

示例：SELECT IP\_SUBNET\_MAX('192.120.80.128/10')

表 5-224 查询分析结果

| 类型 | 查询语句                               | 返回结果            |
|----|------------------------------------|-----------------|
| 场景 | IP_SUBNET_MAX('192.120.80.128/10') | 192.127.255.255 |

ip\_subnet\_min 函数

获取IPv4网段中的最小IP地址。

语法：ip\_subnet\_min(expr)

表 5-225 参数说明

| 参数名称  | 描述     | 类型     | 是否必选 |
|-------|--------|--------|------|
| expr1 | IPv4网段 | String | 是    |

返回值类型：String类型

示例：SELECT IP\_SUBNET\_MIN('192.120.80.128/10')

表 5-226 查询分析结果

| 类型 | 查询语句                               | 返回结果       |
|----|------------------------------------|------------|
| 场景 | IP_SUBNET_MIN('192.120.80.128/10') | 192.64.0.0 |

ip\_subnet\_range 函数

获取IPv4网段范围。

语法：ip\_subnet\_range(expr)

表 5-227 参数说明

| 参数名称  | 描述     | 类型     | 是否必选 |
|-------|--------|--------|------|
| expr1 | IPv4网段 | String | 是    |

返回值类型：String类型

示例：SELECT IP\_SUBNET\_RANGE('192.120.80.128/10')

表 5-228 查询分析结果

| 类型 | 查询语句                                      | 返回结果                             |
|----|-------------------------------------------|----------------------------------|
| 场景 | IP_SUBNET_RANGE<br>( '192.120.80.128/10') | ["192.64.0.0","192.127.255.255"] |

在计算给定网段的最大IP地址、最小IP地址和IP范围时，排除了网段的单播和广播地址，计算出的IP地址为网段的最大可用IP地址、最小可用IP地址和可用IP地址范围。

ip\_to\_city 函数

返回城市的名称。

语法：ip\_to\_city(x)

表 5-229 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv4地址 | String | 是    |

返回值类型：String类型

示例：select ip\_to\_city(ip)

表 5-230 查询分析结果

| 类型 | 查询语句           | 返回结果    |
|----|----------------|---------|
| 场景 | ip_to_city(ip) | Cochise |

ip\_to\_provider 函数

返回IPv4地址的网络运营商的名称。

语法：ip\_to\_provider(x)

表 5-231 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv4地址 | String | 是    |

返回值类型：String类型

示例：select ip\_to\_provider(ip)

表 5-232 查询分析结果

| 类型 | 查询语句               | 返回结果      |
|----|--------------------|-----------|
| 场景 | ip_to_provider(ip) | CONUS-YPG |

ipv6\_to\_country 函数

返回IPv6地址的网络运营商的名称。

语法：ipv6\_to\_country(x)

表 5-233 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv6地址 | String | 是    |

返回值类型：String类型

示例：select ipv6\_to\_country(ip)

表 5-234 查询分析结果

| 类型 | 查询语句                | 返回结果 |
|----|---------------------|------|
| 场景 | ipv6_to_country(ip) | 中国   |

ipv6\_to\_country\_code 函数

返回IPv6地址所在国家编码。

语法：ipv6\_to\_country\_code(x)

表 5-235 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv6地址 | String | 是    |

返回值类型：String类型

示例：select ipv6\_to\_country\_code(ip)

表 5-236 查询分析结果

| 类型 | 查询语句                     | 返回结果 |
|----|--------------------------|------|
| 场景 | ipv6_to_country_code(ip) | CN   |

ipv6\_to\_province 函数

返回IPv6地址所在省级行政区名称。

语法：ipv6\_to\_province(x)

表 5-237 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv6地址 | String | 是    |

返回值类型：String类型

示例：select ipv6\_to\_province(ip)

表 5-238 查询分析结果

| 类型 | 查询语句                 | 返回结果 |
|----|----------------------|------|
| 场景 | ipv6_to_province(ip) | 陕西省  |

ipv6\_to\_city 函数

返回IPv6地址所在城市的名称。

语法：ipv6\_to\_city(x)

表 5-239 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv6地址 | String | 是    |

返回值类型：String类型

示例：select ipv6\_to\_city(ip)

表 5-240 查询分析结果

| 类型 | 查询语句             | 返回结果 |
|----|------------------|------|
| 场景 | ipv6_to_city(ip) | 西安市  |

ipv6\_to\_provider 函数

返回IPv6地址对应的网络服务供应商。

语法：ipv6\_to\_provider(x)

表 5-241 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv6地址 | String | 是    |

返回值类型：String类型

示例：select ipv6\_to\_provider(ip)

表 5-242 查询分析结果

| 类型 | 查询语句                 | 返回结果 |
|----|----------------------|------|
| 场景 | ipv6_to_provider(ip) | 中国电信 |

ipv6\_to\_city\_geo 函数

返回IPv6地址对应城市的经纬度信息。

语法：ipv6\_to\_city\_geo(x)

表 5-243 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv6地址 | String | 是    |

返回值类型：String类型

示例：select ipv6\_to\_city\_geo(ip)

表 5-244 查询分析结果

| 类型 | 查询语句                 | 返回结果                 |
|----|----------------------|----------------------|
| 场景 | ipv6_to_city_geo(ip) | 39.904989,116.405285 |

ipv6\_to\_geo 函数

返回IPv6地址对应城市的经纬度信息。

语法：ipv6\_to\_geo(x)

表 5-245 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv6地址 | String | 是    |

返回值类型：String类型

示例：select ipv6\_to\_geo(ip)

表 5-246 查询分析结果

| 类型 | 查询语句            | 返回结果                 |
|----|-----------------|----------------------|
| 场景 | ipv6_to_geo(ip) | 39.904989,116.405285 |

ipv6\_to\_domain 函数

判断是否为内网地址。

语法：ipv6\_to\_domain(x)

表 5-247 参数说明

| 参数名称 | 描述         | 类型     | 是否必选 |
|------|------------|--------|------|
| x    | 参数值为IPv6地址 | String | 是    |

返回值类型：String类型

示例：select ipv6\_to\_domain(ip)

表 5-248 查询分析结果

| 类型 | 查询语句               | 返回结果     |
|----|--------------------|----------|
| 场景 | ipv6_to_domain(ip) | internet |

5.10.11 电话号码函数

本文介绍电话号码函数的语法规则，包括参数解释、函数示例等。

函数列表

表 5-249 电话号码函数

| 函数                                | 描述            |
|-----------------------------------|---------------|
| <a href="#">mobile_carrier函数</a>  | 分析电话号码所属的运营商。 |
| <a href="#">mobile_city函数</a>     | 分析电话号码所属的城市。  |
| <a href="#">mobile_province函数</a> | 分析电话号码所属的省份。  |

mobile\_carrier 函数

分析电话号码所属的运营商。

语法：mobile\_carrier(expr)

表 5-250 参数说明

| 参数名称 | 描述   | 类型     | 是否必选 |
|------|------|--------|------|
| expr | 电话号码 | String | 是    |

返回值类型：String类型

示例：SELECT MOBILE\_CARRIER('17052294531')

表 5-251 查询分析结果

| 类型 | 查询语句                          | 返回结果 |
|----|-------------------------------|------|
| 场景 | MOBILE_CARRIER('17052294531') | 中国移动 |

## mobile\_city 函数

分析电话号码所属的城市。

语法：mobile\_city(expr)

表 5-252 参数说明

| 参数名称 | 描述   | 类型     | 是否必选 |
|------|------|--------|------|
| expr | 电话号码 | String | 是    |

返回值类型：String类型

示例：SELECT MOBILE\_CITY('17052294531')

表 5-253 查询分析结果

| 类型 | 查询语句                       | 返回结果 |
|----|----------------------------|------|
| 场景 | MOBILE_CITY('17052294531') | 西安市  |

## mobile\_province 函数

分析电话号码所属的省份。

语法：mobile\_province(expr)



表 5-254 参数说明

| 参数名称 | 描述   | 类型     | 是否必选 |
|------|------|--------|------|
| expr | 电话号码 | String | 是    |

返回值类型：String类型

示例：SELECT MOBILE\_PROVINCE('17052294531')

表 5-255 查询分析结果

| 类型 | 查询语句                           | 返回结果 |
|----|--------------------------------|------|
| 场景 | MOBILE_PROVINCE('17052294531') | 陕西省  |

5.10.12 数组函数

本文介绍数组函数的语法规则，包括参数解释、函数示例等。

函数列表

表 5-256 数组函数

| 函数                               | 描述                                    |
|----------------------------------|---------------------------------------|
| <a href="#">array函数</a>          | 将输入的参数构造成数组，参数类型必须相同。                 |
| <a href="#">array_position函数</a> | 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。      |
| <a href="#">cardinality函数</a>    | 计算数组中元素的个数。                           |
| <a href="#">mv_length函数</a>      | 计算数组中元素的个数，同cardinality。              |
| <a href="#">contains函数</a>       | 判断数组中是否包含指定元素。如果包含，则返回true。           |
| <a href="#">mv_contains函数</a>    | 判断数组中是否包含指定元素。如果包含，则返回true，同contains。 |
| <a href="#">mv_prepend函数</a>     | 将指定的元素添加到数组的开始位置。                     |
| <a href="#">mv_append函数</a>      | 将指定的元素添加到数组的末尾。                       |
| <a href="#">mv_slice函数</a>       | 返回从start到end索引的数组。                    |
| <a href="#">mv_to_string函数</a>   | 使用指定的分隔符str连接arr所有元素。                 |
| <a href="#">string_to_mv函数</a>   | 使用指定的分隔符str2将str1拆分为数组                |
| <a href="#">mv_offset函数</a>      | 返回所提供的基于0的索引处的数组元素。                   |
| <a href="#">mv_ordinal函数</a>     | 返回所提供的基于1的索引处的数组元素。                   |

| 函数                              | 描述                                 |
|---------------------------------|------------------------------------|
| <a href="#">mv_offset_of函数</a>  | 返回数组中第一次出现expr的基于0的索引，如果未出现，则返回-1。 |
| <a href="#">mv_ordinal_of函数</a> | 返回数组中第一次出现expr的基于1的索引，如果未出现，则返回-1。 |

array 函数

将参数构建成为数组，参数类型必须相同。

语法：array[expr1,expr ...]

表 5-257 参数说明

| 参数名称 | 描述   | 类型                               | 是否必选 |
|------|------|----------------------------------|------|
| expr | 原始数据 | String/Integer/Long/Double/Float | 是    |

返回值类型：Array类型

示例：SELECT ARRAY['1','2','3','4','5']

表 5-258 查询分析结果

| 类型 | 查询语句                       | 返回结果                      |
|----|----------------------------|---------------------------|
| 场景 | ARRAY['1','2','3','4','5'] | ["1", "2", "3", "4", "5"] |

array\_position 函数

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(expr, ele)

表 5-259 参数说明

| 参数名称 | 描述    | 类型                     | 是否必选 |
|------|-------|------------------------|------|
| expr | 原始数组  | 数组类型。                  | 是    |
| ele  | 指定的元素 | 数组中的一个元素，必须与数组中元素类型相同。 | 是    |

返回值类型：Integer类型

示例：SELECT ARRAY\_POSITION(ARRAY['1','2','3'],'2')

表 5-260 查询分析结果

| 类型 | 查询语句                                   | 返回结果 |
|----|----------------------------------------|------|
| 场景 | ARRAY_POSITION(ARRAY['1','2','3'],'2') | 2    |

cardinality 函数

计算数组中元素的个数，参数类型必须相同。

语法：cardinality(expr)

表 5-261 参数说明

| 参数名称 | 描述   | 类型                          | 是否必选 |
|------|------|-----------------------------|------|
| expr | 原始数组 | Array ( String/<br>Number ) | 是    |

返回值类型：Integer类型

示例：SELECT CARDINALITY(ARRAY['1','2','3'])

表 5-262 查询分析结果

| 类型 | 查询语句                            | 返回结果 |
|----|---------------------------------|------|
| 场景 | CARDINALITY(ARRAY['1','2','3']) | 3    |

mv\_length 函数

计算数组中元素的个数，同cardinality。

语法：mv\_length(expr)

表 5-263 参数说明

| 参数名称 | 描述   | 类型                          | 是否必选 |
|------|------|-----------------------------|------|
| expr | 原始数组 | Array ( String/<br>Number ) | 是    |

返回值类型：Integer类型

示例：SELECT MV\_LENGTH(ARRAY['1','2','3'])

表 5-264 查询分析结果

| 类型 | 查询语句                           | 返回结果 |
|----|--------------------------------|------|
| 场景 | MV_LENGTH (ARRAY['1','2','3']) | 3    |

contains 函数

判断数组中是否包含指定元素。如果包含，则返回true。

语法：contains(expr, ele)

表 5-265 参数说明

| 参数名称 | 描述    | 类型                          | 是否必选 |
|------|-------|-----------------------------|------|
| expr | 原始数组  | Array ( String/ Number )    | 是    |
| ele  | 指定的元素 | String/Number，必须与数组中元素类型相同。 | 是    |

返回值类型：Boolean类型

示例：SELECT CONTAINS(ARRAY['1','2'],'1')

表 5-266 查询分析结果

| 类型 | 查询语句                         | 返回结果 |
|----|------------------------------|------|
| 场景 | CONTAINS(ARRAY['1','2'],'1') | true |

mv\_contains 函数

判断数组中是否包含指定元素。如果包含，则返回true，同contains。

语法：mv\_contains(expr, ele)

表 5-267 参数说明

| 参数名称 | 描述    | 类型                          | 是否必选 |
|------|-------|-----------------------------|------|
| expr | 原始数组  | Array ( String/ Number )    | 是    |
| ele  | 指定的元素 | String/Number，必须与数组中元素类型相同。 | 是    |

返回值类型：Boolean类型

示例：SELECT MV\_CONTAINS(ARRAY['1','2'],'1')

表 5-268 查询分析结果

| 类型 | 查询语句                            | 返回结果 |
|----|---------------------------------|------|
| 场景 | MV_CONTAINS(ARRAY['1','2'],'1') | true |

mv\_prepend 函数

将指定的元素添加到数组的开始位置。

语法：mv\_prepend(expr, arr)

表 5-269 参数说明

| 参数名称 | 描述    | 类型                          | 是否必选 |
|------|-------|-----------------------------|------|
| expr | 指定的元素 | String/Number，必须与数组中元素类型相同。 | 是    |
| arr  | 原始数组  | Array ( String/ Number )    | 是    |

返回值类型：Array类型

示例：SELECT MV\_PREPEND('1', ARRAY ['1','2'])

表 5-270 查询分析结果

| 类型 | 查询语句                             | 返回结果          |
|----|----------------------------------|---------------|
| 场景 | MV_PREPEND ('1', ARRAY['1','2']) | ["1","1","2"] |

mv\_append 函数

将指定的元素添加到数组的末尾。

语法：mv\_append(arr, expr)

表 5-271 参数说明

| 参数名称 | 描述   | 类型                       | 是否必选 |
|------|------|--------------------------|------|
| arr  | 原始数组 | Array ( String/ Number ) | 是    |

| 参数名称 | 描述    | 类型                          | 是否必选 |
|------|-------|-----------------------------|------|
| expr | 指定的元素 | String/Number，必须与数组中元素类型相同。 | 是    |

返回值类型：Array类型

示例：SELECT MV\_APPEND(ARRAY['1','2'],'1')

表 5-272 查询分析结果

| 类型 | 查询语句                           | 返回结果          |
|----|--------------------------------|---------------|
| 场景 | MV_APPEND(ARRAY['1','2'], '1') | ["1","2","1"] |

mv\_slice 函数

返回从start到end索引的数组。

语法：mv\_slice(arr, start, end)

表 5-273 参数说明

| 参数名称  | 描述   | 类型                       | 是否必选 |
|-------|------|--------------------------|------|
| arr   | 原始数组 | Array ( String/ Number ) | 是    |
| start | 起始位置 | Integer                  | 是    |
| end   | 结束位置 | Integer                  | 是    |

返回值类型：Array类型

示例：SELECT MV\_SLICE(ARRAY['1','2','3','4','5'], 2, 4)

表 5-274 查询分析结果

| 类型 | 查询语句                                       | 返回结果      |
|----|--------------------------------------------|-----------|
| 场景 | MV_SLICE(ARRAY['1','2','3','4','5'], 2, 4) | ["3","4"] |

mv\_to\_string 函数

使用指定的分隔符str连接arr所有元素。

语法：mv\_to\_string(arr, str)

表 5-275 参数说明

| 参数名称 | 描述    | 类型                          | 是否必选 |
|------|-------|-----------------------------|------|
| arr  | 原始数组  | Array ( String/<br>Number ) | 是    |
| str  | 指定分隔符 | String                      | 是    |

返回值类型：String类型

示例：SELECT MV\_TO\_STRING(ARRAY['1','2','3','4','5'],'-')

表 5-276 查询分析结果

| 类型 | 查询语句                                         | 返回结果      |
|----|----------------------------------------------|-----------|
| 场景 | MV_TO_STRING(ARRAY['1','2','3','4','5'],'-') | 1-2-3-4-5 |

string\_to\_mv 函数

使用指定的分隔符str2将str1拆分为数组。

语法：string\_to\_mv(str1, str2)

表 5-277 参数说明

| 参数名称 | 描述    | 类型     | 是否必选 |
|------|-------|--------|------|
| str1 | 原始字符串 | String | 是    |
| str2 | 指定分隔符 | String | 是    |

返回值类型：Array类型

示例：SELECT STRING\_TO\_MV('1-2-3-4-5','-')

表 5-278 查询分析结果

| 类型 | 查询语句                          | 返回结果                  |
|----|-------------------------------|-----------------------|
| 场景 | STRING_TO_MV('1-2-3-4-5','-') | ["1","2","3","4","5"] |

mv\_offset 函数

返回所提供的基于0的索引处的数组元素，或对于超出范围的索引返回null。

语法：mv\_offset(arr, index)

表 5-279 参数说明

| 参数名称  | 描述     | 类型                          | 是否必选 |
|-------|--------|-----------------------------|------|
| arr   | 原始数组   | Array ( String/<br>Number ) | 是    |
| index | 指定索引位置 | Integer                     | 是    |

返回值类型：String/Integer/Long/Boolean/Double类型

示例：SELECT MV\_OFFSET(ARRAY['1','2','3','4','5'], 2)

表 5-280 查询分析结果

| 类型 | 查询语句                                     | 返回结果 |
|----|------------------------------------------|------|
| 场景 | MV_OFFSET(ARRAY['1','2','3','4','5'], 2) | 3    |

## mv\_ordinal 函数

返回所提供的基于1的索引处的数组元素，或对于超出范围的索引返回null。

语法：mv\_ordinal(arr, index)

表 5-281 参数说明

| 参数名称  | 描述     | 类型                          | 是否必选 |
|-------|--------|-----------------------------|------|
| arr   | 原始数组   | Array ( String/<br>Number ) | 是    |
| index | 指定索引位置 | Integer                     | 是    |

返回值类型：String/Integer/Long/Boolean/Double类型

示例：SELECT MV\_ORDINAL(ARRAY['1','2','3','4','5'], 2)

表 5-282 查询分析结果

| 类型 | 查询语句                                       | 返回结果 |
|----|--------------------------------------------|------|
| 场景 | MV_ORDINAL (ARRAY['1','2','3','4','5'], 2) | 2    |

## mv\_offset\_of 函数

返回数组中第一次出现expr的基于0的索引，如果未出现，则返回-1。



语法：mv\_offset\_of(arr, expr)

表 5-283 参数说明

| 参数名称 | 描述   | 类型                          | 是否必选 |
|------|------|-----------------------------|------|
| arr  | 原始数组 | Array ( String/ Number )    | 是    |
| expr | 指定元素 | String/Number，必须与数组中元素类型相同。 | 是    |

返回值类型：Integer类型

示例：SELECT MV\_OFFSET\_OF(ARRAY['1','2','3','4','5'], '2')

表 5-284 查询分析结果

| 类型 | 查询语句                                          | 返回结果 |
|----|-----------------------------------------------|------|
| 场景 | MV_OFFSET_OF(ARRAY['1','2','3','4','5'], '2') | 1    |

mv\_ordinal\_of 函数

返回数组中第一次出现expr的基于1的索引，如果未出现，则返回-1。

语法：mv\_ordinal\_of(arr, expr)

表 5-285 参数说明

| 参数名称 | 描述   | 类型                          | 是否必选 |
|------|------|-----------------------------|------|
| arr  | 原始数组 | Array ( String/ Number )    | 是    |
| expr | 指定元素 | String/Number，必须与数组中元素类型相同。 | 是    |

返回值类型：Integer类型

示例：SELECT MV\_ORDINAL\_OF(ARRAY['1','2','3','4','5'], '2')

表 5-286 查询分析结果

| 类型 | 查询语句                                           | 返回结果 |
|----|------------------------------------------------|------|
| 场景 | MV_ORDINAL_OF(ARRAY['1','2','3','4','5'], '2') | 2    |

# 6 日志搜索与分析（即将下线 SQL 分析方式）

## 6.1 日志搜索与分析概述

日志搜索与分析是运维中不可或缺的一环。日志接入成功后，云日志服务LTS支持对采集成功的日志数据进行搜索与分析。通过合理的日志收集、高效的搜索方法和专业的分析工具，可以实现对系统或应用的全面监控和精细化管理。

了解更多日志搜索与分析的概念信息，请参考[基本概念](#)。

### 说明

LTS旧版引擎SQL分析方式即将下线，建议您使用新版SQL引擎（管道符搜索方式）。更多管道符信息请参考[日志搜索与分析（管道符方式）](#)。

### 费用说明

搜索分析本身不产生费用，但由于其依赖日志上报至LTS并开启索引，会产生日志读写流量费用、索引流量费用、存储费用等，更多信息请参考[计费项](#)。

### 约束与限制

日志搜索与分析的限制请参考[搜索与分析限制](#)。

### 日志结构化

执行搜索与分析前，需要将上报的日志进行结构化配置和索引配置。结构化后的数据具有统一的长度和格式，能够显著提升搜索与分析的效率和准确性。

日志数据可分为结构化数据和非结构化数据。

- 结构化数据是指以表格、关系型数据库等数据模型组织的数据，具有严格的长度和格式，便于存储和分析。
- 非结构化数据是指不便于用数据库二维逻辑表来表现的数据，数据结构不规则或不完整，缺乏预定义的数据模型，难以直接进行分析。

日志结构化是以日志流为单位，通过特定的提取方式，将日志流中格式固定或相似度较高的日志提取出来，同时过滤掉不相关的日志。结构化后的日志可以方便的使用SQL语法进行查询与分析。

日志结构化解析是将非结构化或半结构化的日志数据转换为结构化格式的过程，以便于更好地存储、查询和分析，提高日志数据的可读性、可搜索性和查询效率。

云日志服务支持两种日志结构化解析方式：ICAgent结构化解析和云端结构化解析，且一个日志流只能配置一种结构化方式，例如选择ICAgent结构化解析后，不能再选择云端结构化解析，需要删除后，才能重新选择。更多信息请参考图6-1。

若用户在日志接入时已经配置ICAgent结构化解析，则无需再配置云端结构化解析。

- ICAgent结构化解析是在采集侧做结构化，支持插件组合解析，单个日志流的多个采集配置支持不同结构化解析规则，推荐使用ICAgent结构化解析的方式，更多内容请参考[配置ICAgent结构化解析](#)。
- 云端结构化解析是通过不同的日志提取方式将日志流中的日志进行结构化，云端结构化解析会消耗LTS系统算力，未来会按照日志量大小收取日志加工流量费用。

图 6-1 不同解析方式



## 日志搜索

结构化完成后，使用云日志服务LTS提供的[搜索语法](#)用于设置搜索条件，帮助您更有效地搜索日志。详细请参考[搜索日志](#)。

## 日志分析

云日志服务LTS支持使用[SQL分析语法](#)，对结构化后的日志字段进行日志分析，通过统计图表的方式对查询和分析的结果进行可视化展示。详细请参考[分析LTS日志](#)。

## 6.2 设置云端结构化解析日志

## 6.2.1 设置日志云端结构化解析

云日志服务LTS目前支持五种日志结构化方式，分别是正则分析、JSON、分隔符、Nginx和结构化模板。您可以根据日志内容的实际场景进行选择。

- **正则分析**：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。使用正则表达式提取字段，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。
- **JSON**：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。
- **分隔符**：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（例如：冒号、空格或字符等）提取为多个key-value键值的日志解析模式。
- **Nginx**：适用于日志文本中每行内容为一条原始日志，每条日志符合Nginx格式，支持通过log\_format指令来自定义访问日志的格式。
- **结构化模板**：适用于日志结构比较复杂或需要自定义提取key-value键值的场景，可以通过内置系统模板或者自定义模板提取字段。

结构化后的日志数据可理解为数据库中的二维表，结构化配置完成后就可以使用SQL语句对提取的字段进行查询与分析。

### 使用限制

- 若未配置索引配置，结构化字段默认分词符为空，最大长度限制为20KB，超过部分会被截断。
- 若已配置索引配置，结构化字段默认分词符参考[设置LTS日志内容分词](#)，此时最大长度限制为500KB。

### 注意事项

- 日志结构化是以日志流为单位。
- 日志流中的大部分日志需有一定的规则，否则结构化是无意义的。
- 结构化配置修改后，对新写入的日志数据生效，历史日志数据不会生效。

### 云端结构化解析

**步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。

**步骤2** 单击目标日志组或日志流名称，进入日志流详情页面。

**步骤3** 在“日志搜索”页面，单击右上角的日志设置，在弹出页面中，选择“云端结构化解析”，进行日志结构化配置。

- 开启“保留原始日志”后，原始日志将作为content字段的值存储到云日志服务。同时在资源统计和计费时content字段均会统计在内。
- 开启“上传解析失败日志”后，原始日志将作为\_content\_parse\_fail\_字段的值上传到云日志服务。
- 关于开启或关闭“保留原始日志”和“上传解析失败日志”时，日志上报参考如下。

图 6-2 云端结构化解析



表 6-1 日志上报说明

| 参数                                                                                  | 日志上报说明                                                                                                                                                    |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>- 开启“保留原始日志”</li><li>- 开启“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>- 解析成功：上报解析后日志和原始日志 content 字段。</li><li>- 解析失败：为了避免日志冗余，只上报原始日志 content 字段，不重复上报 _content_parse_fail_ 字段。</li></ul> |
| <ul style="list-style-type: none"><li>- 开启“保留原始日志”</li><li>- 关闭“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>- 解析成功：上报解析后日志和原始日志 content 字段。</li><li>- 解析失败：上报原始日志 content 字段。</li></ul>                                         |
| <ul style="list-style-type: none"><li>- 关闭“保留原始日志”</li><li>- 开启“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>- 解析成功：上报解析后日志。</li><li>- 解析失败：上报 _content_parse_fail_ 字段。</li></ul>                                                |
| <ul style="list-style-type: none"><li>- 关闭“保留原始日志”</li><li>- 关闭“上传解析失败日志”</li></ul> | <ul style="list-style-type: none"><li>- 解析成功：上报解析后日志。</li><li>- 解析失败：仅上报内置字段和 label 字段。</li></ul>                                                         |

- 结构化不支持的系统字段包括：groupName、logStream、lineNum、content、logContent、logContentSize、collectTime、category、clusterId、clusterName、containerName、hostIP、hostId、hostName、nameSpace、pathFile、podName。
- **正则分析**：使用正则表达式提取字段。
- **JSON**：通过提取JSON字段将其拆分为键值对。
- **分隔符**：使用分隔符（例如：冒号、空格或字符等）提取字段。
- **Nginx**：通过log\_format指令来自定义访问日志的格式。
- **结构化模板**：通过自定义模板或系统内置模板提取字段。

**步骤4** 云端结构化解析配置完成后，支持修改或删除结构化配置。

- 在云端结构化解析页面中，单击 ，修改结构化配置。
- 在云端结构化解析页面中，单击 ，删除结构化配置。

结构化配置删除后，将无法恢复，请谨慎操作。

----结束

正则分析

正则分析是使用正则表达式提取字段。

**步骤1** 选择示例日志：应选择一条比较典型的日志作为示例日志。

- **从已有日志中选择**：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。  
时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。
  - 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
  - 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
  - 自定义：表示查询指定时间范围的日志数据。
- **从剪切板中粘贴**：单击“从剪切板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。

**步骤2** 字段提取。包括自动生成和手动输入两种方式，可将选择的日志提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

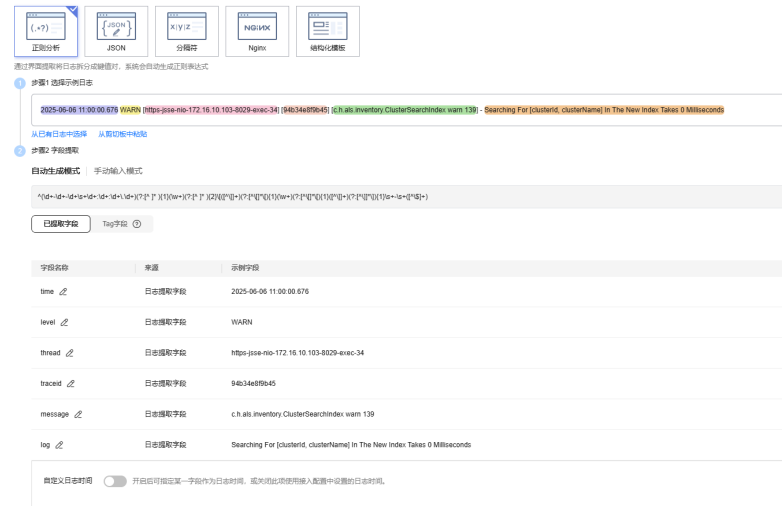
- **自动生成模式**：当用户选择自动生成时，可以用鼠标选中示例日志中待结构化的日志内容，在弹出的对话框中为选中内容设置一个名称，单击“添加”。字段名称只支持输入英文、数字、中划线、下划线及小数点，不能以小数点开头或结尾且不能以双下划线结尾。

例如在选择日志示例下方输入以下原始日志示例：

```
2025-06-06 11:00:00.676 WARN [https-jsse-nio-172.16.10.103-8029-exec-34] [94b34e8f9b45] [c.h.als.inventory.ClusterSearchIndex warn 139] - Searching For [clusterId, clusterName] In The New Index Takes 0 Milliseconds
```

根据原始日志示例，设置字段名称为time、level、thread、message等，自动生成字段参考如图6-3所示。

图 6-3 正则分析自动生成字段



- 手动输入模式：当用户选择手动输入时，可以在输入框中输入正则表达式，单击“生成字段”来进行字段提取。正则表达式通过分组来捕获字段，分组指用圆括号"()"括起来的正则表达式，匹配出的内容就表示一个分组，分组包含如下三种形式：
  - (exp)：把括号内的正则作为一个分组，系统自动分配组号，规则为从正则表达式的左边开始，第一个左括号“(”对应第一个分组，第二个“(”对应第二个分组，依次类推，组号从1开始，从左向右，依次累加。
  - (?<name>exp)：表示命名分组，分组的正则表达式为exp，分组名为name。分组名必须以字母开始，且仅包含字母和数字，可以通过分组名或分组号引用该分组。
  - (?exp)：表示不捕获分组，该分组只在当前位置匹配文本，在该分组之后，无法引用该分组，因为该分组没有分组名，没有分组号，也不会占用分组编号。
  - 分词符指将日志内容切分为多个单词的符号，默认分词符包括, ";=()[]{}@&<>/:\\?\\n\\t\\r，在日志搜索或者对日志进行结构化时，可以选取相邻两分词符之间的单词。
  - 在手工输入方式中，正则表达式的长度不能超过5000个字符，不强制要求用户在输入正则表达式时对分组进行命名，单击“生成字段”会以命名分组中的分组名作为字段名称，对于非命名分组会提取出对应的字段，并给字段名称默认命名field1、field2、field3……。
  - 由于手动输入正则表达式时不能填写双下划线，所以在日志提取字段下方可以将字段名称重命名为带有双下划线的字段。

**步骤3** 若需要指定某一字段作为日志时间，详细请参考[自定义日志时间](#)。

**步骤4** 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

## JSON

JSON是通过提取JSON字段将其拆分为键值对。

**步骤1** 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

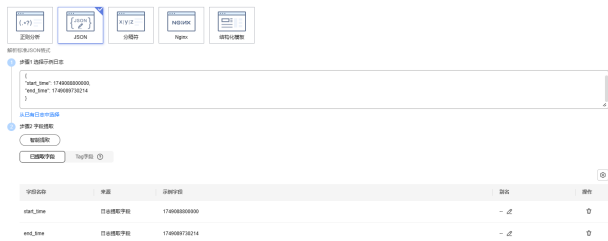
**步骤2** 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

将以下参考示例原始日志输入待操作框中，在“步骤2 字段提取”下单击“智能提取”。自动生成字段参考如[图6-4](#)所示。

```
{
 "start_time": 1749088800000,
 "end_time": 1749089730214
}
```



图 6-4 JSON 字段提取结果



在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考[设置云端结构化字段](#)。

- 当日志提取字段的类型为float时，精度为16位有效数字。如果超过16位有效数字，则会导致提取字段内容不准确，从而影响快速分析，因此建议将字段类型修改为String。
- 当日志提取字段的类型为long时，日志内容超过16位有效数字，只会精确显示前16位有效数字，后面的数字会变为0。
- 当日志提取字段的类型为long时，日志内容超过21位有效数字，则会识别为float类型，建议将字段类型修改为String。

**步骤3** 若需要指定某一字段作为日志时间，详细请参考[自定义日志时间](#)。

**步骤4** 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

分隔符

分隔符是使用分隔符（例如：逗号、空格或字符）提取字段。

**步骤1** 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

**步骤2** 在“步骤2 指定分隔符”需要根据原始日志内容选择分隔符，或自定义其他需要的特殊字符作为分隔符。

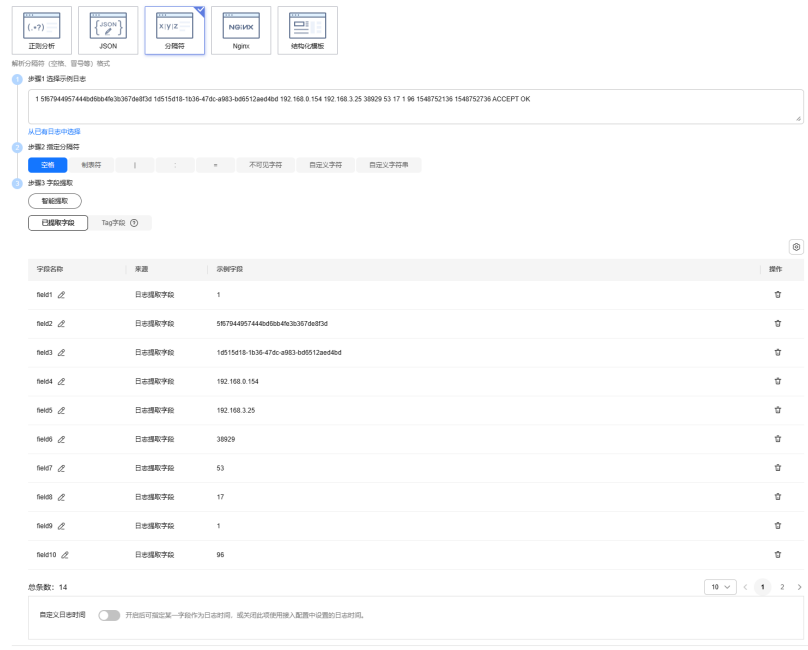
- 不可见字符需要输入长度为4个且以0x开头的16进制字符，例如0x01，最多可输入32个不可见字符。
- 自定义字符支持输入1-10个字符，每个字符都作为独立的分隔符。
- 自定义字符串支持输入1-30个字符，字符串整体作为一个分隔符。

**步骤3** 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

将以下原始日志示例输入待操作框中，在“步骤3字段提取”下单击“智能提取”。自动生成字段参考如图6-5所示。

```
1 5f67944957444bd6bb4fe3b367de8f3d 1d515d18-1b36-47dc-a983-bd6512aed4bd 192.168.0.154
192.168.3.25 38929 53 17 1 96 1548752136 1548752736 ACCEPT OK
```

图 6-5 分隔符字段提取结果



在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考[设置结构化字段](#)。

当日志提取字段的类型为float时，精确度为7位有效数字。

如果超过7位有效数字的话，则会导致提取字段内容不准确，从而影响快速分析，因此建议将字段类型修改为String。

**步骤4** 若需要指定某一字段作为日志时间，详细请参考[自定义日志时间](#)。

**步骤5** 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

## Nginx

Nginx是通过log\_format指令来自定义访问日志的格式。

**步骤1** 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

**步骤2** 在“步骤2 输入Nginx日志配置”中需要输入Nginx日志配置，根据输入或选择的日志进行配置。其中有默认配置可使用，单击“默认Nginx配置”即可。

标准Nginx配置文件中，日志配置的部分通常以log\_format开头。

- 默认配置如下所示，参考表6-2查看字段说明。除了默认字段，请参考表6-3查看常用扩展字段说明。
- ```
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                '$status $body_bytes_sent "$http_referer" '
                '"$http_user_agent" "$http_x_forwarded_for";'
```

表 6-2 默认配置字段参数

字段名称	说明
remote_addr	客户端IP地址。
remote_user	客户端用户名。
time_local	服务器时间，前后必须加上中括号（[]）。
request	请求的URI和HTTP协议。
status	请求状态。
body_bytes_sent	发送给客户端的字节数，不包括响应头的大小。
http_referer	URL跳转来源。
http_user_agent	客户端浏览器等信息。
http_x_forwarded_for	通过代理服务器转发的客户端真实IP地址。

表 6-3 扩展字段参数

字段名称	说明
request_time	整个请求的总时间（单位：秒，精度毫秒）。
upstream_response_time	后端服务器响应时间。
http_cookie	客户端发送的Cookie数据。
http_x_real_ip	客户端的真实IP地址。

字段名称	说明
uri	请求的URI路径（如： /index.html ）。
args	请求行中的参数（如： ? param1=value1&param2=value2 ）。
content_length	请求头中的Content-Length字段。
content_type	请求头中的Content-Type字段。
host	请求头中的Host字段，或处理请求的服务器名称。
server_addr	服务器地址。
server_name	服务器名称。
server_port	服务器端口。
server_protocol	服务器向客户端发送响应时的协议版本（如： HTTP/1.1 ）。
scheme	请求协议（如： http或https ）。

- 用户也可进行自定义配置，具体配置格式要求如下所示。
 - 使用Nginx配置，不可为空。
 - 以log_format开头，并且包含（'）和字段名称。
 - 长度最大限制为5000。
 - 需要与示例日志内容匹配。
 - log_format字段之间的间隔，除大小字母、数字、下划线及中划线外，可使用其他任意字符。
 - 以（'）或者（';）结尾。

步骤3 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

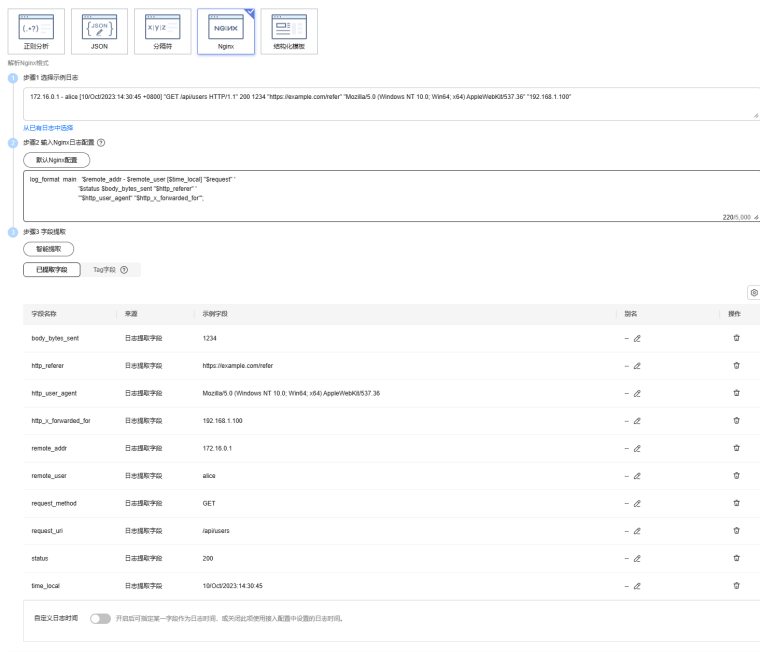
将以下原始日志示例输入待操作框中，在“步骤3 字段提取”下单击“智能提取”。自动生成字段参考如图6-6所示。

```
172.16.0.1 - alice [10/Oct/2023:14:30:45 +0800] "GET /api/users HTTP/1.1" 200 1234 "https://example.com/refer" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36" "192.168.1.100"
```

并使用如下Nginx日志配置。

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

图 6-6 Nginx 字段提取结果



在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考[设置结构化字段](#)。

- 当日志提取字段的类型为float时，精确度为7位有效数字。
- 如果超过7位有效数字的话，则会导致提取字段内容不准确，从而影响快速分析，因此建议将字段类型修改为String。

步骤4 若需要指定某一字段作为日志时间，详细请参考[自定义日志时间](#)。

步骤5 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

结构化模板

结构化模板是通过自定义模板或系统内置模板提取字段。

详情请参考[结构化模板](#)。

相关文档

LTS支持通过调用API接口实现创建结构化配置、查询结构化配置、查询结构化模板、删除结构化配置等，详细请参考[云端结构化配置](#)。

6.2.2 设置云端结构化字段和 tag 字段

设置云端结构化字段

在进行结构化配置字段提取之后，可对结构化字段进行设置，详细请参考[表6-4](#)。

表 6-4 结构化字段设置规则

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除
正则分析（自动生成）	用户自定义。 名称必须以字母开始，且仅包含字母和数字。	是	是
正则分析（手动输入）	- 支持在输入正则表达式时进行命名。 - 支持使用系统默认命名field1、field2、field3等。	是	是
JSON格式	智能提取字段名称，可定义别名。	是	是
分隔符	默认名称field1、field2、field3……，可进行修改。	是	是
Nginx	根据Nginx配置生成，可定义别名。	是	是
自定义模板	用户自定义。	是	是

正则分析（手动输入）、JSON格式、分隔符、Nginx和自定义模板的字段名称需要满足如下要求：

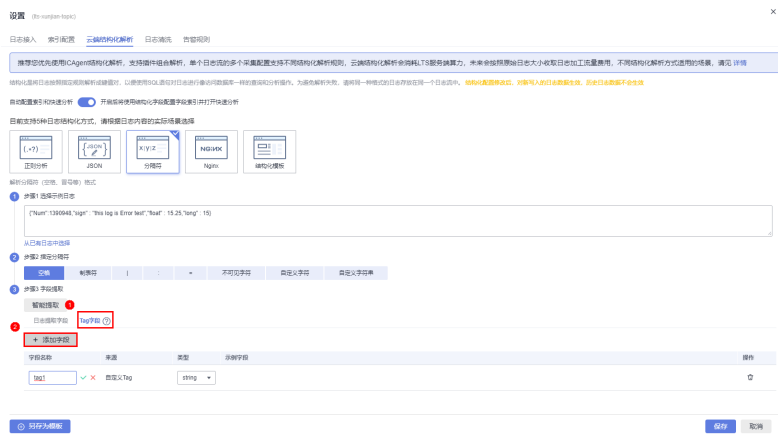
只支持输入英文、数字、中划线、下划线及小数点，不能包含双下划线、下划线加点且不能以小数点开头或结尾。长度为1-64个字符。

设置 tag 字段

设置结构化配置时，可以对日志维度信息进行tag字段设置。设置完成后可以在日志分析页面对设置字段进行SQL查询。

步骤1 在字段提取步骤中，选择“tag字段”页签，单击“添加字段”。

图 6-7 添加 tag 字段



- 步骤2 在tag字段列表中，输入需要设置tag“字段名称”。
- tag字段功能上线前，已设置的结构化配置，在修改结构化配置进行tag字段设置时，系统tag不会带出示例字段。
- 步骤3 如需添加多个字段可单击“添加字段”，继续添加。
- 步骤4 设置完成后，单击“保存”。
- 结束

6.2.3 设置云端结构化自定义日志时间

您可以开启“自定义日志时间”，使用自定义日志内容的时间作为日志搜索页面日志内容的时间。关闭“自定义日志时间”后，则使用日志采集时间作为日志搜索页面日志内容的时间。

开启自定义日志时间

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 单击目标日志组或日志流名称，进入日志流详情页面。
- 步骤3 在“日志搜索”页面，单击右上角的日志设置，在弹出页面中，选择“云端结构化解析”，详细请参考[设置日志云端结构化解析](#)。
- 步骤4 配置完成后，开启自定义日志时间开关，配置如下参数。
- 切换自定义日志时间开关时，可能会导致日志搜索界面在切换时间点附近出现时间偏差，请勿频繁切换自定义日志时间开关。

表 6-5 参数配置表

参数	说明	示例
字段key	已提取字段的名称。单击下拉框选择已提取的字段，该字段为string或long类型。	test
字段value	已提取的字段value，选择字段key后，将自动填充。	2022-07-19 12:12:00
时间格式	请参考 常见日志时间格式 。	yyyy-MM-dd HH:mm:ss
操作	单击“校验”，提示“时间格式和字段value匹配成功”则表示校验成功。	-

----结束

常见日志时间格式

常见日志时间格式如下[表6-6](#)。

默认情况下，云日志服务LTS中的日志时间戳精确到秒，因此时间格式只需配置到秒，无需配置毫秒、微秒等信息。

表 6-6 时间格式

时间格式	说明	示例
EEE	星期的缩写。	Fri
EEEE	星期的全称。	Friday
MMM	月份的缩写。	Jan
MMMM	月份的全称。	January
dd	每月第几天，十进制，范围为01~31。	07, 31
HH	小时，24小时制。	22
hh	小时，12小时制。	11
MM	月份，十进制，范围为01~12。	08
mm	分钟，十进制，范围为00~59。	59
a	AM或PM。	AM、PM
hh:mm:ss a	12小时制的时间组合。	11:59:59 AM
HH:mm	小时和分钟组合。	23:59
ss	秒数，十进制，范围为00~59。	59
yy	年份，十进制，不带世纪，范围为00~99。	04、98
yyyy	年份，十进制。	2004、1998
d	每月第几天，十进制，范围为1~31。	7、31
DDD	一年中的天数，十进制，范围为001~366。	365
u	星期几，十进制，范围为1~7，1表示周一。	2
w	每年的第几周，星期天是一周的开始，范围为00~53。	23
W	每月的第几周，范围为0~5。	2
U	星期几，十进制，范围为0~6，0代表周日。	5
EEE MMM dd HH:mm:ss yyyy	标准的日期和时间。	Tue Nov 20 14:12:58 2020
EEE MMM dd yyyy	标准的日期，不带时间。	Tue Nov 20 2020
HH:mm:ss	标准的时间，不带日期。	11:59:59
%s	Unix时间戳。	147618725

示例

常见的时间标准、示例及对应的时间表达式如下表6-7。

表 6-7 示例

示例	时间表达式	时间标准
2022-07-14T19:57:36+08:00	yyyy-MM-dd'T'HH:mm:ssXXX	自定义
1548752136	%s	自定义
27/Jan/2022:15:56:44	dd/MMM/yyyy:HH:mm:ss	自定义
2022-08-15 17:53:23+08	yyyy-MM-dd HH:mm:ssX	自定义
2022-08-05T08:24:15.536+0000	yyyy-MM-dd'T'HH:mm:ss.SSSZ	自定义
2022-08-20T10:04:03.204000Z	yyyy-MM-dd'T'HH:mm:ss.SSSZ	自定义
2022-08-22T06:52:08Z	yyyy-MM-dd'T'HH:mm:ssZ	自定义
2022-07-24T10:06:41.000	yyyy-MM-dd'T'HH:mm:ss.SSS	自定义
Monday, 02-Jan-06 15:04:05 MST	EEEE, dd-MMM-yy HH:mm:ss Z	RFC850
Mon, 02 Jan 2006 15:04:05 MST	EEE, dd MMM yyyy HH:mm:ss Z	RFC1123
02 Jan 06 15:04 MST	dd MMM yy HH:mm Z	RFC822
02 Jan 06 15:04 -0700	dd MMM yy HH:mm Z	RFC822Z
2023-01-02T15:04:05Z07:00	yyyy-MM-dd'T'HH:mm:ss Z	RFC3339
2022-12-11 15:05:07	yyyy-MM-dd HH:mm:ss	自定义

6.2.4 设置云端结构化模板

云日志服务LTS目前支持的结构化模板有两种，分别为系统模板和自定义模板。

系统模板

支持多种系统模板，不支持修改系统模板的字段类型和删除字段，详情请参考表6-8。

- 步骤1 在云端结构化解析页面中，日志结构化方式选择“结构化模板”。
- 步骤2 在“选择模板”下，选择“系统模板”，选择对应的系统模板，模板日志从对应的云服务接入，可以直接应用模板的数据模型作为示例日志。

步骤3 选择模板后，“模板详情”中会自动显示对应的日志解析结果。

- 结构化配置时，如果使用系统模板，则系统模板中的时间为自定义日志时间。支持通过模板名称搜索模板，方便用户快速查询模板信息。
- string类型的字段不支持使用运算符（>=<）或 in 语法进行范围查询，建议使用星号（*）或问号（?）进行模糊查询。需要重新配置结构化，将该字段修改为数字类型。

表 6-8 系统模板

模板名称	模板说明	字段类型是否可修改	字段是否可删除	系统模板的请求参数
ELB模板	根据ELB资料中提供的日志字段被定义。	否	否	ELB
VPC模板	根据VPC资料中提供的日志字段被定义。	否	否	VPC
CTS模板	字段名称为json日志中的key。	否	否	CTS
APIG模板	根据APIG资料中提供的日志字段被定义。	否	否	APIG
DCS审计日志	根据DCS资料中提供的日志字段被定义。	否	否	DCS_AUDIT
TOMCAT	根据TOMCAT官网提供的字段名称进行nginx解析的名称。	否	否	TOMCAT
NGINX	根据NGINX资料中提供的日志字段被定义。	否	否	NGINX
GAUSSV5审计日志	根据GAUSSV5资料中提供的日志字段被定义。	否	否	GAUSSDB_OPENGAUSS_AUDIT
DDS审计日志	根据DDS资料中提供的日志字段被定义。	否	否	DDS_AUDIT
DDS错误日志	根据DDS资料中提供的日志字段被定义。	否	否	MONGODB_ERROR
DDS慢日志	根据DDS资料中提供的日志字段被定义。	否	否	MONGODB_SLOW
CFW访问控制日志	根据CFW资料中提供的日志字段被定义。	否	否	CFW_ACCESS
CFW攻击日志	根据CFW资料中提供的日志字段被定义。	否	否	CFW_ATTACK
CFW流量日志	根据CFW资料中提供的日志字段被定义。	否	否	CFW_FLOW

模板名称	模板说明	字段类型是否可修改	字段是否可删除	系统模板的请求参数
MYSQL错误日志	根据MYSQL资料中提供的日志字段被定义。	否	否	MYSQL_ERROR
MYSQL慢日志	根据MYSQL资料中提供的日志字段被定义。	否	否	MYSQL_SLOW
POSTGRESQL慢日志	根据POSTGRESQL慢日志资料中提供的日志字段被定义。	否	否	POSTGRESQL_SLOW
POSTGRESQL错误日志	根据POSTGRESQL错误日志资料中提供的日志字段被定义。	否	否	POSTGRESQL_ERROR
SQLSERVER错误日志	根据SQLSERVER资料中提供的日志字段被定义。	否	否	SQLSERVER_ERROR
GeminiDB Redis慢日志	根据GeminiDB Redis资料中提供的日志字段被定义。	否	否	GAUSSDB_REDIS_SLOW
CDN	根据CDN资料中提供的日志字段被定义。	否	否	CDN
SMN	根据SMN资料中提供的日志字段被定义。	否	否	SMN
GAUSSDB_MYSQL错误日志	根据GAUSSDB_MYSQL资料中提供的日志字段被定义。	否	否	GAUSSDB_MYSQL_ERROR
GAUSSDB_MYSQL慢日志	根据GAUSSDB_MYSQL资料中提供的日志字段被定义。	否	否	GAUSSDB_MYSQL_SLOW
ER企业路由器	根据ER企业路由器资料中提供的日志字段被定义。	否	否	ER
MYSQL审计日志	根据MYSQL审计日志资料中提供的日志字段被定义。	否	否	MYSQL_AUDIT
GeminiDB Cassandra慢日志	根据GeminiDB Cassandra慢日志资料中提供的日志字段被定义。	否	否	GAUSSDB_CASSANDRA_SLOW
GeminiDB Mongo慢日志	根据GeminiDB Mongo慢日志资料中提供的日志字段被定义。	否	否	GAUSSDB_MONGO_SLOW
GeminiDB Mongo错误日志	根据GeminiDB Mongo错误日志资料中提供的日志字段被定义。	否	否	GAUSSDB_MONGO_ERROR
WAF访问日志	根据WAF访问日志资料中提供的日志字段被定义。	否	否	WAF_ACCESS

模板名称	模板说明	字段类型是否可修改	字段是否可删除	系统模板的请求参数
WAF攻击日志	根据WAF攻击日志资料中提供的日志字段被定义。	否	否	WAF_ATTACK
DMS重平衡日志	根据DMS重平衡日志资料中提供的日志字段被定义。	否	否	DMS_REBALANCED
CCE审计日志	根据CCE审计日志资料中提供的日志字段被定义。	否	否	CCE_AUDIT
CCE事件日志	根据CCE事件日志资料中提供的日志字段被定义。	否	否	CCE_EVENT
CCE NGINX-INGRESS日志	根据CCE NGINX-INGRESS日志资料中提供的日志字段被定义。	否	否	NGINX_INGRESS
GeminiDB Redis审计日志	根据GeminiDB Redis审计日志资料中提供的日志字段被定义。	否	否	GAUSSDB_REDIS_AUDIT
influx慢日志	根据influx慢日志资料中提供的日志字段被定义。	否	否	GAUSSDB_INFLUX_SLOW
METRIC系统日志	根据日志生成指标任务过滤的监控日志字段被定义。	否	否	SYS_LOG_FOR_LOG2METRIC
Microgateway	根据Microgateway应用网关提供的日志字段被定义。	否	否	MICROGATEWAY
GeminiDB Mongo接口审计日志	根据GeminiDB Mongo接口审计日志提供的日志字段被定义。	否	否	GEMINIDB_MONGODB_AUDIT
CloudTable Doris审计日志	根据CloudTable Doris审计日志提供的日志字段被定义。	否	否	CLOUD_TABLE_DORIS
CoreDNS日志	根据CoreDNS日志提供的日志字段被定义。	否	否	COREDNS_LOG
TaurusDB审计日志	根据TaurusDB审计日志提供的日志字段被定义。	否	否	TAURUS_DB_AUDIT
Workspace用户操作日志	根据Workspace用户操作日志提供的日志字段被定义。	否	否	WKS_USER_EVENT
DNS解析请求日志	根据DNS解析请求日志提供的日志字段被定义。	否	否	DNS_QUERY_LOG

步骤4 单击“保存”完成结构化配置。

----结束

自定义模板

在“选择模板”下，选择“自定义模板”，选择已有的结构化模板。自定义模板主要来源有以下两种：

- 在配置正则分析、JSON、分隔符或Nginx方式时，单击左下角的“另存为模板”，系统会弹出“另存为模板”页面，输入模板名称，单击“确定”，保存自定义模板。该模板会在“自定义模板”下的模板列表展示。
- 新增结构化模板，具体操作如下：
 - 在“选择模板”下，选择“自定义模板”，单击“新增结构化模板”。
 - 在“新增结构化模板”页面中，选择正则分析、JSON、分隔符或Nginx方式，进行配置。
 - 配置完成后输入模板名称，单击“确定”，完成自定义模板的保存，会在“自定义模板”下的模板列表展示。

6.3 创建 LTS 日志索引

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

索引类型

云日志服务LTS支持全文索引和字段索引，详细请参考[表6-9](#)。

表 6-9 索引类型

索引类型	说明
全文索引	开启全文索引后，日志服务根据您设置的分词符将整条日志所有字段值拆分成多个词并构建索引。 - 用户上传的自定义标签（label）字段，不包含在全文索引中，如果您需要搜索自定义标签字段，请添加对应的字段索引。 - LTS内置保留字段，不包含在全文索引中，您需要通过字段索引Key:Value的方式进行搜索，请参考内置保留字段。

索引类型	说明
字段索引	配置字段索引后，您可以指定字段名称和字段值（Key:Value）进行查询，缩小查询范围。 ● 日志服务默认为部分内置保留字段创建字段索引，请参考内置保留字段。 ● 如果您的某个字段单独配置了字段索引，那么该字段值的分词符以字段索引配置为准。 ● 结构化配置中的快速分析列已被移除，如果您要使用快速分析功能，则必须配置字段索引且开启对应字段的快速分析按钮。 ● 例如以下日志示例，如果配置了level和status两个字段索引，其中level是string类型，字段值是error，单独配置了分词符，status是long类型，不需要配置分词符；您可以使用level:error的方式精确搜索level字段值为error的所有日志。 ● 例如以下日志示例，云日志服务LTS会默认为hostName、hostIP、pathFile这些内置保留字段创建字段索引。

以下是一条典型日志示例，content字段值是日志原文，使用分隔符逗号将原始日志解析成3个字段level、status、message。

示例日志中的hostName、hostIP、pathFile是常见的内置保留字段，详细内置字段请参考[内置保留字段](#)。

```
{
  "hostName": "epstest-xx518",
  "hostIP": "192.168.0.31",
  "pathFile": "stdout.log",
  "content": "error,400,I Know XX",
  "level": "error",
  "status": 400,
  "message": "I Know XX"
}
```

费用说明

创建索引会产生索引流量，更多信息请参考[计费项](#)。

注意事项

- 全文索引属性和字段索引属性必须至少启用一种。
- 关闭索引后，历史索引的存储空间将在当前日志流的数据保存时间到期后，自动被清除。
- 云日志服务默认已为部分内置保留字段创建字段索引，请参见[内置保留字段](#)。
- 不同的索引配置，会产生不同的查询和分析结果，请根据您的需求，合理创建索引。全文索引和字段索引互不影响。
- 索引配置修改后，对新写入的日志数据生效，历史日志数据不会生效。
- 在字段索引功能上线前，SQL分析支持的字段来自于云端结构化解析；在字段索引功能上线后，只要用户配置了字段索引，SQL分析支持的字段将来自于字段索引，因此修改字段索引可能对现有的可视化图表、仪表盘、SQL告警、定时SQL、Grafana接入中的查询结果产生影响，请谨慎操作！

配置全文索引

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 单击目标日志组或日志流名称，进入日志流详情页面。
- 步骤3 在“日志搜索”页面，单击快速分析旁边的进入“索引配置”页面。

图 6-8 配置索引



- 步骤4 在索引配置页面中，默认开启“全文索引”按钮，参考[表6-10](#)配置各参数信息。

表 6-10 自定义全文索引配置参数

参数	说明	示例
全文索引	开启“全文索引”开关，表示创建全文索引。	开启
大小写敏感	查询时是否区分英文字母的大小写。 - 开启“大小写敏感”开关，则查询时区分大小写。例如示例日志含有Know，那么您只能使用Know才能查询到该日志。 - 关闭“大小写敏感”开关，则查询时不区分大小写。例如示例日志含有Know，那么您使用关键字KNOW和know都能查到该日志。	开启

参数	说明	示例
包含中文	查询时是否区分中英文。 - 开启“包含中文”开关后，如果日志中包含中文，默认按照一元分词法拆分中文内容，按照分词符的设置拆分英文内容。 - 一元分词是指将中文字符串拆分为单个独立的中文字。 - 使用一元分词符的优点是对海量日志分词效率高，其他中文分词方法对写入速度影响大。 - 打开包含中文功能，会对中文使用一元分词（每个汉字单独分词），如果需要更精确的搜索结果，请用短语搜索，语法为：#"待搜索的短语"。 - 关闭“包含中文”开关后，按照分词符的设置拆分所有内容。	例如示例日志内容为： error,400,I Know 今天是星期一。 - 关闭“包含中文”开关后，按照分词符的设置拆分英文内容，日志会被拆分为error、400、I、Know、今天是星期一，您可以通过error或今天是星期一查找该日志。 - 开启“包含中文”开关后，日志服务后台分词器将日志拆分为error、400、I、Know、今、天、是、星、期、一，您通过error或今天等词都可以查找到该日志。
分词符	根据指定分词符，将日志内容拆分成多个词。当默认设置不能满足您的需求时，您可以自定义设置分词符。 如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整字符串或模糊查询查找对应的日志。 单击“预览”，查看分词预览效果。	例如示例日志内容为： error,400,I Know 今天是星期一。 - 如果不设置任何分词符，整条日志被作为一个词error,400,I Know 今天是星期一，您只能通过完整字符串error,400,I Know 今天是星期一或模糊查询error,400,I K*查找该日志。 - 如果设置分词符为逗号（,），则原始日志被拆分为error、400、I Know 今天是星期一3个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如error、400、I Kn*、今天是*。 - 如果设置分词符为逗号（,）和空格，则原始日志被拆分为error、400、I、Know、今天是星期一5个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如Know、今天是*。

参数	说明	示例
特殊分词符	单击“添加特殊分词符”，参考 ASCII码对照表 输入ASCII值和控制字符。	-

- 步骤5** 全文索引配置完成后，单击“确定”。
- 如果没有开通管道符方式，需要使用SQL日志分析功能，请开启“日志分析”。

图 6-9 日志分析



- “日志分析”功能即将下线，建议直接使用管道符方式进行分析，详细请参考[日志搜索与分析（管道符方式）](#)。
- 开通管道符方式后，新建的日志流在索引配置页面不显示“日志分析”开关。

----结束

配置字段索引

创建字段索引时，最多支持添加500个字段。其中JSON类型字段，最多支持添加100个子字段。

- 步骤1** 设置快速分析采样条数，默认值10万条，最小值为10万条，最大值1000万条。通过采样快速统计字段值取值分布，并非对全量数据进行分析，采样条数越多分析数据越慢。
- 步骤2** 自动配置字段索引。在索引配置页面的字段索引下方，单击“自动配置”，云日志服务LTS会根据近15分钟的第一条日志内容或常见内置保留字段（例如hostIP、hostName、pathFile）自动生成字段索引，您可以根据自己的需要增加或者删除字段。
- 在索引配置页面单击“自动配置”时，默认获取最近15分钟的原始日志和内置字段的交集，LTS自动将原始日志和内置字段的交集、当前结构化字段、tag字段一起组成字段索引下方的表格数据。
 - 若15分钟内没有原始日志，则获取hostIP、hostName、pathFile、结构化字段、tag字段结合共同组成字段索引下方的表格数据。
 - ECS接入选择结构化配置时，进入索引配置页面，若没有单击“自动配置”，则会自动加上如下字段：category、hostName、hostId、hostIP、hostIPv6、pathFile，添加字段时，若某个字段已存在于索引配置，则不会重复添加。
 - CCF接入选择结构化配置时，进入索引配置页面，若没有单击“自动配置”，则会自动加上如下字段：category、clusterId、clusterName、nameSpace、podName、containerName、appName、hostName、hostId、hostIP、hostIPv6、pathFile，添加字段时，若某个字段已存在于索引配置，则不会重复添加。

步骤3 单个添加字段索引。在索引配置页面的字段索引下方，单击“添加字段”，配置字段索引。具体的参数配置请参考[表6-11](#)。

- 通过添加日志结构化中提取的字段，对日志数据进行统计与分析，如果您在结构化界面对字段名称进行了修改，请重新添加字段。
- 字段索引的参数配置仅对该字段生效。
- 当添加的字段在日志内容中不存在时，则配置的该索引字段无效。

表 6-11 自定义字段索引配置参数

参数	说明
字段名称	日志字段名称，例如示例日志中的level。 - 字段名称只支持输入英文、数字、中划线、下划线及小数点，不能以小数点开头或结尾且不能以双下划线结尾。 - 日志服务默认会对部分内置保留字段开启字段索引，请参见内置保留字段。 - 若是内置字段，在字段名称后会显示“内置”字眼，方便用户识别。
执行操作	显示字段的添加状态：新增、不修改、修改、删除。索引字段有变动后，单击“修改对比”，即可查看原配置内容与修改后配置内容的差异。 - 显示新增的字段不支持修改执行操作。 - 修改类型、别名、大小写敏感、自定义分词符、特殊分词符、包含中文、快速分析时，会与原索引配置中的字段进行对比，若任意一项不同，则执行操作变为“修改”。 - 索引配置单击确定后，不会保存执行操作为“删除”的字段。 - 在字段索引下方的搜索框，支持按字段名称、执行操作、类型搜索字段。 - 勾选多个字段后，单击“批量配置”，支持批量配置如下操作：执行操作、类型、大小写敏感、自定义分词符等。
类型	- 日志字段值（Value）的数据类型，可选值为string、long、float、json。 - long类型和float类型不支持设置大小写敏感、包含中文、分词符。
别名	支持为字段名称设置别名，合理设置字段别名可以显著提高日志的可读性、可维护性和查询效率。 设置别名后，只支持使用别名进行SQL搜索分析，不支持使用别名进行关键字搜索。

参数	说明
大小写敏感	查询时是否区分英文字母的大小写。 - 开启“大小写敏感”开关，则查询时区分大小写。例如示例日志message字段中含有Know，那么您只能使用message:Know才能查询到该日志。 - 关闭“大小写敏感”开关，则查询时不区分大小写。例如示例日志message字段中含有Know，那么您使用关键字message:KNOW和message:know都能查到该日志。
自定义分词符	根据指定分词符，将日志内容拆分成多个词。当默认设置不能满足您的需求时，您可以自定义设置分词符。 如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整字符串或模糊查询查找对应的日志。 例如示例日志message字段内容为：I Know 今天是星期一。 - 如果不设置任何分词符，整条日志被作为一个词I Know 今天是星期一，您只能通过完整字符串message:I Know 今天是星期一或模糊查询message:I Know 今天是*查找该日志。 - 如果设置分词符为空格，则原始日志被拆分为I、Know、今天是星期一3个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如message:Know或message:今天是星期一。
特殊分词符	单击“添加特殊分词符”，参考ASCII码对照表输入ASCII值和控制字符。
包含中文	查询时是否区分中英文。 - 开启“包含中文”开关后，如果日志中包含中文，默认按照一元分词法拆分中文内容，按照分词符的设置拆分英文内容。 - 一元分词是指将中文字符串拆分为单个独立的中文字。 - 使用一元分词符的优点是对海量日志分词效率高，其他中文分词方法对写入速度影响大。 - 打开包含中文功能，会对中文使用一元分词（每个汉字单独分词），如果需要更精确的搜索结果，请用短语搜索，语法为：#"待搜索的短语"。 - 关闭“包含中文”开关后，按照分词符的设置拆分所有内容。 例如示例日志message字段内容为：I Know 今天是星期一。 - 关闭“包含中文”开关后，按照分词符的设置拆分英文内容，日志会被拆分为I、Know、今天是星期一，您可以通过message:Know或message:今天是星期一查找该日志。 - 打开“包含中文”开关后，日志服务后台分词器将日志拆分为I、Know、今、天、是、星、期、一，您通过message:Know或message:今天等词都可以查找到该日志。

参数	说明
快速分析	默认为开启状态，开启后，可以对字段值做采样统计，请参见 创建快速分析 。 - 快速分析的原理是对搜索命中的日志采样10万条进行数据统计，不是全量统计。 - 快速分析的字段长度最大为2000字节。 - 快速分析字段展示前100条数据。
操作	单击  ，删除添加的自定义字段。

步骤4 完成后，单击“确定”。

----结束

内置保留字段

在采集日志时，云日志服务会将采集时间、日志类型、主机IP等信息以Key-Value对的形式添加到日志中，这些字段是云日志服务的内置字段。

- 使用API写入日志数据或添加ICAgent配置时，请不要将字段名称设置为内置保留字段，否则可能会造成字段名称重复、查询不精确等问题。
- 云日志服务为日志数据增加的内置保留字段当前免费。更多信息请参见[价格计算器](#)。
- 用户自定义日志字段名称中不能使用双下划线__，否则无法配置索引。

表 6-12 内置保留字段说明

内置保留字段	数据格式	索引与统计设置	说明
collectTime	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为collectTime创建字段索引，索引数据类型为long类型。 查询时输入 collectTime : xxx。	采集时间，指日志被采集器ICAgent采集时的时间。 例如示例中的 "collectTime": "1681896081334"，转换成标准时间是 2023-04-19 17:21:21

内置保留字段	数据格式	索引与统计设置	说明
__time__	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为time创建字段索引，索引数据类型为long类型。该字段不支持查询。	日志时间，指的是日志在控制台页面展示的日志时间。 例如示例中的" <code>__time__</code> ":"1681896081334"，转换成标准时间是2023-04-19 17:21:21 日志时间默认使用采集时间，也支持自定义日志时间。
lineNum	整型	索引设置：开启索引后，日志服务默认为lineNum创建字段索引，索引数据类型为long类型。	行号（偏移量），用来排序日志。 非高精度日志会根据collectTime生成，默认是collectTime * 1000000 + 1，高精度日志就是用户上报的纳秒值。 例如示例中的" <code>lineNum</code> ":"1681896081333991900"。
category	字符串	索引设置：开启索引后，日志服务默认为category创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入category: xxx。	日志类型，表示该日志的来源。 例如ICAgent采集的日志该字段为LTS，某云服务例如DCS上报的日志该字段为DCS。
clusterName	字符串	索引设置：开启索引后，日志服务默认为clusterName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入clusterName: xxx。	集群名称，k8s场景下集群名称。 例如示例中的" <code>clusterName</code> ":"epstest"。
clusterId	字符串	索引设置：开启索引后，日志服务默认为clusterId创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入clusterId: xxx。	集群ID，k8s场景下集群ID。例如示例中的" <code>clusterId</code> ":"c7f3f4a5-xxxx-11ed-a4ec-0255ac100b07"。

内置保留字段	数据格式	索引与统计设置	说明
nameSpace	字符串	索引设置：开启索引后，日志服务默认为nameSpace创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入nameSpace: xxx。	命名空间，k8s场景下命名空间。 例如示例中的"nameSpace":"monitoring"。
appName	字符串	索引设置：开启索引后，日志服务默认为appName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入appName: xxx。	组件名称，k8s场景下工作负载的名称。 例如示例中的"appName":"alertmanager-alertmanager"。
serviceID	字符串	索引设置：开启索引后，日志服务默认为serviceID创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入serviceID: xxx。	工作负载ID，k8s场景下工作负载ID。 例如示例中的"serviceID":"cf5b453xxxad61d4c483b50da3fad5ad"。
podName	字符串	索引设置：开启索引后，日志服务默认为podName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入podName: xxx。	POD名称，k8s场景下POD名称。 例如示例中的"podName":"alertmanager-alertmanager-0"。
podIp	字符串	索引设置：开启索引后，日志服务默认为podIp创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入podIp: xxx。	pod的ip，k8s场景下pod的IP地址。 例如示例中的"podIp":"10.0.0.145"。
containerName	字符串	索引设置：开启索引后，日志服务默认为containerName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入containerName: xxx。	容器名称，k8s场景下容器名称。 例如示例中的"containerName":"config-reloader"。

内置保留字段	数据格式	索引与统计设置	说明
hostName	字符串	索引设置：开启索引后，日志服务默认为hostName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostName: xxx。	主机名称，ICAgent所在主机的名称。 例如示例中的"hostName":"epstest-xx518"。
hostId	字符串	索引设置：开启索引后，日志服务默认为hostId创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostId: xxx。	主机ID，ICAgent所在主机的id，该id由ICAgent生成。例如示例中的"hostId":"318c02fe-xxxx-4c91-b5bb-6923513b6c34"。
hostIP	字符串	索引设置：开启索引后，日志服务默认为hostIP创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostIP: xxx。	主机IP，日志采集器所在主机的ip（适用于ipv4场景） 例如示例中的"hostIP":"192.168.0.31"。
hostIPv6	字符串	索引设置：开启索引后，日志服务默认为hostIPv6创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostIPv6: xxx。	主机IP，日志采集器所在主机的ip（适用于ipv6场景） 例如示例中的"hostIPv6":""。
pathFile	字符串	索引设置：开启索引后，日志服务默认为pathFile创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入pathFile: xxx。	文件路径，采集的日志文件的路径。 例如示例中的"pathFile":"stdout.log"。
content	字符串	索引设置：开启全文索引后，会使用全文索引定义的分词符对content字段的value进行分词；不支持将content字段配置到字段索引中。	日志原文，例如示例中的"content":"level=error ts=2023-04-19T09:21:21.333895559Z"

内置保留字段	数据格式	索引与统计设置	说明
__receive_time__	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为__receive_time__创建字段索引，索引数据类型为long类型。	上报日志的服务端时间，相当于LTS采集端接收到日志的时间。
__client_time__	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为__client_time__创建字段索引，索引数据类型为long类型。	端侧日志的客户端上报时间。
_content_parse_fail_	字符串	索引设置：开启索引后，日志服务默认为_content_parse_fail_创建字段索引，索引数据类型为string类型，分词字符为默认分词符。查询时输入_content_parse_fail_:xxx。	上报日志解析失败的日志内容。
__time	整型，Unix时间戳（毫秒）	不支持将__time字段配置到字段索引中。	不涉及。
logContent	字符串	不支持将logContent字段配置到字段索引中。	不涉及。
logContentSize	整型	不支持将logContentSize字段配置到字段索引中。	不涉及。
logIndexSize	整型	不支持将logIndexSize字段配置到字段索引中。	不涉及。
groupName	字符串	不支持将groupName字段配置到字段索引中。	不涉及。
logStream	字符串	不支持将logStream字段配置到字段索引中。	不涉及。

ASCII 码对照表

表 6-13 ASCII 码对照表

AS CII 值	控制字符	ASC II值	控制字符	AS CII 值	控制字符	AS CII 值	控制字符
0	NUL（空字 符）	32	空格	64	@	96	`
1	SOH（标题开 始）	33	!	65	A	97	a
2	STX（正文开 始）	34	"	66	B	98	b
3	ETX（正文结 束）	35	#	67	C	99	c
4	EOT（传输结 束）	36	\$	68	D	100	d
5	ENQ（询问字 符）	37	%	69	E	101	e
6	ACK（确认回 应）	38	&	70	F	102	f
7	BEL（响铃）	39	'	71	G	103	g
8	BS（退格）	40	(	72	H	104	h
9	HT（水平定位 符号，制表 符）	41	)	73	I	105	i
10	LF（换行）	42	*	74	J	106	j
11	VT（垂直定位 符号）	43	+	75	K	107	k
12	FF（换页键）	44	,	76	L	108	l
13	CR（归位键）	45	-	77	M	109	m
14	SO（取消变 换）	46	.	78	N	110	n
15	SI（启用变 换）	47	/	79	O	111	o
16	DLE（跳出数 据通讯）	48	0	80	P	112	p
17	DC1（设备控 制1）	49	1	81	Q	113	q

AS CII 值	控制字符	ASC II值	控制字符	AS CII 值	控制字符	AS CII 值	控制字符
18	DC2（设备控制2）	50	2	82	R	114	r
19	DC3（设备控制3）	51	3	83	S	115	s
20	DC4（设备控制4）	52	4	84	T	116	t
21	NAK（确认失败回应）	53	5	85	U	117	u
22	SYN（同步用暂停）	54	6	86	V	118	v
23	ETB（区块传输结束）	55	7	87	W	119	w
24	CAN（取消）	56	8	88	X	120	x
25	EM（连接介质中断）	57	9	89	Y	121	y
26	SUB（替换）	58	:	90	Z	122	z
27	ESC（跳出）	59	;	91	[	123	{
28	FS（文件分隔符）	60	<	92	\	124	
29	GS（组群分隔符）	61	=	93	]	125	}
30	RS（记录分隔符）	62	>	94	^	126	~
31	US（单元分隔符）	63	?	95	_	127	DEL（删除）

相关文档

- 创建LTS日志索引后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- LTS支持通过调用API接口创建日志流索引，详细请参考[创建日志流索引](#)。
- LTS支持通过调用API接口查询日志流索引，详细请参考[查询日志流索引](#)。

6.4 搜索日志

当您配置完成日志结构化解析和索引后，就可以通过输入搜索语句，在日志数据中查找包含特定关键词的日志记录。或者根据时间范围来检索日志数据，帮助您定位特定时间段内发生的事件和问题。

搜索语句用于指定日志查询时的过滤规则，返回符合条件的日志。搜索语句可以为关键词、数值、数值范围、空格、星号（*）等。如果为空格或星号（*），表示无过滤条件。更多信息请参见[LTS搜索语法介绍](#)。

搜索日志

- 步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2** 在“日志管理”页面，单击目标日志组或日志流名称，进入日志详情页面。
- 步骤3** 在“日志搜索”页面，您可以根据自己的实际需求，在时间范围选择框，即可查看不同时间段的日志数据。

图 6-10 时间范围



时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- **相对时间**：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- **整点时间**：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- **自定义**：表示查询指定时间范围的日志数据。支持选择3个月的范围，白名单用户支持选择6个月的范围。如有需要，请[提交工单](#)申请。

- 步骤4** 根据[LTS搜索语法介绍](#)在搜索框输入查询条件，用于日志数据的查看、简单搜索和过滤。

图 6-11 日志搜索



1. 在页面搜索区域中，单击搜索框，输入待搜索的关键字，或在弹出的下拉框中选择待搜索的字段和关键词，单击“查询”，开始搜索。
 - 内置保留字段有appName、category、clusterId、clusterName、collectTime等，默认简化显示，并且hostIP、hostName、pathFile默认显示在最前面。更多内置保留字段请参考[创建LTS日志索引](#)。
 - 结构化配置的字段按照key:value显示。
 - 日志搜索框内容过多时支持自动换行并多行显示。支持固定搜索框高度。
2. 在页面搜索区域，使用键盘的“↑”“↓”箭头，选择待搜索的关键字或搜索语法，单击Tab键选中后，单击“查询”，开始搜索。
3. 对已创建快速分析的字段，单击选择字段可直接将其添加到页面搜索框中，进行搜索。请参考[创建LTS快速分析](#)开启快速分析。

通过单击字段添加到搜索框中，如果是同一字段，则将直接替换该方式添加的字段，不会进行AND搜索；如果是不同字段，则对不同字段进行AND搜索。

步骤5 在“日志搜索”页签，您可以进行如下操作，更多操作请参考[日志搜索的常用操作](#)。

1. 在日志统计下方，展示不同时间段的日志总条数柱状图，柱状图旁边显示日志条数刻度。

图 6-12 日志统计



若使用了内嵌功能，支持设置收起或展开式日志条数统计图。关于内嵌参数请参考[云日志服务地址](#)。

2. 在日志内容下方，鼠标悬浮指向**日志内容**中的字段，单击蓝色字体的日志内容，支持以下方式搜索日志：复制、添加到查询、从查询中排除、添加到查询（交互模式）、从查询中排除（交互模式）、新建查询、隐藏。
3. 在日志内容下方，支持选择列表或原始日志方式展示日志内容。

日志高亮的实现原理是当查询条件命中日志后，再将日志与查询条件进行字符串匹配，相同日志部分增加高亮标签，该部分会在界面高亮展示。因此当查询条件比较复杂，尤其是存在或关系时，在界面高亮显示的内容可能会比实际查询结果显示的多。

在不上传原始日志的情况下，全文搜索中文时，可能会出现中文不高亮的情况，如果需要高亮字段（比如：field1）内容中的中文，请使用“field1: 中文关键词”形式。

步骤6 在日志搜索页面，当搜索数据量过大时，LTS开始执行自动迭代查询日志。

- 自动迭代查询日志过程中，搜索框不支持输入，您可以单击“暂停查询”中断自动查询过程，若需要继续查询日志，单击“继续查询”恢复自动查询。建议您等待当前自动迭代查询完成后再发起新的查询任务。
- 对于较长时间的迭代查询，建议您缩小时间范围或添加过滤条件减少迭代次数。

步骤7 支持设置日志数据的版面展示，主要设置字段是否显示或简化显示。

1. 在下拉框单击编辑版面，进入版面设置页面，版面列表自带默认版面、纯净版面、容器日志默认版面，可以设置字段在版面是否显示。

图 6-13 编辑版面



- 云端：适用于有写权限的用户，版面配置信息保存在云端。支持设置账号级别或用户级别的版面配置。
 - 本地缓存：适用于只有读权限的用户，版面配置信息缓存在本地浏览器。
2. 在版本设置页面的版本列表下方，单击“+”新增自定义版面，设置版面名称和版面字段的可见性。

图 6-14 版面设置



新创建的版面可以进行如下操作：

- 编辑：单击  可以编辑版面的显示名称。
- 复制：鼠标悬浮  上，勾选复制到本地缓存，或者勾选云端-账号级别或云端-用户级别，单击“确定”即可复制版面信息。
- 删除：单击  ，单击“确定”即可删除版面信息。

 **警告**

删除版面信息后无法恢复，请谨慎操作。

3. 设置完成后，单击“确定”，返回下拉框显示新增的自定义版面。

步骤8 在“日志搜索”页面，选择列表，支持设置显示列。

1. 选择列表展示日志时，鼠标悬浮在按钮上，单击“显示列设置”，在弹出的日志折叠设置页面中，设置显示列的信息。

图 6-15 显示列设置



2. 时间列、content默认开启显示，在显示列下方，全量显示可见的字段名称，支持设置字段名称的别名，合理设置字段别名可以显著提高日志的可读性、可维护性和查询效率。设置完成后，单击“确定”。

图 6-16 显示列设置页面



3. 您可以通过[创建快速分析](#)或[步骤7](#)显示或隐藏字段。

----结束

交互模式

使用交互模式之前，请确保日志正常上报，且已经完成结构化分析和索引配置。详情请参考[使用ICAgent插件采集主机日志](#)和[创建LTS日志索引](#)。

若用户在日志接入LTS时没有配置ICAgent结构化解析，可以单独给目标日志流配置ICAgent结构化解析或云端结构化解析。推荐使用ICAgent结构化解析的方式，更多内容请参考[配置ICAgent结构化解析](#)。

交互式搜索适用于生成简单的搜索语句，操作简单，通过在界面上设置搜索条件和指定日志查询时的过滤规则，从而筛选日志中满足条件的记录。如果您需要使用更多的函数或者嵌套查询，请手动输入SQL语句。详情请参考[SQL分析语法介绍](#)。

步骤1 单击搜索框前面的“交互式搜索”，进入交互式搜索页面。

图 6-17 交互模式



步骤2 在下拉框选择日志搜索的字段和条件，搜索框则会展示对应字段的值。根据业务需要，用户可以自定义设置搜索方式，添加关联关系或添加组进行搜索。

- 下拉框显示的字段为索引配置字段、结构化字段、[内置保留字段](#)。
- 且（AND）表示在多个条件中需要同时满足所有条件才能成立。
- 或（OR）表示在多个条件中只需满足其中一个条件即可。

例如在界面上选择content、collectTime、category字段，设置不同的条件，随时可以预览搜索语句，如有问题，可以修改搜索条件，操作简单。

（截图仅供参考，请以实际搜索结果为准）

(content:"{"write_traffic":11,"index_traffic":133,"storage":17107}" AND collectTime=1728631667089)
OR category:LTS



步骤3 设置完成后，单击“确定”，LTS根据搜索语句进行日志搜索，在日志搜索下方即可查看搜索结果。

----结束

日志搜索的常用操作

在日志内容展示区域，支持分享日志、查看上下文、下载日志等操作，具体参考[表 6-14](#)。

如果出现服务网络不稳定、系统负载过高或其他不可预见的因素时，日志上报到LTS可能会出现极少日志的重复或丢失的情况。

表 6-14 常用操作

操作	说明
创建快速查询	单击  按钮，创建快速查询。
查看仪表盘	单击  按钮，在弹出来的仪表盘页面中，可查看已创建的仪表盘。
添加告警	单击  按钮，在弹出的页面中，支持 新建告警规则 。
分享日志	单击  复制当前日志搜索页面的链接，用于分享搜索日志。
刷新日志	单击  对日志进行刷新，有两种方式刷新方式：手动刷新和自动刷新。 - 手动刷新：单击“手动刷新”，可直接对日志进行刷新。 - 自动刷新：选择自动刷新的间隔时间，将对日志进行自动刷新。间隔时间范围为15秒、30秒、1分钟和5分钟。
复制	单击  复制日志内容。
查看上下文	单击  查看日志上下文。 支持选择简洁模式查看日志上下文。支持下载上下文内容。
更多操作	单击  进入该时间段的日志详情页，查看更多日志信息。 - 在“扩展字段”页签，查看字段名称、字段值，在对应字段操作列，支持对该字段进行添加到查询、从查询中排除、字段存在、字段不存在、隐藏的方式搜索日志。 - 在“json格式”页签，查看日志的json格式。 - 在“上下文日志”页签，支持设置查询行数、过滤字段、下载日志、简洁模式等操作。
换行/取消换行	单击  按钮，搜索的日志内容将换行显示，默认开启换行按钮。若不需要换行，单击  按钮，取消换行。

操作	说明
下载日志	支持将日志文件或日志查询分析结果下载到本地进行查看。 请确保当前日志流上传了原始日志，否则下载后文件将没有内容。 1. 单击  按钮。 2. 单击“下载日志”，在弹出的下载日志页面中支持前端直接下载、后台离线下线和前往创建转储。开通一次性转储后才能使用后台离线下线和前往创建转储。 说明 一次性转储功能仅支持白名单用户使用，如有需要，请提交工单申请开通。 - 前端直接下载：直接将前端查询的结果保存到本地，下载记录不会出现在日志下载历史中。单次下载支持最大5,000条日志。 勾选“.csv”或“.txt”格式，单击“开始下载日志”，将日志导出至本地。 - 后台离线下线：通过后台任务将日志文件下载到临时OBS桶中，您的浏览器需要有公网访问权限才能在日志下载历史中下载结果文件。单次下载支持最大2,000万条日志。 勾选“.csv”或“.txt”或“.json”，单击“开始下载日志”，将日志导出至本地。 日志转储成功后，云日志服务根据您的日志转储量进行收费。转储费用详情请参见LTS价格详情。 - 前往创建转储：通过OBS转储任务下载日志文件，单次下载支持最大2,000万条日志。 单击“前往创建转储”，跳转至配置转储页面创建一次性转储任务，详细请参考配置日志转储（一次性）。 日志转储成功后，云日志服务根据您的日志转储量进行收费。转储费用详情请参见LTS价格详情。 3. 单击“日志下载历史”，在日志下载历史页面，支持查看、下载、复制日志下载链接、删除日志下载记录。 加载完成的日志文件只在“日志下载历史”页面保留1天，请及时下载到本地保存。
全部折叠/全部展开	单击  设置日志内容展示的行数。若不需要展示日志内容，再 单击一次  按钮即可关闭展示的日志内容。

操作	说明
JSON设置	鼠标悬浮在按钮上，单击“JSON设置”，在弹出的JSON设置页面中，设置格式化显示。 默认开启格式化，JSON默认展开层级为2层。若日志包含多个反斜杠，当日志展示为json格式时，会丢失一个反斜杠，因为json解析会将第一个反斜杠作为转义符处理。 - 开启格式化按钮：设置JSON默认展开层级，最大设置为10层。 - 关闭格式化按钮：对于JSON格式的日志，将不会格式化层级显示。
日志折叠设置	鼠标悬浮在按钮上，单击“日志折叠设置”，在弹出的日志折叠设置页面中，设置长日志字符个数。 日志超过设置的长日志字符个数时，超出字符将被隐藏，单击“展开”按钮可查看全部内容。 默认开启自动折叠长日志，字符个数默认为400个。
日志时间展示	鼠标悬浮在按钮上，单击“日志时间展示”，在弹出的日志折叠设置页面中，默认开启展示毫秒，设置是否展示毫秒、是否展示时区。
不可见字段列表	该列表展示版面设置中配置的不可见性字段。  - 当日志流未配置版面设置时，将不显示按钮。 - 当日志内容为“CONFIG_FILE”且未配置版面设置时，不可见字段默认有appName、clusterId、clusterName、containerName、hostIPv6、NameSpace、podName和serviceID。

跨流搜索

在当前日志流详情页面，支持跨流搜索日志，即无需退出当前日志搜索页面，即可选择其他日志流搜索日志。

步骤1 在当前日志流详情页面，单击。

图 6-18 跨流搜索



步骤2 在弹出的跨流搜索页面，选择需要搜索的目标日志流。

图 6-19 目标日志流

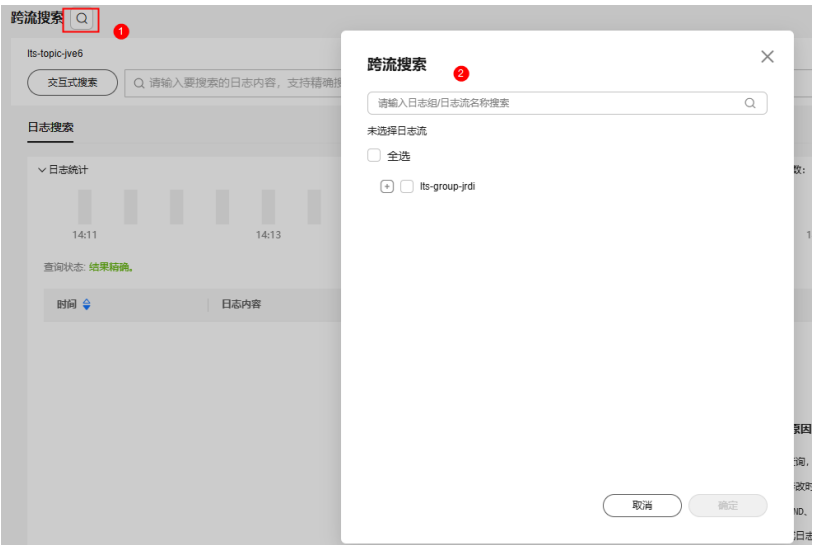


步骤3 单击“确定”。

步骤4 进入目标日志流详情页，根据业务需要参考[搜索语法介绍](#)进行搜索日志。

若需要更换日志流，单击左上角跨流搜索旁边，在弹出的页面重新选择日志流进行搜索。

图 6-20 更换日志流



步骤5 搜索完成后，单击右上角的“退出跨流搜索”，即可返回当前日志流搜索页面。

- 当跨流搜索的日志流的分词配置不同时，会导致跨流日志关键词高亮不一致。
- 最多可同时搜索50个日志流。
- 仅支持搜索原始日志，暂不支持使用SQL进行统计分析。
- 支持跨日志组选择日志流，但不支持跨区域选择日志流。
- 跨流搜索选择多个索引配置不一致的日志流，例如A日志流配置了字段索引Key1，B日志流没有配置该字段，使用搜索语句**Key1:xxx**搜索日志时，仅能查看A日志流的相关日志，B日志流搜索日志失败，请根据界面报错提示修改后再重新搜索日志。

----结束

相关文档

- 日志接入LTS成功后，您可通过云日志服务LTS的日志告警功能，通过设置告警规则提醒您日志中的异常情况，请参考[配置日志告警规则](#)。
- 如果在搜索日志时遇到问题，您可以参考[日志搜索与分析常见问题](#)获取解决方案。
- 为您推荐几种在真实使用场景下日志搜索与分析的实践教程。更多信息请参考[日志搜索与分析最佳实践](#)。
 - a. 云日志服务LTS支持分页显示查询分析结果，可控制每次返回的日志数量。您可以参考[分页显示查询分析结果](#)。
 - b. WAF接入LTS后，支持通过LTS分析WAF日志。您可以参考[在LTS页面分析华为云WAF日志](#)。
 - c. LTS支持将日志查询界面嵌入到客户自建系统。您可以参考将[LTS日志查询页面嵌入用户自建系统](#)。

6.5 搜索语法介绍

云日志服务（LTS）提供一套搜索语法用于设置搜索条件和指定日志查询时的过滤规则，从而筛选日志中满足条件的记录，筛选结果可以用于分析语句，进行更复杂的分析处理。

为了快速了解并使用搜索语法，建议您了解以下信息：[搜索方式](#)、[短语搜索](#)、[运算符](#)、[搜索语句示例](#)。

- 使用搜索语法前，请您在索引配置处设置对应分词符，如无特殊需要，可直接使用默认的分词符，";=()[]{}@<>/:\\?\\n\\t\\r。
- 搜索语法不支持对分词符进行搜索。例如搜索语句`var/log`，其中`/`为分词符，搜索语句等同于`var log`，搜索的是同时包含`var`和`log`的所有日志。同理，搜索语句`"var:log"`、`var;log`等搜索的也是同时包含`var`和`log`的所有日志。
- 查询日志使用搜索语法的常见问题和相关报错的处理方法请参考[日志搜索相关问题](#)。

搜索方式

搜索语句是用来指定日志搜索时的过滤规则，返回符合条件的日志。

根据索引配置方式可分为全文搜索和字段搜索，根据搜索精确程度可分为精确搜索和模糊搜索。其他类型的搜索方式包括范围搜索、短语搜索等。

表 6-15 搜索方式说明

搜索方式	说明	示例
全文搜索	配置全文索引后，日志服务根据您设置的分词符将整条日志拆分成多个关键词。 - content为日志原文对应的内置字段，搜索语句GET等同于content:GET，默认匹配日志原文的内容。 - 多个关键词默认通过AND连接，搜索语句GET POST等同于GET and POST。	- GET POST - GET and POST - content:GET and content:POST 上述三个搜索语句功能相同，均表示搜索同时包含关键词GET和POST的日志。
字段搜索	配置字段索引后，您可以指定字段名称和字段值（key:value）进行搜索。根据字段索引中设置的数据类型，您可以进行多种类型的基础搜索和组合搜索。 - value参数不可为空，通过搜索语句key:""匹配字段值为空的日志。 - 字段搜索和 not 运算符配合使用时，还会匹配到不包含该字段的日志。	- request_time>60 and request_method:po*表示搜索request_time字段值大于60且request_method字段值以po开头的日志。 - request_method:""表示搜索request_method字段值为空的日志。 - not request_method:GET表示搜索不包含request_method字段和request_method字段值不为GET的日志。
精确搜索	使用精确的词进行搜索。 日志服务搜索采用的是分词法，搜索时不会保证关键词出现的顺序。 搜索语句为abc def，会匹配所有同时包含abc和def的日志，日志abc def或者def abc都会命中，如果需要确保关键词出现的顺序，请您采用短语搜索 #"abc def"。	- GET POST表示搜索同时包含关键词GET和POST的日志。 - request_method:GET表示搜索request_method字段值包含GET的日志。 #"var/log"表示搜索包含短语/var/log的日志。

搜索方式	说明	示例
模糊搜索	在搜索语句中指定一个词，在词的中间或者末尾加上模糊搜索关键字，即星号（*）或问号（?），云日志服务会在所有日志中查询到符合条件的100个词，返回包含这100个词并满足查询条件的所有日志。指定的词越精确，查询结果越精确。 - 星号（*）代表匹配多个字符，问号（?）代表匹配1个字符。 - 当星号（*）和问号（?）作为分词符时，不具备模糊搜索功能，其中问号（?）为默认的分词符，使用其模糊搜索功能前需将其从分词符中移除。 - 星号（*）或问号（?）不能用在词的开头。 - long数据类型和float数据类型不支持使用星号（*）或问号（?）进行模糊搜索。 - 当模糊条件前缀很短且日志中符合条件的词超过100个时，查询结果会不精确。	- GE*表示在所有日志中查找以GE开头的词，并返回包含这些词的日志。 - request_method:GE*表示在所有日志中查找request_method字段值以GE开头的词，并返回包含这些词的日志。
范围搜索	long数据类型和float数据类型支持范围搜索。 string类型的字段不支持范围查询。 - 方式1：通过 =（等于）>（大于）<（小于）运算符搜索日志。 - 方式2：通过 in 运算符搜索日志，支持修改开闭区间。	- request_time>=60表示在所有日志中查找request_time字段值大于等于60的日志。 - request_time in (60 120]表示在所有日志中查找request_time字段值大于60且小于等于120的日志。
短语搜索	短语搜索用于完全匹配日志中的目标短语，可以确保关键词出现的顺序。 短语搜索不支持模糊搜索	#"abc def"表示在所有日志中查找包含目标短语abc def 的日志。

- 分词符

云日志服务LTS会根据分词符，将日志内容拆分成多个词。日志服务默认配置的分词符为, "" ;=() [] {} @&<> /: \? \n \t \r。

例如：日志2023-01-01 09:30:00，默认分词符会将其分为四部分：2023-01-01、09、30、00。

此时搜索语句2023无法匹配到该条日志，可以通过2023-01*或2023-01-01搜索到该条日志。

如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整日志内容或模糊搜索查找对应的日志。

- 关键词顺序
只有短语搜索`#"abc def"`才能保证关键词出现的顺序，其他搜索方式多个关键词默认AND连接。
例如：`request_method:GET POST`查询的是同时包含GET和POST的日志，不会保证GET和POST的顺序。如有需要推荐采用短语搜索。
- 中文搜索
中文搜索时不需要采用模糊查询，如有需要推荐采用短语搜索，可以匹配到更精确的结果。
云日志服务LTS的英文是以单词的形式进行拆分的，单词的长度不一致，因此可以通过模糊搜索匹配拥有相同前缀英文单词的日志。
中文采用的是一元分词，每个字都是独立的，拆分后每部分的长度都是1。
例如：搜索语句**星期一**，代表搜索同时包含**星、期、一**的日志；搜索语句`#"星期一"`，代表搜索包含目标短语**星期一**的日志。
- 语法关键词
日志搜索语句的语法关键词包括：`&& || AND OR and or NOT not in : > < = ( ) [ ]`
其中 **and AND or OR NOT not in** 作为语法关键词使用时，前后需要使用空格分隔；
如果日志中本身包含语法关键词且需要搜索时，搜索语句需要用**双引号**包裹，否则可能会导致语法错误或搜索到错误的结果。
例如：搜索语句`content:and`，包含语法关键词 **and**，需要修改为`content:"and"`。

短语搜索

短语搜索用于准确匹配目标短语，例如搜索语句`#"abc def"`，区分先后顺序，将匹配所有同时包含abc、def，且abc位于def前面的日志。短语搜索和关键词搜索的区别请参考表6-16。

- 短语搜索：在关键词搜索语法的基础上实现，短语搜索能够区分关键词的顺序，用于精准匹配目标短语，搜索结果更加精确。短语搜索适用于英文短语、中文短语的搜索，不支持模糊搜索。
- 关键词搜索：关键词搜索是基于分词实现，通过分词符先将搜索内容拆分为多个关键词，然后匹配日志。关键词搜索不会区分多个关键词在日志中出现的顺序，因此只要日志中按照搜索的与或非逻辑能命中关键词，该日志就会被搜索到。

表 6-16 搜索区别

搜索方式	说明	示例
短语搜索	区分关键词的顺序，用于精准匹配目标短语，搜索结果更加精确。	假设您的日志流中存在两条原始日志，如下： - 原始日志1：this service is lts - 原始日志2：lts is service 则搜索：#"is lts"，会命中1条日志；搜索：#"lts is"，会命中1条日志。
关键词搜索	不区分关键词的顺序，按照搜索逻辑命中关键词即可。	假设您的日志流中存在两条原始日志，如下： - 原始日志1：this service is lts - 原始日志2：lts is service 则搜索：is lts，会命中2条日志；搜索：lts is，会命中2条日志。

使用限制如下：

- 短语搜索不支持搭配模糊搜索。
短语搜索中的星号（*）和问号（?）会被视为普通字符，因此短语搜索不支持搭配模糊搜索，可以用来搜索日志中的星号（*）和问号（?）。
- 短语搜索不支持对分词符进行搜索。
例如搜索语句#"var/log"，其中 / 为分词符，搜索语句等同于#"var log"，会搜索包含目标短语var log的日志。同理，搜索语句#"var:log"、#"var;log"等搜索的也是包含目标短语var log的日志。
- 中文搜索推荐采用短语搜索。
由于中文默认采用的是一元分词，每个汉字单独分词，搜索时会匹配同时包含搜索语句中每一个汉字的日志，本身便具有模糊搜索的特性，当需要更加精确的结果时，推荐采用短语搜索。

运算符

搜索语句支持的运算符请参考[表6-17](#)。

- 除in运算符外，其他运算符不区分大小写。
- 运算符的优先级由高到低排序如下所示：
 - a. 冒号（:）
 - b. 双引号（" "）
 - c. 圆括号（）
 - d. and、not

e. or

表 6-17 运算符说明

运算符	说明	示例
and	与运算符，如果多个关键词之间没有语法关键词，默认为and关系。 and作为运算符使用时前后需要使用空格分隔。例如 1 and 2 代表搜索同时包含1 和2的日志； 1and2 代表搜索包含词语 1and2 的日志。	GET 200等同于GET and 200
AND	与运算符，等同于and。	GET AND 200
&&	与运算符。 &&作为运算符使用时不需要使用空格分隔。例如： 1 && 2 等同于 1&&2 ，代表搜索同时包含1 和2的日志。	1&&2
or	or运算符。 or 作为运算符使用时前后需要使用空格分隔。	request_method:GET or status:200
OR	或运算符，等同于or。	request_method:GET OR status:200
	或运算符。 作为运算符使用时不需要使用空格分隔。	request_method:GET status:200
not	非运算符。 - not 作为运算符使用时需要使用空格分隔。 - not 运算符和字段搜索配合使用时还会匹配到不包含对应字段的日志。	request_method:GET not status:200、not status:200
()	用于提高括号内搜索条件的优先级。	(request_method:GET or request_method:POST) and status:200
:	用于字段搜索（key:value）。 如果字段名称（key）或者字段值（value）内有空格或冒号（:）等保留字符，请使用双引号（" "）包裹字段名称（key）或者字段值（value）。例如： "request method":GET - message:"This is a log" - time:"09:00:00" - ipv6:"2024:AC8:2ac::d09"	request_method:GET

运算符	说明	示例
""	使用双引号（""）包裹一个语法关键词，可以将该语法关键词转换成普通字符，例如："and"表示搜索包含and的日志，此处的and不代表运算符。	request_method:"GET"
\	转义符号，用于转义双引号（""），转义后的引号表示符号本身。	例如：日志内容为instance_id:nginx"01"，您可以使用instance_id:nginx\"01\"进行查询。
*	通配符搜索，匹配零个、单个、多个字符。 *不支持放在关键词开头，推荐放在关键词的中间部分或者结尾。	request_method:P*T
?	通配符搜索，匹配单个字符。 ?不支持放在关键词开头，推荐放在关键词的中间部分或者结尾。	例如：request_method:P?T，可以匹配到PUT，无法匹配到POST。
>	搜索某字段值大于某数值的日志。	request_time>100
>=	搜索某字段值大于或等于某数值的日志。	request_time>=100
<	搜索某字段值小于某数值的日志。	request_time<100
<=	搜索某字段值小于或等于某数值的日志。	request_time<=100
=	搜索某字段值等于某数值的日志，仅适用于float、long类型的字段。对于该类型的字段，等号（=）和冒号（:）作用相同。	request_time=100等同于request_time:100
in	搜索某字段值处于某数值范围内的日志，中括号表示闭区间，小括号表示开区间，两个数字之间使用空格分隔。 in只能为小写字母，且作为运算符使用时前后需要使用空格分隔。	- request_time in [100 200] - request_time in (100 200]
#""	用于搜索包含目标短语的日志，可以保证关键词出现的顺序。 短语搜索中的星号（*）和问号（?）会被视为普通字符，因此短语搜索不支持模糊搜索，可以用来搜索日志中的星号（*）和问号（?）。	request_method:#"GET POST"

搜索语句示例

同一条搜索语句，针对不同的日志内容和索引配置时，会有不同的搜索结果。本文基于如下日志样例和索引介绍搜索语句示例。

图 6-21 搜索示例

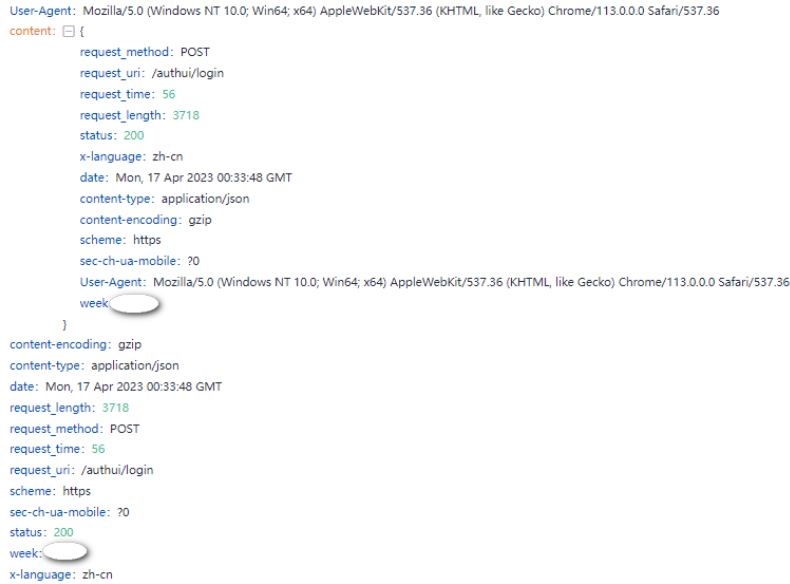


表 6-18 普通搜索示例

搜索需求	搜索语句
搜索POST请求且状态码为200的日志。	request_method:POST and status=200
搜索GET请求或POST请求成功（状态码为200~299）的日志。	(request_method:POST or request_method:GET) and status in [200 299]
搜索GET请求或POST请求失败的日志。	(request_method:POST or request_method:GET) not status in [200 299]
搜索非GET请求的日志。	not request_method:GET
搜索GET请求成功且请求时间小于60秒的日志。	request_method:GET and status in [200 299] not request_time>=60
搜索请求时间为60秒的日志。	- request_time:60 - request_time=60
搜索请求时间大于等于60秒，并且小于200秒的日志。	- request_time>=60 and request_time<200 - request_time in [60 200)
搜索包含and的日志。	content:"and" 此处使用双引号将and包裹，and为普通字符串，不代表运算符。

搜索需求	搜索语句
搜索不存在user字段的日志。	not user:*
搜索user字段值为空的日志。	user:""
搜索星期字段值不为星期一的日志。	not week:星期一
搜索sec-ch-ua-mobile字段值为?0的日志。	sec-ch-ua-mobile:#"?0" 日志内容中包含*或?且需要搜索时，需要采用短语查询。

更复杂的搜索示例请参考[表6-19](#)。

表 6-19 模糊搜索

搜索需求	搜索语句
搜索包含以GE开头的词的日志。	GE*
搜索包含以GE开头，结尾只有一个字符的词的日志。	GE?
搜索request_method字段值包含以G开头的词的日志。	request_method:G*
搜索request_method字段值包含以P开头，以T结尾，中间还有单个字符的词的日志。	request_method:P?T
搜索request_method字段值包含以P开头，以T结尾，中间包含零个、单个或多个字符的词的日志。	request_method:P*T

基于分词符的搜索，例如：User-Agent字段值为**Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36**。

- 设置分词符为空时，该字段值将被当成一个整体，则您使用**User-Agent:Chrome**搜索语句进行搜索时，无法搜索到日志。
- 设置分词符为, "";"=()[]{}?@&<>/:\\n\\t\\r后，该字段值会被拆分为**Mozilla、5.0、Windows、NT、10.0、Win64、x64、AppleWebKit、537.36、KHTML、like、Gecko、Chrome、113.0.0.0、Safari、537.36**。
此时可以使用**User-Agent:Chrome**等搜索语句进行搜索。

表 6-20 基于分词符的搜索

搜索需求	搜索语句
搜索User-Agent字段值中包含Chrome的日志。	User-Agent:Chrome
搜索User-Agent字段值中包含以Win开头的词的日志。	User-Agent:Win*
搜索User-Agent字段值中包含Chrome和Linux的日志。	User-Agent:"Chrome Linux"
搜索User-Agent字段值中包含Firefox或Chrome的日志。	User-Agent:Chrome OR User-Agent:Linux
搜索User-Agent字段值包含Chrome，但不包含Linux的日志。	User-Agent:Chrome NOT User-Agent:Linux

6.6 分析日志

日志分析提供对结构化后的日志字段进行SQL查询与分析的功能。对原始日志结构化后，等待1~2分钟左右，即可对结构化后的日志进行SQL查询与分析。

约束与限制

此功能支持全部用户使用的局点有：华南-广州、华北-北京四、华北-乌兰察布一、华东-上海一、中国-香港、西南-贵阳一、亚太-新加坡、华北-北京一，支持部分白名单用户使用的局点有：华东-上海二、亚太-曼谷、华南-深圳、中东-利雅得、亚太-雅加达，其他局点暂不支持该功能。

前提条件

- 已成功采集到日志。
- 对日志内容已完成结构化配置，具体操作请参考[设置云端结构化解析日志](#)。
日志结构化后字段名称与系统SQL内置保留字段名称相同，或者字段名称中带有中划线、下划线、小数点这三种特殊字符时，SQL查询需要加英文双引号。系统SQL内置保留字段名称包括："time"、"select"、"where"等。
- 在索引配置页面开启“日志分析”才能使用SQL日志分析，详细请参考[创建LTS日志索引](#)。

分析日志

- 步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2** 单击目标日志组或日志流名称，进入日志详情页面。
- 步骤3** 选择“日志分析”页签。
- 步骤4** 在可视化页面支持交互式分析，通过该模块配置简单的分析语句，查询可视化数据，配置可视化图表。设置过滤条件，通过添加指标、添加分组、添加排序进行数据分析，方便用户操作。

图 6-22 日志分析



步骤5 选择时间范围，参考[SQL分析语法介绍](#)输入SQL语句，单击“查询”，在下方区域通过不同类型图表展示搜索结果。

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。
- SQL查询约束有：
 - a. 单次查询返回结果最多10W条。
 - b. 当聚合结果超过10W时，聚合结果可能存在误差。
- SQL查询语句中，string类型的where条件的键值有限制：
 - a. 精确查找value需添加英文单引号，模糊查找value需添加英文单引号或者双引号，key与SQL内置保留字段名称相同时需添加英文双引号。
 - b. 建议使用where条件时，使用where "key"='value'，或者where "key" like '%value%'。
- SQL查询语句中，float和long类型的where条件不受限制，但当与关键词冲突时可能会导致查询异常，建议使用where "key"='value'，或者where "key" like '%value%'进行查询。
- 日志搜索框支持自定义上下拖动调整高度。
- 输入搜索语法后，单击  设置格式化sql和反格式化sql，优化搜索语句，提高搜索效率。

步骤6 当设置时间范围内日志量超过10亿行时会触发迭代查询，此时第一次查询结果界面显示为“查询状态：结果不精确”，可以多次单击查询按钮通过迭代查询分多次完成全部日志的查询，直到界面最终显示为“查询状态：结果精确”。

步骤7 根据SQL查询返回的数据，依照业务需求选择不同图表类型，呈现查询结果。详细请参考[使用统计图表将日志可视化](#)。

步骤8 对查询结果可执行如下操作：

- 单击“新建”，在弹出的“创建可视化图表”中，根据业务需求填写“图表名称”，开启“同时添加到仪表盘”，单击“确定”，可视化图表保存成功。
- 单击“保存”，对在弹出的“保存可视化图表”中，根据业务需求填写“图表名称”，开启“同时添加到仪表盘”，单击“确定”，可视化图表保存成功；当选定某个可视化图表时，单击“保存”，可对该图表进行修改。
- 单击“另存为”，在弹出的“另存为可视化图表”中，根据业务需求填写“图表名称”，开启“同时添加到仪表盘”，单击“确定”，对已有可视化图表进行复制。需要先保存一个图表后，才可另存为可视化图表。
- 单击“下载”，可下载当前SQL查询结果的可视化数据，该文件为.csv。
- 单击  按钮添加告警，在弹出的“新建告警规则”中，为选中的可视化图表配置[配置日志告警规则](#)。需要先保存一个图表后，才能新建告警规则。
- 单击“展开图表”，可对当前日志流下的可视化图表展开；单击“收起图表”，可收起当前日志流下展开的可视化图表。

----结束

6.7 创建快速分析

日志包含了系统性能及业务等信息，例如关键词ERROR的多少反映了系统的健康度，关键词BUY的多少反映了业务的成交量等。当您需要了解这些信息时，可以通过快速分析功能，查询指定日志关键词，云服务日志LTS针对您配置的关键词进行统计，并生成指标数据，以便您实时了解系统性能及业务等信息。

- 通过采样快速统计字段值取值分布，并非对全量数据进行分析，采样条数默认值10万条，最大值1000万条，采样条数越多分析数据越慢。您可以在索引配置页面设置快速分析采样条数，详细操作请参考[创建LTS日志索引](#)。
- 支持通过查询时间和查询条件过滤日志进行分析。
快速分析是对通过查询语句查询到的日志进行分析，当查询到的日志数目为零时，快速分析没有结果。
- 支持将快速分析生成查询语句。
单击快速分析的某一行分析结果，可自动生成查询语句，查询日志并生成新的快速分析。
- 在快速分析的字段中，当分析时间范围内未开启快速分析、字段不存在或字段值为null时，分析结果为null。
 - 当字段为string类型时，单击null添加到搜索框中将显示为字段："null"OR NOT 字段：*。
 - 当字段为float或long类型时，单击null添加到搜索框中将显示为NOT 字段：*。
 - 未开启快速分析期间，不会存储分析所用的列存数据，分析结果对应为null，此时查询日志没有实际含义，不一定能匹配到日志。

前提条件

- 快速分析的对象为结构化日志中提取的关键字段，创建快速分析前请确保日志正常上报，且已经完成结构化分析和索引配置。详细请参考[使用ICAgent插件采集主机日志](#)和[创建LTS日志索引](#)。

- 若用户在日志接入LTS时没有配置ICAgent结构化解析，可以单独给目标日志流配置ICAgent结构化解析或云端结构化解析。推荐使用ICAgent结构化解析的方式，更多内容请参考[配置ICAgent结构化解析](#)。

创建快速分析

快速分析以日志流为单位，请参考如下步骤创建快速分析。

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 单击目标日志组或日志流名称，进入日志详情页面。

步骤3 在“日志搜索”页签，单击快速分析后面的  进入“索引配置”页面，在字段索引下方，单击“自动配置”，LTS自动配置字段索引，同时开启快速分析。

图 6-23 快速分析



步骤4 单击“确定”，快速分析创建完成。

- 在快速分析下方的搜索框输入字段名称，即可显示搜索结果，方便您快速定位目标字段。
- 字段前面显示abc表示String类型字段。
- 字段前面显示1.2表示float类型字段。
- 字段前面显示123表示long类型字段。
- 快速分析的字段长度最大为2000字节。
- 快速分析字段展示前100条数据。
- 支持显示当前字段的采样数量。
- 单击  即可查看一键生成的图表展示，string类型的字段支持展示字段分布值统计和智能聚合时间折线图，long和float数值类型的字段只支持展示智能聚合时间折线图。单击图表即可进入详情页面。

图 6-24 string 类型字段展示图表



- 单击字段分布值统计或智能聚合时间折线图，会自动跳转到“日志分析”页面并生成对应的SQL查询语句进行查询，更加直观地展示字段值的分布和变化趋势。更多信息请参见[分析日志](#)。

步骤5 设置隐藏或显示字段。

- 在显示字段下方的目标字段后面单击  即可隐藏单个字段。
- 在显示字段旁边，单击  即可一键隐藏全部字段。
- 在隐藏字段下方的目标字段后面单击  即可显示单个字段。
- 在隐藏字段旁边，单击  即可一键显示全部字段。

----结束

二次分析

快速分析功能支持对float类型和long类型的字段进行二次分析，具体如下：

快速统计最大项、最小项、平均值和总和，分别单击目标字段下的**Max**、**Min**、**Avg**、**Sum**，快速查找所有项中的最大项、最小项、平均值和总和。

6.8 保存搜索与分析语句

当您需要重复使用某一个搜索与分析语句时，可以将其设置为快速查询语句，通过保存的快速查询语句，可实现对日志的快速执行搜索与分析操作。

创建快速查询

- 步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2** 单击目标日志组或日志流名称，进入日志详情页面。
- 步骤3** 在“日志搜索”页签，单击 ，输入“快速查询名称”和“快速查询语句”。默认开启快速查询和快速查询（保存本地）。

图 6-25 创建快速查询

创建快速查询

✕

★ 快速查询名称

请输入快速查询名称

?

★ 快速查询语句

请输入快速查询语句

快速查询

开启后，添加到快速查询列表中

快速查询 (保存本地)

开启后，添加到快速查询 (保存本地) 列表中

取消

确定

- 快速查询名称，用于区分多个快速查询语句。名称自定义，需要满足如下要求：
 - 只支持输入英文、数字、中文、中划线、下划线及小数点。
 - 不能以小数点、下划线开头或以小数点结尾。
 - 长度为1-64个字符。
- 快速查询语句，搜索日志时需要重复使用的关键字，例如“error*”。

步骤4 单击“确定”，完成快速查询条件的创建。在左侧导航栏的快速查询页签，即可查看到保存成功的语句。

- 单击快速查询语句的名称，即可查看日志搜索结果。
- 鼠标悬浮在快速查询语句上面，单击快速查询语句后面的，在打开的“编辑快速查询”页面，修改快速查询语句，单击“确定”。

图 6-26 编辑快速查询

日志流

快速查询

请输入快速查询名称

Q

快速查询

▼

containerfile-123

编辑

▼

test

快速查询 (保存本地)

▼

- 鼠标悬浮在快速查询语句上面，单击快速查询语句后面的，在打开的“删除快速查询”页面，单击“确定”即可删除快速查询语句。删除后将无法再执行快速查询，请谨慎操作。

----结束

6.9 查看上下文

查看上下文是指定其中一条日志，将该日志在原始日志中的前若干条（上文）或后若干条日志（下文）也查找出来。您可以通过查找指定日志生成时间点前后的日志，在业务故障排查中快速查找相关故障信息，用于在运维过程中快速定位问题。

功能优势

- 免除登录主机的繁琐步骤，在LTS控制台的日志搜索页面，快速查看任意指定日志的上下文信息。
- 结合事件发生的时间范围，在LTS控制台的日志搜索页面，选择日志时间范围即可快速查找可疑日志，然后再查询该日志的上下文，快速定位问题。

查看上下文

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 单击目标日志组或日志流名称，进入日志详情页面。

步骤3 在“日志搜索”页签下方的原始日志页面，单击。

图 6-27 查看上下文



步骤4 在弹出的上下文日志页面中，查看日志上下文，详细操作请参考[表6-21](#)。

图 6-28 上下文日志



表 6-21 查看上下文日志功能介绍

功能	说明
查询行数	根据需要选择查询日志的行数。
高亮显示	输入需要高亮的字符串，回车确认，在日志内容中高亮显示。
过滤日志	输入需要过滤的字符串，回车确认，在日志内容中高亮显示。当高亮显示和过滤日志同时设置时，均可高亮显示。
显示字段	查看上下文，默认字段为content，单击“显示字段”选择查看其他字段的上下文。
更早	从当前位置往前查看设置 查询行数 的二分之一。例如：当查询行数设置为100时，单击“更早”则从当前位置朝前显示50行，此时行号为-50；再次单击“更早”，依次叠加分别为-100、-150、-200.....
当前位置	当前日志位置。当设置了更早或更新时，单击“当前位置”可回到查看上下文开始的位置，即行数为0时。
更新	从当前位置往后查看设置 查询行数 的二分之一。例如：当查询行数设置为100时，单击“更新”则从当前位置朝后显示50行，此时行号为50；再次单击“更新”，依次叠加分别为100、150、200.....
简洁模式	• 打开“简洁模式”，只显示行号和content内容。 • 关闭“简洁模式”，显示日志详情。
下载	目前仅支持下载content字段内容到本地查看。

----结束

6.10 查看实时日志

日志接入云日志服务后，每隔大约1分钟上报一次。因此，在实时日志页面，您最多需要等待1分钟左右，即可查看实时上报的日志，实现对日志数据的快速检索与分析。

约束限制

- 若实时日志大于1MB且总数小于2000条时，LTS会清空前500KB的日志，保留最新的500KB日志。
- 若实时日志未超过1MB且总数超过2000条时，LTS会清空前1000条日志，保留最新的1000条日志。

前提条件

切换到实时日志页面后，有日志上报。

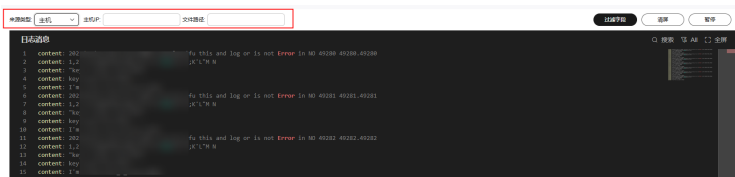
查看实时日志

如果您正在使用实时查看功能，请停留在实时查看页面，请勿切换页面。如果离开实时查看页面，实时查看功能将会停止，重新开启后上一次查看的实时日志将不会显示。

正常情况下，每隔5秒加载一次。如果这5秒内没有产生日志，则不显示；5秒后会继续调用接口，刷新出产生的日志数据。即如果每5秒都有日志数据产生，则加载数据时延为5秒。

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 单击目标日志组或日志流名称，进入日志详情页面。
- 步骤3 选择“实时日志”页签，查看实时日志。

图 6-29 实时日志



日志每隔大约1分钟上报一次，在日志消息区域，您最多需要等待1分钟左右，即可查看实时上报的日志。

有实时日志上报后，您可以进行如下操作：

- 来源类型选择主机时， 通过设置主机IP和文件路径筛选日志。
- 来源类型选择K8S时， 通过设置实例名称、容器名称或文件路径筛选日志。
- 字段过滤：从索引配置、结构化配置、最新日志获取字段信息，勾选需要过滤的字段查看日志。
- 清屏：清除日志消息区域已经显示出来的日志。
- 暂停：暂停日志消息的实时显示，页面定格在当前已显示的日志。
暂停后，“暂停”会变成“继续”，再次单击“继续”，日志消息继续实时显示。

----结束

6.11 SQL 分析语法介绍

6.11.1 SQL 查询语法概述

SQL是用于访问和处理数据库的标准计算机语言。LTS SQL提供了查询日志流中结构化数据的语句, 以下均将 LTS SQL 称为 SQL。

SQL语言由用于处理数据库和数据库对象的命令和函数组成。使用该语言时需遵循有关表达式和文本使用的规则。因此在SQL参考章节，除了SQL语法参考外，还会看到有关表达式、函数和操作符等信息。SQL基本查询语句如下：

约束与限制

此功能支持全部用户使用的局点有：华南-广州、华北-北京四、华北-乌兰察布一、华东-上海一、中国-香港、西南-贵阳一、亚太-新加坡、华北-北京一，支持部分白名单用户使用的局点有：华东-上海二、亚太-曼谷、华南-深圳、中东-利雅得、亚太-雅加达，其他局点暂不支持该功能。

语法格式

```
SELECT [ ALL | DISTINCT ] { * | exprs }  
FROM { <subquery> }  
[ WHERE where_condition ]  
[ GROUP BY [ col_name_list ]  
[ HAVING expr ]  
[ ORDER BY expr [ ASC | DESC ], expr [ ASC | DESC ], ... ]  
[ LIMIT limit ]  
[ OFFSET offset ]
```

数据类型

SQL查询中支持的数据类型如[表6-22](#)。如果当前字段数据类型需要改为其他数据类型，系统会进行数据类型的转换。例如STRING类型的字段转为LONG类型。字段数据类型转换之后的结果将会显示默认值，如STRING类型的数据转换为LONG类型的数
据，结果会显示为LONG类型的默认值0。同理，当空值被转换为非空类型值时，也会使用默认值进行替换。例如，当把STRING类型空值转换为数字类型时，将会返回默认
值0。

SQL语法中，字符必须被单引号（"）包裹，无符号或双引号（""）包裹的为字段或表
名称，如：'msg'表示字符串msg，msg或"msg"表示日志结构化msg字段。

表 6-22 SQL 查询支持的数据类型

原生数据类型	默认值	说明
STRING	""	原生STRING类型
FLOAT	0.0	原生FLOAT类型
LONG	0	原生LONG类型

查询语句

表 6-23 SQL 查询语句

语句	说明	示例
DISTINCT	返回去重后的结果。	SELECT DISTINCT visitCount
FROM	表示当前查询数据的源数据集， 可以是当前日志流的结构化数据, 也可以是当前日志流结构化数据的一个子集。 不加FROM的时候默认从当前日志流结构化数据查询，如果查询的数据源是一个子集， 则需要自己编写子查询语句。	SELECT visitCount

语句	说明	示例
WHERE	指定查询的过滤条件，支持算术运算符、关系运算符和逻辑运算符。具体过滤条件可填在where_condition处。	SELECT visitCount WHERE visitCount > 0
GROUP BY	指定作为分组依据的结构化字段，支持根据单字段或多字段分组。具体的结构化字段列表可填入col_name_list处。	SELECT host, count(*) AS pv WHERE visitCount > 0 GROUP BY host
HAVING	只能与GROUP BY配合使用。指定用于过滤GROUP BY结果的结构化字段。	SELECT host, count(*) AS pv GROUP BY host HAVING pv > 10
ORDER BY	后面的字段必须是用于GROUP BY分组的字段，对GROUP BY的查询结果进行排序，用于排序的可以是任意一个结构化字段。	SELECT host, count(*) AS pv GROUP BY host ORDER BY pv
ASC/DESC	ASC为升序，DESC为降序，默认为ASC。	SELECT host, count(*) AS pv GROUP BY host ORDER BY pv DESC
LIMIT	对查询结果进行限制，用于限制返回的结构化日志条数。一次查询最多返回100000条结构化日志。 如果不使用LIMIT语句，默认返回查询结果中最新的100条数据。	SELECT host LIMIT 100

示例

表 6-24 常用 SQL 查询语句示例

查询需求	查询语句
标准查询	SELECT "field" WHERE "field" = 'value'
统计行数	SELECT count(*)
列的别名	SELECT count(*) AS "pv"
去重查询	SELECT DISTINCT("field")
分页查询	SELECT "field" LIMIT 100
排序查询	SELECT "__time" ORDER BY "__time"
分组查询	SELECT "field" GROUP BY "field"

查询需求	查询语句
分组统计	SELECT "field",count(*) GROUP BY "field"
模糊查询	SELECT "field" LIKE 'value%'
查询总和	SELECT sum("field")
查询最大值	SELECT max("field")
查询最小值	SELECT min("field")
查询平均值	SELECT avg("field")
SQL嵌套子查询	SELECT sum(pv) FROM (SELECT "field",count(*) AS "pv" GROUP BY "field")
HAVING子句过滤	SELECT "field",count(*) AS "pv" GROUP BY "field" HAVING "pv" > 10
查询包含GET，POST请求	SELECT * WHERE "request_method" IN ('GET', 'POST')
查询不包含GET，POST请求	SELECT * WHERE "request_method" NOT IN ('GET', 'POST')
查询非GET请求的日志	SELECT * WHERE "request_method" != 'GET'
查询GET请求成功并且状态码为200且请求时间小于60秒的日志	SELECT * WHERE "request_method" = 'GET' AND "request_time" < 60
查询请求时间大于等于60秒，并且小于200秒的日志	SELECT * WHERE "request_time" >=60 and "request_time" < 200
查询GET请求或POST请求的日志	SELECT * WHERE "request_method" = 'GET' OR "request_method" = 'POST'

下面的语句是根据ELB结构化日志构造出的查询语句，它包含所有的基础查询语法，仅供参考。

```
SELECT url AS Url, host AS Host, failure_rate AS FailureRate,
CONCAT(CAST(access_count AS varchar), ' times') AS "All",
CONCAT(CAST(rsp_200_count AS varchar), ' times') AS "COUNT_200"
FROM ( SELECT
CONCAT(host, CASE WHEN STRPOS(router_request_uri, '?') = 0 THEN router_request_uri ELSE
SUBSTR(router_request_uri, 1, 1) END) AS url,
host,count(1) AS access_count,
SUM(CASE WHEN status = 200 THEN 1 ELSE 0 END) AS "rsp_200_count",
(CASE WHEN COUNT(1) < 30 THEN 0 ELSE round(SUM(CASE WHEN status >= 400 THEN 1 ELSE 0 END) *
100.0 / COUNT(1), 2) END) AS failure_rate
WHERE host NOT IN ('monitor-new.olayc.cn')
GROUP BY host,router_request_uri
HAVING router_request_uri NOT IN ('/robots.txt', '/null', '/undefined')
)
```



```
ORDER BY FailureRate DESC
LIMIT 100
```

6.11.2 SQL 聚合函数

聚合函数是对结构化后的日志的指定列进行的统计运算。聚合函数返回的是单个值，经常与SELECT语句和GROUP BY语句一起使用。LTS支持如下表格所示聚合函数，具体请参考表6-25。

在聚合函数的使用中请注意以下几点：

- 聚合函数可用在任何查询的SELECT子句中。您可以使用如AGG(expr) FILTER(WHERE whereExpr)的语法在聚合之前进行过滤，即聚合函数只会聚合满足过滤条件的列。
- 在同一个SQL查询语句中，根据过滤条件的不同，对应的聚合函数所呈现的结果会不同。
- 只有COUNT函数可以跟DISTINCT搭配使用。
- 聚合操作没有固定的执行顺序。如果在执行具有多个聚合函数的SQL语句查询时，执行聚合函数的顺序对运算结果有影响，即结果会因执行顺序的不同而不同，那么每次执行这个查询，得出的结果可能会不一致。
- 如果需要聚合的数据为FLOAT类型时，数次执行同一个查询可能也会因此得出不同的聚合结果。如果您希望执行同一个查询时都能得出同样的结果，建议使用ROUND函数来消除多次查询之间的不一致。

语法格式

```
SELECT COUNT(fieldname1)
```

聚合函数语句

表 6-25 聚合函数语句

语句	说明	示例
COUNT(*)	统计行数。	SELECT COUNT(*)
COUNT(DISTINCT expr)	统计字段中去重后的行数，字段值可以是字符串或者数字，返回值为估算值（默认存在2.3%的标准误差）。	SELECT COUNT(DISTINCT host)
SUM(expr)	返回数字总和。	SELECT SUM(visitCount)
MIN(expr)	返回数字中的最小值。	SELECT MIN(visitCount)
MAX(expr)	返回数字中的最大值。	SELECT MAX(visitCount)
AVG(expr)	返回平均值。	SELECT AVG(visitCount)
EARLIEST(expr)	表达式必须是数值类型的，返回expr的最早的值，即查询的时候最先遇到的值。	SELECT EARLIEST(visitCount)

语句	说明	示例
LATEST(expr)	表达式必须是数值类型的，返回expr的最新的值，即查询的时候最后遇到的值。	SELECT LATEST(visitCount)
APPROX_QUANTILE_DS(expr, probability)	计算数值expr的近似分位数，probability应介于0和1之间。	APPROX_QUANTILE_DS(expr, probability)

6.11.3 SQL 同比和环比函数

本文介绍同比和环比函数的基础语法和示例。

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。

语法格式

- 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。
compare(x,n)
- 对比当前时间周期内的计算结果与n1、n2、n3秒之前时间周期内的计算结果。
compare(x, n1, n2, n3...)

参数说明

表 6-26 同比函数参数说明

参数	说明
x	目标列的列名，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）。

返回类型

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]。

示例说明

计算当前1小时和昨天同时段的访问量比值。

- 选择查询和分析的时间范围为**1小时（整点时间）**，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 )
```
- 查询和分析结果。
 - 5994.0**表示当前1小时（例如2021-01-02 00:00:00~2021-01-02 01:00:00）的网站访问量。

- **6000.0**表示昨天同时段（例如2021-01-01 00:00:00~2021-01-01 01:00:00）的网站访问量。
- **0.999**表示当前1小时与昨天同时段的网站访问量比值。

图 6-30 查询和分析结果

```
EXPRESSO  
[5994.0,6000.0,0.999]
```

3. 分列显示查询和分析结果

```
SELECT  
diff[1] as "today",  
diff[2] as "yesterday",  
diff[3] as "ratio"  
FROM(SELECT compare(pv, 86400) AS diff FROM (SELECT count(*) AS pv ))
```

图 6-31 分页查询和分析结果

today	yesterday	ratio
5993	6029	0.99402887

ts_compare 函数

ts_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。

ts_compare函数必须按照时间列进行分组（GROUP BY）。

语法格式

- 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。
ts_compare(x, n)
- 对比当前时间周期内的计算结果与n1、n2、n3秒之前时间周期内的计算结果。
ts_compare(x, n1, n2, n3...)

参数说明

表 6-27 环比函数参数说明

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）。

返回类型

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, n秒前的UNIX时间戳]。

示例说明

环比今天3小时与昨天3小时的网站访问量。

选择查询和分析的时间范围为今天某3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），date_trunc('hour',__time)表示使用date_trunc函数将时间对齐到小时。

- 查询和分析语句

```
SELECT
  t_time,
  ts_compare(PV, 86400) AS data
FROM(
  SELECT
    date_trunc('hour', __time) AS t_time,
    count(*) AS PV
  GROUP BY
    t_time
  ORDER BY
    t_time
)
GROUP BY
  t_time
```
- 查询和分析结果

t_time	data
2021-10-26T06:00:00.000Z	[159.0,224.0,0.7098214285714286,1.6351416E9]
2021-10-26T07:00:00.000Z	[100.0,148.0,0.6756756756756757,1.6351452E9]
2021-10-26T08:00:00.000Z	[100.0,100.0,1.0, 1.6016544E9, 1.6351488E9]

6.11.4 SQL JSON 函数

功能描述

JSON函数用于解析JSON对象或JSON数组，并从中提取值。

语法格式

```
SELECT json_extract(Results, '$[0].EndTime')
```

JSON 函数语句

表 6-28 JSON 函数语句

语句	说明	示例	返回值类型
json_extrac t	用于从JSON对象或JSON数组中提取一组JSON值（数组或对象）。	json_extract(x, json_path)	JSON格式的string类型
json_extrac t_scalar	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）。如果指定JSON路径下不是标量，则返回null。	json_extract_scal ar(x,json_path)	varchar类型

示例及说明

- json_extract函数**
获取Results字段中EndTime字段的值。
 - 字段样例
Results:[{"EndTime":1626314520},{"FireResult":2}]
 - 查询和分析语句
SELECT json_extract(Results, '\$[0].EndTime')
 - 查询和分析结果

表 6-29 查询和分析结果

EXPR\$0
1626314520

- json_extract_scalar函数**
从Results字段中获取RawResultCount字段的值，并将这些值转换为bigint类型进行求和。
 - 字段样例
Results:[{"EndTime":1626314520},{"RawResultCount":1}]
 - 查询和分析语句
SELECT sum(cast(json_extract_scalar(Results,'\$[1].RawResultCount') AS bigint))
 - 查询和分析结果

表 6-30 查询和分析结果

EXPR\$0
1546

6.11.5 SQL IP 函数

使用限制

LTS提供的IP与地域之间的关系，来自于第三方IP库，且数据是周期性更新（约半年），不承诺IP与地域关系完全正确；后续LTS会优化，缩短IP库的更新周期，为用户提供更好的体验。

单IP函数聚合查询的数据量上限为500万，查询的数据超出上限可能导致查询超时。

功能描述

IP函数是分析目标IP地址所属的国家、省份、城市及对应的网络运营商。

语法规式

```
SELECT count(*) AS PV, ip_to_province(client_ip) AS province GROUP BY province
```

IP 函数语句

表 6-31 IP 函数语句

语句	说明	示例
ip_to_province	分析目标IP地址所属省份。	ip_to_province(x)
ip_to_country	分析目标IP地址所属国家或地区。	ip_to_country(x)
ip_to_city	分析目标IP地址所属城市。	ip_to_city(x)
ip_to_provider	分析目标IP地址所对应的网络运营商。	ip_to_provider(x)
ip_to_geo	根据传入的ip返回ip所在的经纬度。	ip_to_geo(x)

示例及说明

- ip_to_province函数**

统计请求总数Top3的省份。

a. 查询和分析语句

SELECT count(*) AS PV, ip_to_province(client_ip) AS province GROUP BY province ORDER BY PV desc LIMIT 3

b. 查询和分析结果

表 6-32 查询和分析结果

PV	province
101	广东
83	上海
78	山东

- ip_to_country函数**

统计请求总数的Top3的国家或地区。

a. 查询和分析语句

SELECT count(*) AS PV, ip_to_country(client_ip) AS country GROUP BY country ORDER BY PV desc LIMIT 3

b. 查询和分析结果

表 6-33 查询和分析结果

PV	country
100	中国

PV	country
76	美国
55	加拿大

- **ip_to_city函数**

统计请求总数的Top3的城市。

- a. 查询和分析语句

```
SELECT count(*) AS PV, ip_to_city(client_ip) AS city GROUP BY city ORDER BY PV desc LIMIT 3
```

- b. 查询和分析结果

表 6-34 查询和分析结果

PV	city
109	广州
89	上海
23	西安

- **ip_to_provider函数**

统计请求总数的Top3的运营商。

- a. 查询和分析语句

```
SELECT count(*) AS PV, ip_to_provider(client_ip) AS provider GROUP BY provider ORDER BY PV desc LIMIT 3
```

- b. 查询和分析结果

表 6-35 查询和分析结果

PV	provider
115	电信
65	att.com
44	联通

- **ip_to_geo函数**

根据传入的ip返回经纬度。

- a. 查询和分析语句

```
SELECT count(*) AS PV, ip_to_geo (client_ip) AS geo GROUP BY geo ORDER BY PV desc LIMIT 3
```

- b. 查询和分析结果

表 6-36 查询和分析结果

PV	geo
101	*, *

PV	geo
83	47.369013, -68.326674
78	32.715891, -117.161588

6.11.6 SQL 数学函数

功能描述

数学函数为标量函数中的一种，只支持数值类型的字段，能够实现对数值进行取整、取绝对值、求余等功能，具体请参考[表6-37](#)。

在数学运算中，如果表达式里涉及的操作数皆为整数，那么SQL将会采用整数运算，否则便会切换到浮点运算。您可以将其中一个操作数转换为FLOAT类型来强制进行切换，运行时SQL会将大多数表达式中的32位浮点数扩展到64位。

语法格式

```
SELECT ABS(fieldname1) AS fieldname1_abs
```

数学函数语句

表 6-37 数学函数语句

语句	说明	示例
ABS(expr)	取绝对值。	SELECT ABS(fieldname1)
CEIL(expr)	向上取整，即向上取最接近的整数值	SELECT CEIL(fieldname1)
FLOOR(expr)	向下取整，即向下取最接近的整数值。	SELECT FLOOR(fieldname1)
TRUNCATE(expr, digits)	将expr截断为特定的digits位数。如果数字为负数，则会截断小数点左侧的许多位置。如果未指定，数字默认为零。	SELECT TRUNCATE(fieldname1, 2)

语句	说明	示例
ROUND(expr, digits)	ROUND(expr, digits)对expr值进行四舍五入，保留小数位数由digits指定。expr可以是整数或浮点数，但digits必须是整数。返回值的类型由expr的类型决定。如果没有指定digits，则使用默认值0。如果digits是负数，则返回expr四舍五入后的整数。当expr是非数字值时，会被转换为数字0。如果expr是无限位数的数字，则被转换为最接近的DOUBLE类型的有限位数的数字。	SELECT ROUND(fieldname1, 2)
x + y	加法。	SELECT fieldname1 + fieldname2
x - y	减法。	SELECT fieldname1 - fieldname2
x * y	乘法。	SELECT fieldname1 * fieldname2
x / y	除法。	SELECT fieldname1 / fieldname2
MOD(x, y)	求余，即取x除以y后的余数。	SELECT MOD(fieldname1, fieldname2)
LN(expr)	对数(以e为底)。	SELECT ln(expr)
LOG10(expr)	对数(以10为底)。	SELECT LOG10(expr)
POWER(expr,power)	expr的power次幂。	SELECT POWER(expr ,2)
SQRT(expr)	expr的平方根	SELECT SQRT(expr)
SIN(expr)	正弦	SELECT SIN(expr)
COS(expr)	余弦	SELECT COS(expr)
TAN(expr)	正切	SELECT TAN(expr)
COT(expr)	余切	SELECT COT(expr)
ASIN(expr)	反正弦	SELECT ASIN(expr)
ACOS(expr)	反余弦	SELECT ACOS(expr)
ATAN(expr)	反正切	SELECT ATAN(expr)

示例及说明

ACOS(expr)函数

求参数值的反余弦， $y=\arccos x$ ， x 的取值范围 $[-1,1]$ 。

1. 字段样例
- x:0.5
2. 查询和分析语句
- select ACOS(x)
3. 查询和分析结果

表 6-38 查询和分析结果

x	EXPR\$1
0.5	1.0471975511965979

ATAN(expr)函数

ATAN求参数值的反正切， $y= \arctan x$ ， x 的取值范围 R 。

1. 字段样例
- x:0.5
2. 查询和分析语句
- select ATAN(X)
3. 查询和分析结果

表 6-39 查询和分析结果

x	EXPR\$1
0.5	1.0471975511965979

ATAN2(expr)函数

ATAN2从直角坐标（ x, y ）到极坐标（ r, θ ）的转换角度 θ 。

1. 字段样例
- x:3; y:4
2. 查询和分析语句
- SELECT x , y, ATAN2(x,y)
3. 查询和分析结果

表 6-40 查询和分析结果

x	y	EXPR\$0
3	4	0.6435011087932844

6.11.7 SQL 时间函数

功能描述

时间函数可以与__time一起使用，任何存储为毫秒时间戳的列都可以使用MILLIS_TO_TIMESTAMP 函数，或者任何存储为字符串时间戳的列都可以使用TIME_PARSE 函数。默认情况下，时间操作使用UTC时区，您可以通过参数"timezone"设置为另一个时区的名称（如"Asia/Shanghai"）或设置为偏移量（如"+08:00"）来更改时区。

语法格式

语句	说明	语法格式
CURRENT_DATE	在连接时区的当期日期。	SELECT CURRENT_DATE
CURRENT_TIMESTAMP	在连接时区的当前时间戳。	SELECT CURRENT_TIMESTAMP
DATE_TRUNC(<unit>,<expr>)	截断时间戳，将其作为新时间戳返回。	SELECT DATE_TRUNC('minute',__time)
TIME_FORMAT(<expr>,<pattern>,<timezone>)	使用给定的 pattern 或 ISO8601（例如 2000-01-02T03:04:05Z）将字符串解析为时间戳。timezone（如果提供）应为时区名称，如 "America/Los_Angeles"或偏移量，如"+08:00"，并将用作不包括时区偏移量的字符串的时区。模式和时区必须是字面量。无法解析为时间戳的字符串将返回空值。	SELECT TIME_FORMAT(__time,'yy-MM-dd HH:mm:ss','+08:00')

语句	说明	语法格式
TIME_PARSE(<expr>,<pattern>,<timezone>)	使用给定的 pattern 或 ISO8601（例如 2000-01-02T03:04:05Z）将字符串解析为时间戳。时区（如果提供）应为时区名称，如 "America/Los_Angeles" 或偏移量，如 "+08:00"，并将用作不包括时区偏移量的字符串的时区。模式和时区必须是字面量。无法解析为时间戳的字符串将返回空值。	SELECT TIME_PARSE("timestamp", 'yyyy-MM-dd HH:mm:ss', '+08:00')
MILLIS_TO_TIMESTAMP(expr)	将时间戳转换为时间格式。	SELECT MILLIS_TO_TIMESTAMP(expr)
TIMESTAMP_TO_MILLIS(expr)	将时间转换为时间戳格式	SELECT TIMESTAMP_TO_MILLIS(expr)
EXTRACT(<extract_unit> FROM expr)	从 expr 中提取时间部分，并将其作为数字返回。	SELECT EXTRACT(MINUTE FROM __time)
TIMESTAMPDIFF(<unit>,<expr1>,<expr2>)	返回 expr1 和 expr2 之间的 unit	SELECT TIMESTAMPDIFF(minute, expr1, expr2)
TIME_SERIES	补全您查询时间窗口内缺失的数据。	TIME_SERIES(__time, period, time_format, [padding_value], <timezone>)
TIME_FLOOR	向下舍入时间戳，将其作为新时间戳返回。期间可以是任何 ISO8601 期间，例如 P3M（季度）或 PT12H（半天）。如果提供时区，则应为时区名称（如 "America/Los_Angeles"）或偏移量（如 "-08:00"）。	TIME_FLOOR(<timestamp_expr>, <period>, [<origin>, [<timezone>]])

TIME_SERIES 函数

time_series 函数用于补全您查询时间窗口内缺失的数据。

- 必须搭配ORDER BY语法使用，并且time_series函数是ORDER BY的第一个参数
- 查询语句中不能使用OFFSET语句
- time_series函数不支持作为子查询使用

语法格式

```
time_series(__time, period, time_format, [padding_value], <timezone>)
```

参数说明

表 6-41

参数	说明
__time	时间序列。
period	ISO 8601标准的时间窗口大小。例如P1M（1月）、P1D（1天）、PT1H（1小时）、PT1M（1分钟）、PT1S（1秒钟）。
time_format	返回结果的时间格式。（参考 Joda DateTimeFormat 模式）。
padding_value	补全的内容。包括： ● 0 或 zero：将缺失的值设置为0（默认值）。 ● null：将缺失的值设置为null。 ● last：将缺失的值设置了上一个时间点对应的值。 ● next：将缺失的值设置了下一个时间点对应的值。 ● avg：将缺失的值设置为前后两个时间点的平均值。
timezone	时区。例如：北京时区：+08:00

返回类型

bigint类型。

示例说明

按照一天的时间粒度进行数据补全，将缺失的值设置为0，并添加时区。

- 查询和分析语句

```
select time_series(__time, 'P1D', 'yyyy-MM-dd HH:mm:ss', '0', '+08:00') as t_time, count(*) as num
group by t_time order by t_time
```

- 查询和分析结果

t_time	num
2021-10-01 08:00:00	5
2021-10-02 08:00:00	0
2021-10-03 08:00:00	0

2021-10-04 08:00:00	21
2021-10-05 08:00:00	17
2021-10-06 08:00:00	0
2021-10-07 08:00:00	34

CURRENT_DATE/ CURRENT_TIMESTAMP 函数

CURRENT_DATE返回查询当天的凌晨零点的ISO8601时间，返回的为UTC时间，该函数可直接参与时间戳之间的运算。

CURRENT_TIMESTAMP返回查询当前的ISO8601时间，返回的为UTC时间，该函数可直接参与时间戳之间的运算。

1. 字段样例
- `__time: 2023-02-14T02:35:56.706Z`
2. 查询和分析语句
- `select __time,CURRENT_DATE, CURRENT_TIMESTAMP,CURRENT_TIMESTAMP`
3. 查询和分析结果

表 6-42 查询和分析结果

<code>__time</code>	<code>CURRENT_DATE</code>	<code>CURRENT_TIMESTAMP</code>
2023-02-14T02:35:56.706Z	2023-02-14T00:00:00.000Z	2023-02-14T14:35:57.000Z

DATE_TRUNC(<unit>, <timestamp_expr>)函数

舍去时间戳<timestamp_expr>中精度大于所选单位<unit>的值，将其置为零，并作为新时间戳返回。单位可以是'milliseconds'（毫秒），'second'（秒），'minute'（分），'hour'（时），'day'（日），'week'（周），'month'（月），'quarter'（季），'year'（年），'decade'（十年），'century'（千年），'millennium'（世纪），unit不区分大小写。

1. 字段样例
- `__time: 2023-02-14T02:35:56.706Z`
2. 查询和分析语句
- `SELECT __time,DATE_TRUNC('minute', __time),DATE_TRUNC('day', __time),DATE_TRUNC('year', __time)`
3. 查询和分析结果

表 6-43 查询和分析结果

<code>__time</code>	<code>EXPR\$1</code>	<code>EXPR\$2</code>	<code>EXPR\$3</code>
2023-02-14T02:35:56.706Z	2023-02-15T08:50:00.000Z	2023-02-15T00:00:00.000Z	2023-01-01T00:00:00.000Z

TIME_PARSE(<string_expr>, [<pattern>, [<timezone>]])/ TIME_FORMAT(<timestamp_expr>, [<pattern>, [<timezone>]])函数

TIME_PARSE将给定的字符串<timestamp_expr>，依据用户自定义的<pattern>参数以 [Joda DateTimeFormat](#) 模式解析为时间戳。若<pattern>未填写则以默认的ISO8601将字符串解析。<timezone>为时区，可省略。

TIME_FORMAT将给定的时间戳< timestamp_expr>，依据用户自定义的<pattern>参数以 [Joda DateTimeFormat](#) 模式解析为字符串。若<pattern>未填写则以默认的ISO8601将时间戳解析。<timezone>为时区，可省略。

1. 字段样例
__time: 2023-02-16T07:38:25.306Z
start_time:2023-02-14 02:35:56
2. 查询和分析语句
SELECT __time,TIME_PARSE(start_time,'yyyy-MM-dd HH:mm:ss'),TIME_FORMAT(__time,'yyyy-MM-dd HH:mm:ss')
3. 查询和分析结果

表 6-44 查询和分析结果

__time	EXPR\$1	EXPR\$2
2023-02-16T07:38:25.306Z	2023-02-14T02:35:56.000Z	2023-02-16 07:38:25

MILLIS_TO_TIMESTAMP(millis_expr)/ TIMESTAMP_TO_MILLIS(timestamp_expr)函数

MILLIS_TO_TIMESTAMP函数将毫秒值转化为ISO8601格式的时间戳，转化后的参数可进行时间戳之间的运算。TIMESTAMP_TO_MILLIS将时间戳转化为毫秒值。

1. 字段样例
__time: 2023-02-16T07:54:15.106Z, start_time: 1676534055106
2. 查询和分析语句
SELECT __time,MILLIS_TO_TIMESTAMP(start_time),TIMESTAMP_TO_MILLIS(__time)
3. 查询和分析结果

表 6-45 查询和分析结果

__time	EXPR\$1	EXPR\$2
2023-02-16T07:54:15.106Z	2023-02-16T07:54:05.000Z	1676534055106

TIME_EXTRACT(<timestamp_expr>,[<unit>],[<timezone>]])/ EXTRACT(<unit> FROM timestamp_expr)函数

TIME_EXTRACT函数，从timestamp_expr中提取时间部分，并将其作为数字返回。单位可以是EPOCH（返回纪元以来的秒值unix）、SECOND（当前分钟的秒值）、

MINUTE（当前小时中的分钟值）、HOUR（当天的小时值）、DAY（当月的天数）、DOW（当前周的天数）、DOY（当前年的天数）、WEEK（当前年的周数）、MONTH（当前的月数）、QUARTER（当年的季度）或YEAR（返回当前年份）。<timezone>为时区，可省略。EXTRACT函数为TIME_EXTRACT的简写形式。

1. 字段样例
- `__time: 2023-02-16T07:54:15.106Z, start_time: 1676534055106`
2. 查询和分析语句
- `SELECT __time,MILLIS_TO_TIMESTAMP(start_time),TIMESTAMP_TO_MILLIS(__time)`
3. 查询和分析结果

表 6-46 查询和分析结果

<code>__time</code>	<code>EXPR\$1</code>	<code>EXPR\$2</code>
2023-02-16T07:54:15.106Z	2023-02-16T07:54:05.000Z	1676534055106

TIME_FLOOR(<timestamp_expr>, <period>, [<origin>, [<timezone>]])函数

TIME_FLOOR函数是向下舍入时间戳，将其作为新时间戳返回。期间可以是任何ISO8601期间，例如P3M（季度）或 PT12H（半天）。如果提供时区，则应为时区名称（如“America/Los_Angeles”）或偏移量（如“-08: 00”）。

1. 字段样例
- `__time 2025-03-05T16:53:21.689+08:00`
2. 查询和分析语句
- `SELECT __time,TIME_FLOOR(__time,'PT1M') AS t`
3. 查询和分析结果

表 6-47 查询和分析结果

<code>__time</code>	<code>t</code>
2025-03-05T16:53:21.687+08:00	2025-03-05T16:53:00.000+08:00

参考信息

• unit说明

unit	说明
second	秒
minute	分
hour	时
day	日
week	周

month	月
quarter	季
year	年

• **extract_unit说明**

extract_unit	说明
SECOND	秒
MINUTE	分
HOURL	时
DAY	每月的第几天
DOW	每周的第几天
DOY	每年的第几天
WEEK	每年的第几周
MONTH	月
QUARTER	季
YEAR	年

6.11.8 SQL 最值函数

功能描述

SQL提供最值函数，对字段进行最值求解，具体请参见[表6-48](#)

最值函数对零个或多个字段进行操作，并返回单个值。

在最值函数的使用中请注意以下几点：

- 如果没有设置字段，则返回空值。字段必须能够转换为常见的数据类型。
- 如果所有字段都为空值，则返回空值。如果只有部分字段为空值，这些字段会被忽略。
- 如果字段中既有数字也有字符串，则函数将它们作为字符串进行比较。
- 如果所有字段都是整数，则函数将它们作为LONG值进行比较。
- 如果所有字段都是数字且至少有一个是FLOAT值，则函数将它们作为FLOAT值进行比较。

语法格式

```
SELECT GREATEST(fieldname1,fieldname2) AS the_greatest_field
```

最值函数语句

表 6-48 最值函数语句

语句	说明	示例
GREATEST([expr1, ...])	返回零个或多个字段间的最大值。	SELECT GREATEST(fieldname1,fi eldname2)
LEAST([expr1, ...])	返回零个或多个字段间的最小值。	SELECT LEAST(fieldname1,fieldn ame2)

6.11.9 SQL 字符串函数

功能描述

SQL提供字符串函数，用于对字符类型的数据执行拼接、大小写转换等操作，具体请参见[表6-49](#)。

SQL语法中，字符必须被单引号（'）包裹，无符号或双引号（"）包裹的为字段或表名称，如：'msg'表示字符串msg，msg或"msg"表示日志结构化msg字段。

语法格式

```
SELECT (fieldname1 || fieldname2) AS fieldname1_fieldname2
```

字符串函数语句

表 6-49 字符串函数语句

语句	说明	示例
CONCAT(expr1, expr2...)	拼接列举的所有字符串。	SELECT str1, str2, str3, CONCAT(str1, str2, str3) WHERE str1 IS NOT NULL
TEXTCAT(expr, expr)	拼接两个字符串。	SELECT str1, str2, TEXTCAT(str1, str2) WHERE str1 IS NOT NULL
STRING_FORMAT(pattern[, args...])	根据JAVA的String格式对字符串进行格式化。	SELECT str1, STRING_FORMAT(str1, '%s') WHERE str1 IS NOT NULL
LENGTH(expr)	返回字符串的长度，即字符串中UTF-16字符个数。	SELECT LENGTH(str1) WHERE str1 IS NOT NULL

语句	说明	示例
LOWER(expr)	将字符串转换为小写形式。	SELECT LOWER(str1) WHERE str1 IS NOT NULL
POSITION(string1 IN string2 [FROM fromIndex])	返回string1在string2中首次出现位置的索引。搜索从指定索引开始，如果没有指定索引，则从索引1开始。如果string1不存在于string2中，则返回0。	SELECT POSITION(str1 IN str2 FROM 5)
REGEXP_EXTRACT(expr, pattern, [index])	返回字符串中匹配指定正则表达式的子字符串。索引从1开始。如果没有匹配，则返回空值。如果没有指定索引，或者索引为0，则返回第一个匹配的子字符串。如想精确匹配，请在正则表达式前后分别加上符号^和\$。	SELECT REGEXP_EXTRACT(str1, '[A-Za-z]+://[A-Za-z0-9.-]+(/[^\]*)', 5)
REGEXP_LIKE(expr, pattern)	判断字符串是否匹配指定的正则表达式。如想精确匹配，请在正则表达式前后分别加上符号^和\$。该函数与LIKE语句用法类似，区别在于LIKE语句搜索的是匹配指定模式的内容。	SELECT REGEXP_LIKE(str1, '\.(jpg jpeg png gif)\$')
REPLACE(expr, pattern, replacement)	使用replacement替换expr中与pattern相同的子字符串。	SELECT REPLACE(expr,pattern, replacement)
STRPOS(string1, string2)	返回string2在string1中首次出现位置的索引。查找从索引1开始。如果查找没有结果，则返回0。	SELECT STRPOS(str1, str2) WHERE str1 IS NOT NULL AND str2 IS NOT NULL
SUBSTRING(expr, index, [length])	截取字符串。从指定索引处开始截取，结束位置由指定长度决定。长度按照UTF-16字符个数计算。	SELECT SUBSTRING(str1, 3, 10) WHERE str1 IS NOT NULL
RIGHT(expr, [length])	从字符串最右处开始往左截取指定长度。	SELECT RIGHT(str1, 5) WHERE str1 IS NOT NULL
LEFT(expr, [length])	从字符串最左处开始往右截取指定长度。	SELECT LEFT(str1, 5) WHERE str1 IS NOT NULL
SUBSTR(expr, index, [length])	与SUBSTRING相同。	SELECT SUBSTR(str1, 3, 10) WHERE str1 IS NOT NULL

语句	说明	示例
UPPER(expr)	将字符串转换为大写形式。	SELECT UPPER(str1) WHERE str1 IS NOT NULL
REVERSE(expr)	反转字符串。	SELECT REVERSE(str1) WHERE str1 IS NOT NULL
LPAD(expr, length, chars)	在字符串左侧填充指定字符，直至字符串达到指定长度。如果指定长度小于字符串本身的长度，则按照指定长度对字符串执行截断操作。如果字符串或指定字符为空值，则返回空值。如果指定字符为空白，不会执行填充操作，但如有必要可能会删减字符。	SELECT LPAD(str1, 50, 'testStr') WHERE str1 IS NOT NULL
RPAD(expr, length, chars)	在字符串右侧填充指定字符，直至字符串达到指定长度。如果指定长度小于字符串本身的长度，则按照指定长度对字符串执行截断操作。如果字符串或指定字符为空值，则返回空值。如果指定字符为空白，不会执行填充操作，但如有必要可能会删减字符。	SELECT RPAD(str1, 50, 'testStr') WHERE str1 IS NOT NULL
CONTAINS_STRING(<expr>, str)	判断expr是否包含str字符串	SELECT CONTAINS_STRING(log_level, 'warn')
ICONTAINS_STRING(<expr>, str)	判断expr是否包含str字符串，不区分字符串大小写	SELECT ICONTAINS_STRING(log_level, 'WARN')

示例及说明

REPEAT函数

REPEAT(expr, [N])函数将expr重复N次。

1. 字段样例

field4:is

2. 查询和分析语句

```
select field4, REPEAT(field4, 3)
```

3. 查询和分析结果

表 6-50 查询和分析结果

field4	EXPR\$1
Is	isis

6.11.10 SQL SPLIT 函数

功能描述

SPLIT函数用于通过指定的分隔符拆分字符串，并返回拆分后的子串集合。

语法格式

```
SELECT split_to_map(x, delimiter01, delimiter02)
```

SPLIT 函数语句

语句	说明	示例	参数
split	split函数用于通过指定的分隔符拆分字符串，并返回拆分后的子串集合。	split(x, delimiter,[limit])	- x：参数值为varchar类型。 - delimiter：分隔符。 - limit：限制字符串拆分的个数，大于0的整数。
split_part	split_part函数通过指定的分隔符拆分字符串，并返回指定索引的内容。	split_part(x, delimiter, part)	- x：参数值为varchar类型。 - delimiter：分隔符。 - part：指定要返回字段的索引值。
split_to_map	split_to_map函数用于使用指定的第一个分隔符拆分字符串，然后再使用指定的第二个分隔符进行第二次拆分。	split_to_map(x, delimiter01, delimiter02)	- x：参数值为varchar类型。 - delimiter01：分隔符1。 - delimiter02：分隔符2。

示例及说明

- split函数**
将目标字符串按指定字符串分割，limit用于限制分割后的最大单词数，若不填写，则默认全部分割。

- a. 字段样例
Id: dc1dab7e-b045-4e77-bda4-914d083d1bf7
- b. 查询和分析语句
SELECT split(Id,'-'), split(Id,'-',2)
- c. 查询和分析结果

表 6-51 split 函数查询和分析结果

EXPR\$0	EXPR\$1
["dc1dab7e","b045","4e77","bda4-914d083d1bf7"]	["dc1dab7e","b045-4e77-bda4-914d083d1bf7"]

- **split_part函数**
通过指定的分隔符拆分字符串，并返回指定索引的内容字段样例，索引下标从0开始。若索引下标超过分割数量或者为负数，则返回空字符串。
 - a. 字段样例
Id: dc1dab7e-b045-4e77-bda4-914d083d1bf7
 - b. 查询和分析语句
SELECT split_part(Id,'-',1)
 - c. 查询和分析结果

表 6-52 split_part 函数查询和分析结果

EXPR\$0
b045

- **split_to_map函数**
用于使用指定的第一个分隔符拆分字符串，然后再使用指定的第二个分隔符进行第二次拆分，展示形式为{ “KEY1” :” VALUE1 ” ,” KEY2 ” :” VALUE2 ” }。无法被二次分割的value值为空。
 - a. 字段样例
Request:request_id:"e3ac4b70c7d244f080d434e300d8065a" ;request_time: "1674965051000"
 - b. 查询和分析语句
SELECT split_to_map(Request,';',';')
 - c. 查询和分析结果

表 6-53 split_to_map 函数查询和分析结果

EXPR\$0
{"request_id ":"e3ac4b70c7d244f080d434e300d8065a", "request_time":"1674965051000"}

6.11.11 SQL 比较运算符

功能描述

比较运算符用于比较两个值，并返回真(true)或假(false)。比较运算符可以对数值类型进行大小比较，对STRING类型进行包含比较，比如数值类型的字段num1 < num2是否为真，STRING类型的str1是否存在于字符串strs中等，具体请参见[表6-54](#)。

语法格式

```
SELECT fieldname1 WHERE fieldname1 > fieldname2
```

比较运算符语句

表 6-54 比较运算符语句

语句	说明	示例
x = y	等于。	SELECT num1 = num2
x <> y	不等于。	SELECT num1 <> num2
x > y	大于。	SELECT num1 > num2
x >= y	大于或等于。	SELECT num1 >= num2
x < y	小于。	SELECT num1 < num2
x <= y	小于或等于。	SELECT num1 <= num2
x BETWEEN y AND z	等同于x >= y AND x <= z。	SELECT num1 BETWEEN num2 AND num3
x NOT BETWEEN y AND z	等同于x < y OR x > z。	SELECT num1 NOT BETWEEN num2 AND num3
x LIKE pattern	如果x匹配SQL LIKE模式，则返回true。	SELECT str1 LIKE '*'
x NOT LIKE pattern	如果x不匹配SQL LIKE模式，则返回true。	SELECT str1 NOT LIKE '*'
x IS NULL	如果x是空值或空白字符串，则返回true。	SELECT str1 IS NULL
x IS NOT NULL	如果x既不是空值也不是空白字符串，则返回true。	SELECT str1 IS NOT NULL

语句	说明	示例
x IN (values)	如果x为其中一个列举值，则返回true。	SELECT str1 IN ('testStr1', 'testStr2')
x NOT IN (values)	如果x不在列举值中，则返回true。	SELECT str1 NOT IN ('testStr1', 'testStr2')
x IN (subquery)	如果x是通过指定子查询返回，则返回true。	SELECT str1 WHERE str2 IN (SELECT DISTINCT str2 LIMIT 100)
x NOT IN (subquery)	如果x不是通过指定子查询返回，则返回True。	SELECT str1 NOT IN (SELECT str2 LIMIT 100)

6.11.12 SQL IP 地址函数

功能函数

对于IPv4地址函数，地址参数可以是IPv4点分十进制字符串（例如"192.168.0.1"）或表示为整数的IP地址（例如3232235521）。subnet 参数应该是一个字符串，格式为CIDR表示法中的IPv4地址子网（例如"192.168.0.0/16"）。

IP 函数语句

语句	说明	示例
IPV4_MATCH(address,subnet)	如果subnet属于address的子网地址则返回true，否则返回false。如果 address不是有效的IPv4地址，则返回false。如果 address是整数而不是字符串，则此函数更高效。	SELECT IPV4_MATCH (address,subnet)
IPV4_PARSE(address)	将address解析为整数的IPv4地址。如果address是有效的IPv4地址，则它可以被解析。如果address不是有效的IPv4地址，则返回null。	SELECT IPV4_PARSE(address)

语句	说明	示例
IPV4_STRINGIFY(address)	将整数address转换为以点分隔的IPv4地址字符串。如果address是有效的IPv4地址的整数，则它可以被解析。如果address不能表示为IPv4地址，则返回null。	SELECT IPV4_STRINGIFY(address)

示例及说明

IPV4_MATCH(address, subnet)函数

IPV4_MATCH函数，如果address属于subnet的子网ip，则返回true，否则返回false。如果address不是有效的IPv4地址，则返回false。如果address 是整数而不是字符串，则此函数具有更高的执行效率。

1. 字段样例
- Ipv4: 192.168.1.18
2. 查询和分析语句
- select IPV4,IPV4_MATCH(Ipv4, '192.168.0.0/16')
3. 查询和分析结果

表 6-55 查询和分析结果

IPV4	EXPR\$1
192.168.1.18	true

IPV4_PARSE(address)/ IPV4_STRINGIFY(address)函数

将address解析为整数的IPv4地址。如果address是有效的IPv4地址，则它可以被解析。如果address不是有效的IPv4地址，则返回null。

1. 字段样例
- Ipv4: 192.168.0.1
- Num: 3232235521
2. 查询和分析语句
- select IPV4_PARSE(Ipv4), IPV4_STRINGIFY(Num)
3. 查询和分析结果

表 6-56 查询和分析结果

EXPR\$0	EXPR\$1
-1062731775	192.168.0.1

6.11.13 SQL 归约函数

功能描述

归约函数对零个或多个表达式进行操作，并返回单个表达式。如果没有表达式作为参数传递，则结果为 NULL。表达式必须全部转换为公共数据类型，即结果的类型有：

- 如果所有的参数都是 NULL，结果是 NULL，否则，NULL 参数被忽略。
- 如果所有的参数包含了数字和字符串的混合，参数都被解释为字符串。
- 如果所有的参数是整型数字，参数都被解释为长整型。
- 如果所有的参数是数值且至少一个参数是double，则参数都被解释为double。

语法格式

```
GREATEST([expr1, ...])/ LEAST([expr1, ...])
```

示例及说明

GREATEST([expr1, ...])/ LEAST([expr1, ...])函数

GREATEST函数，计算零个或多个表达式，并根据上述比较返回最大值。

LEAST函数，计算零个或多个表达式，并根据上述比较返回最小值。

1. 字段样例

Num: 11785730

2. 查询和分析语句

```
select Num,GREATEST( "Num"/10,(select count(1)) ),LEAST("Num"/10,(select count(1)))
```

3. 查询和分析结果

表 6-57 归约函数查询和分析结果

Num	EXPR\$1	EXPR\$2
11785730	1178573	1

6.11.14 SQL 其他函数

功能描述

SQL提供的函数，还支持一些转换类型和CASE WHEN等逻辑运算，具体请参见[表 6-58](#)。

语法格式

```
SELECT CAST(fieldname1 AS VARCHAR) AS fieldname1_str
```

其他函数语句

表 6-58 其他函数语句

关键字	说明	示例
CAST(value AS TYPE)	转换数据类型。只支持转换为VARCHAR、FLOAT。	SELECT fieldname1, CAST(fieldname1 AS VARCHAR)
CASE WHEN boolean_expr1 THEN result1 \[WHEN boolean_expr2 THEN result2 ... \] \[ELSE resultN \] END	简单CASE函数。	SELECT CASE WHEN httpStatus = 200 THEN 1 ELSE 0 END
NULLIF(value1, value2)	如果value1和value2相等，返回空值，否则返回value1。	SELECT fieldname1, fieldname2, NULLIF(fieldname1, fieldname2)
NVL(expr,expr-for-null)	如果"expr"为空值或空白字符串，则返回"expr-for-null"。	SELECT NVL(str1, 'expr-for-null')

6.11.15 SQL JOIN 语法

JOIN子句可以关联查询两个或多个表数据，本文介绍JOIN子句的基本使用方法。

语法

```
select key
from t1
LEFT|RIGHT|INNER JOIN t2
on t1.key=t2.key
```

当前日志服务支持LEFT JOIN、RIGHT JOIN和INNER JOIN三种JOIN子句方式。具体功能如下：

表 6-59

JOIN方式	说明
LEFT JOIN	以左表（t1）的结果为基础，关联右表（t2）数据。 当表名为纯数字时，需要给表名加上双引号转换成字符串。例如：表名是123，JOIN语句中该表应写成“123”。
RIGHT JOIN	以右表（t2）的结果为基础，关联左表（t1）数据。
INNER JOIN	两个表的结果（elb1，elb2）交集数据

示例

有两个表，access表示主机的接入指标包含路径，时延，状态码，host为主机指标包含cpu和内存。通过JOIN可以关联接入和主机指标，查看相同主机的不同维度的指标情况。

- **LEFT JOIN**

- a. 查询语句

```
SELECT
  "access".__time,
  "access".host_ip,
  "access".cost,
  "host".cpu,
  "host".memory
FROM
  log "access"
  LEFT JOIN (select memory,cpu,host_ip from log) host ON "access".host_ip = "host".host_ip
```

- b. 返回结果，总共60条数据。

- **RIGHT JOIN**

- a. 查询语句

```
SELECT
  "access".__time,
  "host".host_ip,
  "access".cost,
  "host".cpu,
  "host".memory
FROM
  log "access"
  RIGHT JOIN (select memory,cpu,host_ip from log) host ON "access".host_ip = "host".host_ip
```

- b. 返回结果，总共60条数据。

- **INNER JOIN**

- a. 查询语句

```
SELECT
  "access".__time,
  "host".host_ip,
  "access".cost,
  "host".cpu,
  "host".memory
FROM
  log "access"
  INNER JOIN (select memory,cpu,host_ip from log) host ON "access".host_ip = "host".host_ip
```

- b. 返回结果，总共45条数据。

6.11.16 SQL 查询样例

本章以ELB日志为例进行介绍，对LTS中的ELB原始日志进行查询，具体查询步骤如下。

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中，选择“日志管理”，单击目标日志组和日志流名称，进入日志详情页面。

步骤3 系统获取ELB原始日志，在日志搜索页面查看具体日志。

步骤4 单击右上角日志设置，在弹出页面中，选择“云端结构化解析”。

步骤5 选择结构化模板，根据ELB模板进行结构化配置。其他日志内容可选择其他结构化模板。

步骤6 在“可视化”页签下输入SQL查询语句对相应的字段进行查询，即可返回所需的日志内容。

----结束

查询结果呈现

表格

下面的语句查询请求的host，request_uri所对应的日志各有多少条，发送的请求体的大小（MB），请求返回的状态码分别是2xx, 3xx, 4xx, 5xx的占比，并按照日志条数降序排列。

```
SELECT "router_request_uri" as "request_uri", "host", COUNT(*) as pv,
round(sum(body_bytes_sent) / 1024.0 , 5) as "body_bytes_sent(MB)",
round(sum(case when status >= 200 and status < 300 then 1 else 0 end ) * 100.0 / COUNT(1), 6) as "2xx
ratio(%)",
round(sum(case when status >= 300 and status < 400 then 1 else 0 end ) * 100.0 / count(1), 6) as "3xx
ratio(%)",
round(sum(case when status >= 400 and status < 500 then 1 else 0 end ) * 100.0 / count(1), 6) as "4xx
ratio(%)",
round(sum(case when status >= 500 and status < 600 then 1 else 0 end ) * 100.0 / count(1), 6) as "5xx
ratio(%)",
GROUP BY "host", "router_request_uri"
ORDER BY pv DESC
LIMIT 100
```

柱状图

根据以下语句查询结果绘制柱状图，x轴选择"request_uri"，y轴选择"pv"，表示每种requestURL请求分别有多少条。

```
SELECT "router_request_uri" as "request_uri", "host", COUNT(*) as pv,
round(sum(body_bytes_sent) / 1024.0 , 5) as "body_bytes_sent(MB)",
round(sum(case when status >= 200 and status < 300 then 1 else 0 end ) * 100.0 / COUNT(1), 6) as "2xx
ratio(%)",
round(sum(case when status >= 300 and status < 400 then 1 else 0 end ) * 100.0 / count(1), 6) as "3xx
ratio(%)",
round(sum(case when status >= 400 and status < 500 then 1 else 0 end ) * 100.0 / count(1), 6) as "4xx
ratio(%)",
round(sum(case when status >= 500 and status < 600 then 1 else 0 end ) * 100.0 / count(1), 6) as "5xx
ratio(%)",
GROUP BY "host", "router_request_uri"
ORDER BY pv DESC
LIMIT 100
```

折线图

根据以下语句查询结果绘制折线图，x轴选择"_time_"，y轴选择"QPS"。表示查询时间段内间隔5s的QPS变化。

```
select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"TT"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') AS _time_, COUNT(*) as QPS from log group
by _time_
```

饼图

根据以下语句查询结果绘制饼图，类目选择"status"，数据选择"rm"，表示查询时间段内status不同值的占比。

```
SELECT status, COUNT(1) AS rm GROUP BY status
```

数字图

根据以下语句查询结果绘制数字图，查看最近一小时一共有多少次正常的请求。

```
SELECT count(*) AS normalRequest WHERE status = 200
```

7 日志可视化

7.1 日志可视化概述

云日志服务支持用户使用SQL查询语法对结构化日志进行统计分析，同时将分析结果以图表方式可视化展示，提供多种图表类型，满足不同场景的可视化呈现需求，用于运维和运营分析。

LTS提供40多种开箱即用仪表盘模板、预置样例数据，配置丰富，简单易用，降低使用门槛，减少用户重复开发。其他云服务日志只要接入LTS即可直接使用仪表盘模板，帮助企业基于日志数据进行数字化运营，助力企业数字化转型。

表 7-1 可视化方式

可视化方式	说明
统计图表	统计图表是云日志服务根据SQL查询语法渲染出的结果，包括表格、柱状图、折线图等多种图表类型，详细请参考 使用统计图表将日志可视化 。 旧版SQL引擎即将下线，新SQL引擎支持使用管道符（搜索 分析）配置图表，功能更丰富。新SQL引擎的语法兼容SQL92标准，与旧SQL引擎略有不同，如果查询失败，您需要手动修改SQL语句。
仪表盘	仪表盘是云日志服务提供的实时数据分析大盘。您可以在仪表盘查看多个基于SQL语句查询分析结果的统计图表，并能将多张统计图表同步保存到仪表盘中。详细请参考 使用仪表盘将日志可视化 。 云日志服务提供多种内置仪表盘模板，开箱即用，详细请参考 日志仪表盘模板（SQL分析方式） 。 管道符方式的仪表盘请参考 日志仪表盘模板（管道符方式） 。

约束与限制

此功能支持全部用户使用的局点有：华南-广州、华北-北京四、华北-乌兰察布一、华东-上海一、中国-香港、西南-贵阳一、亚太-新加坡、华北-北京一，支持部分白名单

用户使用的局点有：华东-上海二、亚太-曼谷、华南-深圳、中东-利雅得、亚太-雅加达，其他局点暂不支持该功能。

7.2 使用统计图表将日志可视化

7.2.1 统计图表概述

日志上报LTS后，支持通过SQL分析语法搜索关键日志数据，并将查询结果通过统计图表的方式进行可视化展示，帮助用户更容易地理解日志数据之间的变化。用户还可以将图表分析结果保存到仪表盘，进行长期监控。

约束与限制

- 一个日志流最多可创建100个图表。
- 一个仪表盘最多可添加50个图表。

统计图表类型

支持使用表格、柱状图、折线图等图表类型展示不同场景数据，详细请参考[表7-2](#)。

表 7-2 图表类型

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。
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。
折线图	折线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。
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。
数字图	数字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。
数字折线图	折线图和数字图的组合。折线图用于表示数据趋势和变化的，数字图则展示关键性指标。在一些需要同时显示趋势和关键数据点的场合适用。
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。
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。

统计图表操作说明

支持对图表进行新建、保存、另存为等操作。详细请参考[表7-3](#)。

表 7-3 操作说明

功能名称	功能描述
新建	选择不同的图表类型，将图表分析结果保存到仪表盘。
保存	对当前可视化图表进行保存。
另存为	对已有可视化图表进行复制。
下载	将图表分析结果下载到excel。
展开图表	可展开当前日志流下的可视化图表。
收起图表	可收起当前日志流下的可视化图表。

7.2.2 LTS 表格

表格作为最常见的数据展示类型，是组织整理数据最基本的手段，通过对数据的整理，达到快速引用和分析的目的。通过查询分析语法得到的数据结果默认以表格方式进行展示。

查看表格

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。
- 步骤3 选择“统计图表”。
- 旧版日志分析即将下线，推荐使用新版SQL引擎管道符（搜索|分析）在“统计图表”页面配置图表，功能更丰富。
- 步骤4 输入查询和分析语句，设置查询和分析的时间范围，单击“查询”。更多语法请参考[SQL函数](#)。
- 步骤5 日志下方默认使用表格展示日志数据查询结果，在通用配置下方，参考[表7-4](#)配置参数。

表 7-4 表格参数说明

类别	参数	说明
标准配置	格式化	将表格数据按照指定格式进行显示。
	单位	自定义配置表格数据的单位。
	小数点位数	设置显示数值小数点位数。
	图表名称字号	设置图表名称的字号大小。

类别	参数	说明
查询分析设置	隐藏字段	选择目标字段，将该字段在表格中隐藏。
表格配置	每页显示	每页显示的数据条数。
	显示总数	显示表格数据的总条目数。
列配置	对齐方式	表格数据的对齐方式，支持左对齐，右对齐以及居中。
	开启搜索	开启后，即可对表格列数据进行搜索功能。
	开启排序	开启后，即可对表格列数据进行排序功能。
	字体大小	表格字体的大小，取值范围为12px~24px。

----结束

7.2.3 LTS 柱状图

柱状图是使用垂直或水平的柱子显示类别之间的数值比较，用于描述分类数据，并统计每一个分类中的数量。

云日志服务（LTS）提供的柱状图，有垂直柱子和水平柱子，矩形块宽度一定，高度代表数值大小。有多列数据映射到Y轴时，采用分组柱状形式显示。

默认采用垂直柱子，您可以根据自己的需要进行选择。基本构成如下：

- X轴（横轴）
- Y轴（纵轴）
- 矩形块
- 图例

查看柱状图

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤3 选择“统计图表”。

旧版日志分析即将下线，推荐使用新版SQL引擎管道符（搜索|分析）在“统计图表”页面配置图表，功能更丰富。

步骤4 输入查询和分析语句，设置查询和分析的时间范围，单击“查询”。更多语法请参考[SQL函数](#)。

步骤5 单击图标，使用柱状图展示查询数据。在通用配置下方，参考[表7-5](#)配置参数。

表 7-5 柱状图参数说明

类别	参数	说明
标准配置	格式化	将Y轴按照指定格式进行显示。
	单位	自定义配置Y轴的单位。
	小数点位数	设置显示数值小数点位数。
	图表名称字号	设置图表名称的字号大小。
柱配置	方向	选择基础柱状图或横向柱状图。
	柱宽度	设置柱宽度。
	是否显示值	开启后，显示各个条形体对应的数值。
	值字体大小	设置各个条形体对应的数值字体大小
	是否堆叠	开启后，将堆叠显示Y轴数据。
查询分析设置	X轴数据	支持数字或字符串数据。
	Y轴数据	支持数字或字符串数据，可以选择多个数据。
图例配置	隐藏图例	开启后，可以隐藏图例和对比值的显示。
	图例位置	图例在图表中的位置，选择图表顶部或图表右边。
	对比数值	选择显示最大值、最小值、平均值、求和值等，可勾选多个。
图形配置	上边距	坐标轴距离图表上边界距离。
	下边距	坐标轴距离图表下边界距离。
	左边距	坐标轴距离图表左边界距离。
	右边距	坐标轴距离图表右边界距离。
Tooltip配置	不排序、升序、降序	提示框配置，当Y轴数据选择多个时，可对其进行排序显示。
X轴	显示X轴	开启后，显示X轴数据。
	X轴名称	设置X轴名称。
Y轴	显示Y轴	开启后，显示Y轴数据。
	Y轴名称	设置Y轴名称。
	Y轴位置	设置Y轴位置，左边或者右边。

----结束

7.2.4 LTS 折线图

折线图属于趋势类分析图表，一般用于表示一组数据在一个有序数据类别（多为连续时间间隔）上的变化情况，用于直观分析数据变化趋势。在折线图中，可以清晰的观测到数据在某一个周期内的变化，主要反映在：

- 递增性或递减性
- 增减的速率情况
- 增减的规律（如周期变化）
- 峰值和谷值

所以，折线图是用于分析数据随时间变化趋势的最佳选择。同时，也可以绘制多条线用于分析多组数据在同一时间周期的变化趋势，进而分析数据之间的相互作用和影响（如同增同减，成反比等）。

查看折线图

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。
- 步骤3 选择“统计图表”。
- 旧版日志分析即将下线，推荐使用新版SQL引擎管道符（搜索|分析）在“统计图表”页面配置图表，功能更丰富。
- 步骤4 输入查询和分析语句，设置查询和分析的时间范围，单击“查询”。更多语法请参考[SQL函数](#)。
- 步骤5 单击 图标，使用折线图展示查询数据。在通用配置下方，参考[表7-6](#)配置参数。

表 7-6 折线图参数说明

类别	参数	说明
标准配置	格式化	在下拉框选择K,Mil,Bil、1000000或Byte,KB,MB等格式，将Y轴按照指定格式进行显示。
	单位	自定义配置Y轴的单位。
	小数点位数	设置显示数值小数点位数。
	图表名称字号	设置图表名称的字号大小。
查询分析设置	X轴数据	支持数字或字符串数据。
	Y轴数据	支持数字或字符串数据，可以选择多个数据。
	维度列	请从下拉列表中选择，一般为有序数据类别。

类别	参数	说明
	趋势对比	当X轴为时间数据时，且不设置维度列时，可开启该按钮。 开启后，设置比较对象时间，时间小于等于24小时。设置完成后，将当前时间的数据与对象时间数据进行比较。
图例配置	隐藏图例	开启后，可以隐藏图例和对比值的显示。
	图例位置	选择图表顶部或图表右边。
	对比数值	选择显示最大值、最小值、平均值、求和值等，可勾选多个。
图形配置	连接方式	设置线图显示格式，可选择直线或曲线。
	线宽	折线的线宽。
	是否显示点	开启该功能后，显示折线的连接点。
	上边距	坐标轴距离图表上边界距离。
	下边距	坐标轴距离图表下边界距离。
	左边距	坐标轴距离图表左边界距离。
	右边距	坐标轴距离图表右边界距离。
Tooltip配置	排序方式	提示框配置，当Y轴数据选择多个时，可对其进行排序显示。
X轴	显示X轴	开启后，显示X轴数据。
	X轴名称	设置X轴名称。
Y轴	显示Y轴	开启后，显示Y轴数据。
	Y轴名称	设置Y轴名称。
	Y轴位置	设置Y轴位置，左边或者右边。

----结束

7.2.5 LTS 饼图

饼图用于表示不同分类的占比情况，通过弧度大小来对比各种分类。饼图通过将一个圆饼按照分类的占比划分成多个区块，整个圆饼代表数据的总量，每个区块表示该分类占总体的比例大小，所有区块的加和等于100%。基本构成如下：

- 扇形
- 文本百分比
- 图例

查看饼图

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。
- 步骤3 选择“统计图表”。
- 旧版日志分析即将下线，推荐使用新版SQL引擎管道符（搜索|分析）在“统计图表”页面配置图表，功能更丰富。
- 步骤4 输入查询和分析语句，设置查询和分析的时间范围，单击“查询”。更多语法请参考[SQL函数](#)。
- 步骤5 单击 图标，使用饼图展示查询数据。在通用配置下方，参考[表7-7](#)配置参数。

表 7-7 饼图参数说明

类别	参数	说明
标准配置	格式化	在下拉框选择 K,Mil,Bil、1000000 或 Byte,KB,MB 等格式，将Y轴按照指定格式进行显示。
	单位	自定义配置Y轴的单位。
	小数点位数	设置显示数值小数点位数。
	图表名称字号	设置图表名称的字号大小。

类别	参数	说明
饼图配置	饼图类型	包括饼图、环图和南丁格尔玫瑰图。 ● 饼图 饼图是一种用于展示各部分组成在整体中所占百分比的图形。它通过将一个圆形划分为不同的扇区，每个扇区的面积（或弧长和圆心角）大小对应所表示的数据的比例，从而直观地展示出各部分与整体之间的关系。 ● 环图 环图本质上是 将饼图中心挖空，相比于饼图来说有如下优点： - 在原有构成的基础上增加了总数显示，展示了更多的信息。 - 两个饼图直接进行比较是非常不直观的，两个环图间可以通过环状条长度进行简单的对比。 ● 南丁格尔玫瑰图 南丁格尔玫瑰图本质上并不是环图，而是在极坐标系下画出来的柱状图，每一个分类数据被圆弧平分，使用圆弧的半径长短表示数据的大小，相比于饼图来说有如下优点： - 饼图适用于不超过10条的分类数据，南丁格尔玫瑰图则适用于分类较多的场景（10-30条数据）。 - 由于半径和面积是成平方的关系，南丁格尔玫瑰图放大了各个分类数据之间值的差异，尤其适合对比大小相近的数值。 - 由于圆形有周期的特性，南丁格尔玫瑰图也适用于表示一个周期的时间概念，比如星期、月份。
	是否显示刻度	开启后，显示饼图上的文本标签，可用于说明图形的一些数据信息，比如值，名称等。
	刻度文本格式	可配置为分类、百分比、分类：百分比或分类：数值（百分比）。
	标签位置	开启是否显示刻度后，可配置此参数，调整标签在图表中的位置。
查询分析设置	数据	分类数据对应的数值。
	第一层数据	
	类目	分类数据。
	展示数量	显示分类数据的个数。
	排序方式	升序或降序。

类别	参数	说明
	其余归为其他	开启后，除了展示的数据，其余归为其他方式展示。
	添加分层	单击添加分层，设置第二层数据，每层数据包括类目、展示数量、排序方式、其余归为其他。
图例配置	隐藏图例	开启后，可以隐藏图例和图例内容的显示。
	图例内容	选择显示值和百分比，可勾选多个。
	图例位置	图例在图表中的位置，选择图表顶部或图表右边。
图形配置	外半径	指定饼图外半径值。取值范围为40~100。
	内半径	指定饼图内半径值。取值范围为0~100。
	上边距	坐标轴距离图表上边界距离。
	下边距	坐标轴距离图表下边界距离。
	左边距	坐标轴距离图表左边界距离。
	右边距	坐标轴距离图表右边界距离。

----结束

7.2.6 LTS 数字图

数字图通常用来表示单一数据点或关键性指标，能够更好地展示单一信息和数据的相对大小，是一种非常清晰的信息展示方式，适用于需要重点突出关键信息和数据的场合。通过数字图可以快速地呈现信息和数据，使得用户能够快速、直观地理解数据的趋势和关键指标。

查看数字图

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。
- 步骤3 选择“统计图表”。
- 旧版日志分析即将下线，推荐使用新版SQL引擎管道符（搜索|分析）在“统计图表”页面配置图表，功能更丰富。
- 步骤4 输入查询和分析语句，设置查询和分析的时间范围，单击“查询”。更多语法请参考[SQL函数](#)。
- 步骤5 单击  图标，使用数字图展示查询数据。在通用配置下方，参考[表7-8](#)配置参数。

表 7-8 数字图参数说明

类别	参数	说明
查询分析设置	数值列	支持数字或字符串数据。
	同比数据	选择待对比的字段，在图表中显示该字段对应的值。
主体设置	图表名称字号	设置图表名称的字号大小。
	格式化	将数据按照指定格式进行显示。
	数值字号	显示值的字号，取值范围为12px~80px。
	数值单位	显示值的单位
	单位字号	显示值单位的字号，取值范围为12px~50px。
	小数点位数	设置显示数值小数点位数。
	添加对比值	开启后，显示待对比字段对应的值。
	对比值格式化	将待对比数据按照指定格式进行显示。
	对比值字号	待对比值的字号，取值范围为12px~50px。
	对比值单位	待对比值的单位。
	对比值单位字号	显示待对比值单位的字号，取值范围为12px~50px。
	描述	对显示的数值及对比值趋势的描述，显示在数值下方。
背景配置	背景色	图表的背景颜色，支持深色或浅色。

----结束

7.2.7 LTS 数字折线图

数字图和折线图组合，同时显示趋势和关键数据点。可以帮助用户更好的理解数据和趋势变化，从而更好的进行业务决策。

查看数字折线图

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。
- 步骤3 选择“统计图表”。
- 旧版日志分析即将下线，推荐使用新版SQL引擎管道符（搜索|分析）在“统计图表”页面配置图表，功能更丰富。
- 步骤4 输入查询和分析语句，设置查询和分析的时间范围，单击“查询”。更多语法请参考[SQL函数](#)。

步骤5 单击  图标，使用数字折线图展示查询数据。在通用配置下方，参考[表7-9](#)配置参数。

表 7-9 数字折线图参数说明

类别	参数	说明
查询分析设置	X轴数据	支持数字或字符串数据。
	Y轴数据	支持数字或字符串数据，可以选择多个数据。
图表样式	连接方式	设置线图显示格式，可选择直线或曲线。
主体设置	图表名称字号	设置图表名称的字号大小。
	数据格式	将数据按照指定格式进行显示。
	数值字号	显示值的字号，取值范围为12px~80px。
	数值单位	显示值的单位。
	单位字号	显示值单位的字号，取值范围为12px~50px。
	小数点位数	设置显示数值小数点位数。
背景配置	背景色	图表的背景颜色，支持深色或浅色。

----结束

7.2.8 LTS 地图

以地图作为背景，通过图形颜色、图像标记的方式展示地理数据信息。云日志服务（LTS）提供的地图，包括中国地图和世界地图。您在查询和分析语句中使用特定函数（中国地图：ip_to_province函数，世界地图：ip_to_country函数）后，云日志服务（LTS）将以地图形式展示分析结果。

基本构成如下：

- 地图画布
- 色块

查看地图

- 步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤3 选择“统计图表”。

旧版日志分析即将下线，推荐使用新版SQL引擎管道符（搜索|分析）在“统计图表”页面配置图表，功能更丰富。

步骤4 输入查询和分析语句，设置查询和分析的时间范围，单击“查询”。更多语法请参考[SQL函数](#)。

步骤5 单击  图标，使用地图展示查询数据。在通用配置下方，参考[表7-10](#)配置参数。

表 7-10 地图参数说明表

参数	说明
地图类型	区域定位：设置地图显示的区域范围，包含中国省级地图和世界地图两种。
省份	地图类型为中国省级地图，该字段为具体省份。例如浙江省
国家	地图类型为世界地图，该字段为具体国家。例如中国
数值列	选择用于展示数值的字段。
图表名称字号	设置图表名称的字号大小。

----结束

7.2.9 LTS 漏斗图

漏斗图适用于业务流程比较规范、周期长、环节多的单流程单向分析，通过漏斗各环节业务数据的比较能够直观地发现和说明问题所在的环节，进而做出决策。漏斗图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。

查看漏斗图

- 步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2** 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。
- 步骤3** 选择“统计图表”。
- 旧版日志分析即将下线，推荐使用新版SQL引擎管道符（搜索|分析）在“统计图表”页面配置图表，功能更丰富。
- 步骤4** 输入查询和分析语句，设置查询和分析的时间范围，单击“查询”。更多语法请参考[SQL函数](#)。
- 步骤5** 单击  图标，使用漏斗图展示查询数据。在通用配置下方，参考[表7-11](#)配置参数。

表 7-11 漏斗图参数说明

参数	说明
系列名称	漏斗图的名称。
数值列	选择数值字段，某个字段对应的数值越大，在越上面。
隐藏图例	开启后，可以隐藏漏斗图上方的字段名显示。

----结束

7.3 使用仪表盘将日志可视化

7.3.1 创建日志仪表盘

仪表盘是一种数据可视化工具，它汇总并呈现关键性能指标、重要数据和分析结果，为用户提供了一目了然的业务或系统运行状况概览。

云日志服务提供多种仪表盘模板，用户可以直接使用LTS提供的仪表盘模板展示日志数据，或者将查询分析结果的统计图表同步保存到仪表盘中进行展示。

前提条件

- 已成功采集到日志。
- 对日志内容已完成结构化配置，具体操作请参考[设置云端结构化解析日志](#)。

约束与限制

此功能支持全部用户使用的局点有：华南-广州、华北-北京四、华北-乌兰察布一、华东-上海一、中国-香港、西南-贵阳一、亚太-新加坡、华北-北京一，支持部分白名单用户使用的局点有：华东-上海二、亚太-曼谷、华南-深圳、中东-利雅得、亚太-雅加达，其他局点暂不支持该功能。

创建仪表盘

- 一个账号最多可创建100个仪表盘。
- 一个日志流最多可创建100个图表。
- 一个仪表盘最多可添加50个图表。

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 单击，在“添加仪表盘分组”对话框，自定义填写“分组名称”。

分组名称只支持中英文、数字、中划线、下划线及小数点，且不能以小数点开头或结尾，长度不超过64。

图 7-1 添加仪表盘分组



步骤4 单击“确定”，创建分组成功。

步骤5 在左侧导航仪表盘下方，单击仪表盘分组名称。

步骤6 单击“添加仪表盘”，进入“创建仪表盘”页面，参考表7-12填写仪表盘相关参数。

表 7-12 创建仪表盘参数

参数名称	说明
仪表盘名称	自定义仪表盘名称，用于区分日志流下不同的仪表盘。 仅支持中英文、数字、中划线、下划线、小数点，不能以小数点开头和结尾，长度不超过255。
企业项目	选择业务需要的企业项目，默认为default。也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。 - 企业项目需要开通后才能使用，请参考如何开通企业项目。 - 支持将该企业项目资源迁出，详细请参考迁出企业项目资源。
添加到仪表盘分组	将新建的仪表盘进行分组管理。 不开启“添加到仪表盘分组”，新建仪表盘会添加到系统自带的“默认分组”中。 开启“添加到仪表盘分组”，新建仪表盘按“分组类型”进行添加： - 已有分组：选择已有的仪表盘分组。 - 新建分组：输入新建仪表盘分组名称。
简洁模式	仪表盘页面简介模式。 - 开启“简洁模式”，仪表盘界面不显示编辑、删除、添加过滤器等按钮。 - 关闭“简洁模式”，仪表盘界面显示编辑、删除、添加过滤器等按钮。
添加图表	添加可视化图表：将日志流的可视化图表加入仪表盘。 - 在“添加图表”区域中，鼠标悬浮在添加可视化图表模块，单击“开始添加图表”，进入添加可视化图表界面。 - 选择业务需要的日志流，根据业务需要勾选一个或多个图表名称前的 ☐，单击“确定”，进入仪表盘详情页后，调整图表信息，单击“保存设计”。 - 如果当前日志流未配置或没有当前需要的可视化图表，单击“前往添加图表”，新建图表。

参数名称	说明
	使用仪表盘模板：支持选择自定义模板（用户从已创建的仪表盘中提取的模板）和系统模板（LTS提供的系统内置模板，用户无法修改）： - 在“添加图表”区域中，鼠标悬浮在使用仪表盘模板模块，单击“使用仪表盘模板”，进入使用仪表盘模板界面。 - 根据业务需要选择仪表盘模板，单击下一步选择业务需要的日志流，根据业务需要勾选一个或多个日志流名称前的 ☐，单击“确定”。

- 步骤7** 仪表盘创建成功后，在仪表盘列表生成一条仪表盘信息。
- 单击仪表盘操作列的编辑，修改仪表盘名称和简洁模式。
 - 单击仪表盘操作列的移动分组，修改仪表盘分组。
 - 单击仪表盘操作列的删除按钮即可删除仪表盘。

----结束

新建可视化图表到仪表盘

- 步骤1** 在仪表盘目录下方，选中仪表盘分组，单击待操作的仪表盘名称进入详情页。
- 步骤2** 单击 ，在添加可视化图表界面中，选择相应日志流。单击“前往添加图表”。
- 步骤3** 在“添加图表”页面，单击“新建”，参照表7-13填写相关参数，填写完成后单击“确定”。

表 7-13 创建图表

参数	说明
图表名称	自定义图表名称，用于区分日志流下不同的图表。 仅支持中英文、数字、中划线、下划线、空格、括号、小数点，不能以小数点、空格开头或结尾。长度为1-64个字符。
可视化对象	- 默认语句“SELECT *”，表示查询该日志流内的结构化数据，其中*为结构化字段。 - 自行编辑SQL语句。
图表类型	LTS提供表格、柱状图、折线图等多种图表类型供用户选择。
同时添加到仪表盘	- 开启“同时添加到仪表盘”，勾选一个或多个仪表盘前面的 ☐，可将图表同步添加至勾选的仪表盘中。 - 关闭“同时添加到仪表盘”，则表示新建图表不在仪表盘显示。

步骤4 单击“确定”，可视化图表创建成功。

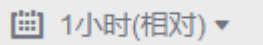
----结束

相关操作

创建仪表盘后，单击仪表盘名称进入详情页，您可以编辑仪表盘中的图表、移除仪表盘中的图表、调整仪表盘中图表的位置等，详细请参考[表7-14](#)。

表 7-14 相关操作

操作	说明
编辑仪表盘中的图表	将光标移至图表框右上角，单击  ，在下拉框中选择“编辑图表”，在可视化页面编辑图表。具体操作请参考 分析日志 。
移除仪表盘中的图表	将光标移至图表框右上角，单击  ，在下拉框中选择“移除图表”，单击“保存设计”，可将已创建图表删除。
调整仪表盘中图表的位置	将光标移至待操作的图表框内，选中该图表，可将该图表移动至仪表盘内任意位置，单击“保存设计”，调整当前图表布局。
调整仪表盘中图表的大小	将光标移至待操作的图表框右下角边缘，选中该图表，可根据业务展示内容需求调整图表大小，单击“保存设计”，调整当前图表布局。
编辑仪表盘中的过滤器	将光标移至过滤器框右上角，单击  ，在下拉框中选择“编辑”，在添加过滤器页面编辑过滤器，具体操作请参考 添加过滤器 。
复制仪表盘中的过滤器	将光标移至过滤器框右上角，单击  ，在下拉框中选择“复制”，跳转到添加过滤器页面，单击“确定”即可复制过滤器。
删除仪表盘中的过滤器	将光标移至过滤器框右上角，单击  ，在下拉框中选择“删除”，在弹出的“删除过滤器”提示框中，单击“确定”即可删除过滤器。
调整仪表盘中过滤器的大小	将光标移至待操作的过滤器右下角边缘，可根据业务展示内容需求调整过滤器大小，单击“保存设计”，调整当前过滤器布局。
自动刷新	单击右上角的  ，开启仪表盘自动刷新功能，选择自动刷新的时间，可使仪表盘中的所有图表数据自动进行刷新。自动刷新的时间有1分钟、5分钟、15分钟。
手动刷新	选择待操作的仪表盘，单击  可手动刷新当前页面。
全屏显示	选择待操作的仪表盘，单击  ，可全屏显示仪表盘。全屏后，勾选保持在线按钮，可以保持在线状态，会话一直有效，当前账号不会退出。

操作	说明
退出全屏显示	将光标移至屏幕上方，单击弹出的  ，或者单击  ，或者按键盘中的“Esc”可退出全屏模式。
全屏显示单个图表	选择待操作的仪表盘，单击取消退出编辑模式。将光标移至图表框右上角，单击  ，在下拉框中选择“全屏”，可全屏显示图表数据。
退出全屏显示单个图表	将光标移至屏幕上方，单击弹出的  ，或者单击  ，在下拉框中选择“退出全屏”，或者按键盘中的“Esc”可退出全屏模式。
手动刷新单个图表	选择待操作的仪表盘，将光标移至图表框右上角，单击  ，在下拉框中选择“刷新”，或者在全屏模式下，单击  ，在下拉框中选择“刷新”，可手动刷新当前图表页面。
查询时间设置	选择待操作的仪表盘，单击 前面的下拉框。 时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。 ● 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。 ● 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。 ● 自定义：表示查询指定时间范围的日志数据。
查看图表详情	选择待操作的仪表盘，将光标移至图表框右上角，单击  ，在下拉框中选择“查看图表详情”，可查看图表详情。
添加告警	选择待操作的仪表盘，将光标移至图表框右上角，单击  ，在下拉框中选择“添加告警”，可新建告警规则。
复制	选择待操作的仪表盘，将光标移至图表框右上角，单击  ，在下拉框中选择“复制”，可复制图表到当前仪表盘。
复制到其他仪表盘	选择待操作的仪表盘，将光标移至图表框右上角，单击  ，在下拉框中选择“复制到其他仪表盘”，可将该图表复制到其他仪表盘。

操作	说明
复制搜索分析语句	选择待操作的仪表盘，将光标移至图表框右上角，单击  ，在下拉框中选择“复制搜索分析语句”，可复制该图表的搜索分析语句。
导出图表数据	选择待操作的仪表盘，将光标移至图表框右上角，单击  ，在下拉框中选择“导出图表数据”，可导出图表数据。
提取模板	支持提取用户自定义创建的仪表盘模板，操作简单，即可快速复用已有的仪表盘模板。 1. 在自定义创建的仪表盘列表页面，单击目标仪表盘名称进入仪表盘详情页面。 2. 单击右上角  提取模板。 3. 在提取模板页面，按照业务需求填写参数。 4. 填写完成后，单击“确定”即可完成模板提取。如果没有选分组，返回仪表盘页面，在仪表盘模板下方的默认分组，可以查看提取成功的仪表盘模板。
导出仪表盘	• 在自定义创建的仪表盘列表页面，单击目标仪表盘操作列的“更多 > 导出仪表盘”即可导出仪表盘到本地查看。 • 在提取的仪表盘模板列表页面，单击目标仪表盘操作列的“导出仪表盘”即可导出仪表盘到本地查看。 • 在提取的仪表盘模板、自定义创建的仪表盘、系统自带的仪表盘模板列表页面，单击目标仪表盘名称进入仪表盘详情页面，单击右上角  即可导出仪表盘到本地查看。 • 在AOM控制台，可以将LTS导出的仪表盘导入AOM仪表盘，便于在AOM上可视化分析。详细操作请参考创建仪表盘章节中的导入仪表盘。

7.3.2 添加日志仪表盘过滤器

在云日志服务仪表盘中添加过滤器，即对整个仪表盘进行查询过滤或变量替换操作。

过滤器用于为仪表盘中的所有统计图表批量修改查询条件。每张统计图表实际为一个查询和分析语句，过滤器实质上是操作该查询和分析语句。

- 过滤器类型：通过日志字段的Key和Value进行过滤。在执行过滤器操作时，将其作为过滤条件增加到查询和分析语句前，使用AND或NOT连接。例如Key: Value AND [search query] | [sql query]，表示在原查询和分析语句的结果中，查找包含Key:Value的日志。在过滤器类型的过滤器中，Value可以多选，也可以直接输入。多选时过滤条件之间为或（or）关系。
- 时序过滤器类型：动态添加Label和Value进行过滤。在添加过滤器时，可以添加过滤条件，各个过滤条件之间使用AND连接。

前提条件

- 已成功采集到日志。
- 已添加图表到仪表盘。
- 日志流已配置结构化规则，具体操作请参考[结构化配置](#)。

限制条件

一个仪表盘最多可添加10个过滤器。

添加过滤器

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 在左侧导航栏中选择“仪表盘”，选择待操作的仪表盘。
- 步骤3 单击仪表盘名称进入详情页。
- 步骤4 单击，在“过滤器”页面中，参考[添加日志仪表盘过滤器](#)配置过滤器的相关参数。
- 不支持过滤数值型的字段。

表 7-15 添加过滤器参数

参数	说明
过滤器名称	仅支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。
查询方式	过滤器选中的条件与已有图表查询条件组合方式。支持AND和NOT两种方式，默认为AND。
Key值	配置需要过滤的字段。仅支持输入英文、数字、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾，且不能是纯数字。
别名	Key值的别名，用于区分不同字段。仅支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点开头、下划线开头或以小数点结尾。

参数	说明
静态列表项	设置Key值对应的Value，多次单击“添加静态列表项”可添加多个Value。 单击“添加静态列表项”可添加Value，需配置如下参数： • 列表项名称：需要过滤的Key值对应的Value字段名称。仅支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。 • 别名：Value的别名。 • 默认选中：开启“默认选中”，可直接对添加的Value进行过滤选中。 • 操作：单击  删除添加的Value。
添加动态列表项	动态列表项为对应查询和分析语句的查询结果，在不同的时间范围，查询结果是动态变化的。 • 关闭“添加动态列表项”，无法设置查询结果的动态变化。 • 开启“添加动态列表项”，通过添加动态列表项显示查询结果的动态变化，即为Key值配置动态Value。需配置如下参数： 日志组：选择待查询的日志组。 日志流：选择待查询的日志流。 SQL引擎：支持管道符版本和非管道符版本。管道符版本的SQL查询默认语法为* select *，非管道符版本的SQL查询默认语法为select * 动态列表来源：支持字段模糊匹配和SQL查询。 字段模糊匹配：选择当前日志流中配置的结构化字段。 SQL查询：输入SQL查询语句。单击“查询”，可预览动态列表项。

步骤5 设置完成后，单击“确定”即可完成过滤器的添加。

----结束

7.3.3 日志仪表盘模板（管道符方式）

7.3.3.1 APIG 仪表盘模板

APIG（API Gateway）提供高性能、高可用、高安全的API托管服务，能快速将企业服务能力包装成标准API服务，帮助您轻松构建、管理和部署任意规模的API，并上架API云商店进行售卖。借助API网关，可以简单、快速、低成本、低风险地实现内部系统集成、业务能力开放及业务能力变现。API网关帮助您变现服务能力的同时，降低企业研发投入，让您专注于企业核心业务，提升运营效率。

APIG仪表盘模板支持[查看APIG访问中心](#)、[查看APIG监控中心](#)、[分析APIG秒级监控](#)。

前提条件

- 已采集APIG日志，详情请参见[API网关APIG接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 APIG 访问中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“APIG仪表盘模板 > APIG访问中心”，查看图表详情。

- 过滤请求域名，所关联的查询分析语句如下所示：

```
* | select distinct(host)
```

- 过滤app_id，所关联的查询分析语句如下所示：

```
* | select distinct(app_id)
```

- PV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv", 86400) as diff from (select count(1) as "pv" from log))
```

- PV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv", 604800) as diff from (select count(1) as "pv" from log))
```

- UV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv", 86400) as diff from (select APPROX_COUNT_DISTINCT(my_remote_addr) as "uv" from log))
```

- UV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv", 604800) as diff from (select APPROX_COUNT_DISTINCT(my_remote_addr) as "uv" from log))
```

- 访问量PV分布(中国)**图表所关联的查询分析语句如下所示：

```
* | select ip_to_province(my_remote_addr) as province, sum(ori_pv) as pv from (select my_remote_addr, count(1) as ori_pv group by my_remote_addr ORDER BY ori_pv desc LIMIT 10000) where IP_TO_COUNTRY (my_remote_addr) = '中国' group by province HAVING province not in ('','保留地址','')
```

- 访问量PV分布(世界)**图表所关联的查询分析语句如下所示：

```
* | SELECT ip_to_country(my_remote_addr) as country,sum(ori_pv) as PV from (select my_remote_addr, count(1) as ori_pv group by my_remote_addr ORDER BY ori_pv desc LIMIT 10000) GROUP BY country HAVING country not in ('','保留地址','')
```

- 平均时延分布（中国）**所关联的查询分析语句如下所示：

```
* | SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(my_remote_addr) as province,sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select my_remote_addr, sum(request_time) as rt,count(1) as ori_pv group by my_remote_addr ORDER BY ori_pv desc LIMIT 10000) WHERE IP_TO_COUNTRY (my_remote_addr) = '中国' GROUP BY province ) where province not in ('','保留地址','')
```

- 平均时延分布（世界）**所关联的查询分析语句如下所示：

```
* | SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(my_remote_addr) as country,sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select my_remote_addr, sum(request_time) as rt,count(1) as ori_pv group by my_remote_addr ORDER BY ori_pv desc LIMIT 10000) GROUP BY country )where country not in ('','保留地址','')
```

- 今日PV/UV**所关联的查询分析语句如下所示：

```
* | select TIME_CEIL(_time,'PT600S') AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(my_remote_addr) as UV from log group by _time_ order by _time_
```

- **区域访问TOP10(省份)**所关联的查询分析语句如下所示：

```
* | select ip_to_province(my_remote_addr) as "province", sum(ori_pv) as "访问次数" from(select my_remote_addr, count(1) as ori_pv group by my_remote_addr ORDER BY ori_pv desc LIMIT 10000)group by "province" HAVING "province" <> '-1' order by "访问次数" desc limit 10
```
- **区域访问TOP10(城市)**图表所关联的查询分析语句如下所示：

```
* | select ip_to_city(my_remote_addr) as "city", sum(ori_pv) as "访问次数" from(select my_remote_addr, count(1) as ori_pv group by my_remote_addr ORDER BY ori_pv desc LIMIT 10000) group by "city" HAVING "city" <> '-1' order by "访问次数" desc limit 10
```
- **Host访问TOP10**图表所关联的查询分析语句如下所示：

```
* | select host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```
- **UserAgent访问TOP10**图表所关联的查询分析语句如下所示：

```
* | select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc limit 10
```
- **设备占比(终端)**图表所关联的查询分析语句如下所示：

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type , count(1) as total group by type
```
- **设备占比(系统)**图表所关联的查询分析语句如下所示：

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total group by type HAVING type != 'other'
```
- **TOP URL**图表所关联的查询分析语句如下所示：

```
* |select router_uri , count(1) as pv, APPROX_COUNT_DISTINCT(my_remote_addr) as UV, round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group by router_uri ORDER by pv desc
```
- **TOP 访问IP**图表所关联的查询分析语句如下所示：

```
* | select my_remote_addr as "来源IP",ip_to_country(my_remote_addr) as "国家",ip_to_province(my_remote_addr) as "省份",ip_to_city(my_remote_addr) as "城市",ip_to_provider(my_remote_addr) as "运营商",count(1) as "PV" group by my_remote_addr ORDER by "PV" desc limit 100
```

----结束

查看 APIG 监控中心

步骤1 在仪表盘模板下方，选择“APIG仪表盘模板 > APIG监控中心”，查看图表详情。

- **过滤请求域名**，所关联的查询分析语句如下所示：

```
* | select distinct(host)
```
- **过滤app_id**，所关联的查询分析语句如下所示：

```
* | select distinct(app_id)
```
- **访问量PV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_CEIL ( _time, 'PT300S' ) AS _time_, count( 1 ) AS PV FROM log GROUP BY _time_ order by _time_
```
- **请求成功率**图表所关联的查询分析语句如下所示：

```
* | select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```
- **平均延迟**图表所关联的查询分析语句如下所示：

```
* | select round(avg(request_time) * 1000, 3) as cnt
```
- **4XX请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
```
- **404请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 404
```
- **429请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 429
```
- **504请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 504
```

- **5XX请求数**图表所关联的查询分析语句如下所示：

```
status" >= 500 | SELECT TIME_CEIL ( TIME_PARSE(time_local, 'dd/MMMM/yyyy:HH:mm:ss ZZZ'),  
'PT300S') AS _time_, count( 1 ) AS cnt FROM log GROUP BY _time_ order by _time_
```

- **状态码分布**图表所关联的查询分析语句如下所示：

```
* | SELECT status, COUNT(1) AS rm GROUP BY status
```

- **访问量UV**图表所关联的查询分析语句如下所示：

```
* | select TIME_CEIL(__time,'PT600S') AS _time_, APPROX_COUNT_DISTINCT(my_remote_addr) as UV  
from log group by _time_ order by _time_
```

- **流量**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_CEIL(__time,'PT600S') AS _time_,sum(request_length) / 1024.0 AS "入流量/  
KB",sum(bytes_sent) / 1024.0 AS "出流量/KB" group by _time_ order by _time_
```

- **访问失败率**图表所关联的查询分析语句如下所示：

```
* | select TIME_CEIL(__time,'PT600S') AS _time_,sum(case when status >= 400 then 1 else 0 end) *  
100.0 / count(1) as "失败率", sum(case when status >=500 THEN 1 ELSE 0 END)*100.0/COUNT(1) as  
"5XX比例" group by _time_ order by _time_
```

- **延迟**图表所关联的查询分析语句如下所示：

```
* | select TIME_CEIL(__time,'PT600S') as _time_,avg(request_time) * 1000 as "平均",  
APPROX_QUANTILE_DS("request_time", 0.50)*1000 as "P50", APPROX_QUANTILE_DS("request_time",  
0.90)*1000 as "P90",APPROX_QUANTILE_DS("request_time", 0.99)*1000 as  
'P99',APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as 'P9999' group by _time_ order by  
_time_
```

- **Host请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0  
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END,  
3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出  
流量(KB)" FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( my_remote_addr )  
AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率  
(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",  
sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY "host" ) ORDER BY pv  
DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,  
2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,  
3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 )  
AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS  
"P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1  
ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟  
(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv,round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 )  
AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE  
0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"host" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT upstream_uri, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE  
0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END,  
3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出  
流量(KB)" FROM ( SELECT upstream_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( my_remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 /  
count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
sum( request_length ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE  
"host" != " " GROUP BY upstream_uri ) ORDER BY pv desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT upstream_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM( SELECT upstream_uri, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY upstream_uri ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT my_remote_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( my_remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY addr having length(my_remote_addr) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT my_remote_addr as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " and "my_remote_addr" != '-' GROUP BY addr ) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT my_remote_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " and "my_remote_addr" != '-' GROUP BY addr) ORDER BY "访问失败率(%)" desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT upstream_uri, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT upstream_uri, count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY upstream_uri ) ORDER BY "平均延迟(ms)" desc
```

----结束

查看 APIG 秒级监控

步骤1 在仪表盘模板下方，选择“APIG仪表盘模板 > APIG秒级监控”，查看图表详情。

- 过滤请求域名，所关联的查询分析语句如下所示：

```
* | select distinct(host)
```

- 过滤app_id，所关联的查询分析语句如下所示：

```
* | select distinct(app_id)
```

- **QPS**图表所关联的查询分析语句如下所示：

```
* | select TIME_CEIL(__time,'PT1S') AS _time_, COUNT(*) as QPS from log group by _time_ order by _time_ limit 100000
```


- **成功率**图表所关联的查询分析语句如下所示：

```
* | select TIME_CEIL(__time,'PT5S') as "time", cast(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1) as double PRECISION) as '成功率' from log group by "time" order by "time" limit 10000
```

- **延迟**图表所关联的查询分析语句如下所示：

```
* | select TIME_CEIL(__time,'PT5S') as "time", avg(request_time)* 1000.0 as "访问延迟", avg(upstream_response_time)* 1000.0 as "Upstream延迟" from log group by "time" order by time limit 10000
```

- **流量**图表所关联的查询分析语句如下所示：

```
* | select TIME_CEIL(__time,'PT5S') as "time", cast(sum("request_length") / 1024.0 as double PRECISION) as "请求流量", cast(sum("body_bytes_sent") / 1024.0 as double PRECISION) as "返回body流量" group by "time" order by "time" limit 10000
```

- **状态码**图表所关联的查询分析语句如下所示：

```
* | select TIME_CEIL ( __time, 'PT5S' ) AS "time", SUM( CASE WHEN "status" >= 200 AND "status" < 300 THEN 1 ELSE 0 END ) AS "2XX", SUM( CASE WHEN "status" >= 300 AND "status" < 400 THEN 1 ELSE 0 END ) AS "3XX", SUM( CASE WHEN "status" >= 400 AND "status" < 500 THEN 1 ELSE 0 END ) AS "4XX", SUM( CASE WHEN "status" >= 500 AND "status" < 600 THEN 1 ELSE 0 END ) AS "5XX", SUM( CASE WHEN "status" < 200 OR "status" >= 600 THEN 1 ELSE 0 END ) AS "其他" FROM log WHERE TIME_PARSE ( time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ' ) IS NOT NULL GROUP BY "time" ORDER BY "time" ASC LIMIT 100000
```

- **后端响应码**图表所关联的查询分析语句如下所示：

```
* | select TIME_CEIL ( __time, 'PT5S' ) AS "time", SUM( CASE WHEN "upstream_status" >= 200 AND "upstream_status" < 300 THEN 1 ELSE 0 END ) AS "2XX", SUM( CASE WHEN "upstream_status" >= 300 AND "upstream_status" < 400 THEN 1 ELSE 0 END ) AS "3XX", SUM( CASE WHEN "upstream_status" >= 400 AND "upstream_status" < 500 THEN 1 ELSE 0 END ) AS "4XX", SUM( CASE WHEN "upstream_status" >= 500 AND "upstream_status" < 600 THEN 1 ELSE 0 END ) AS "5XX", SUM( CASE WHEN "upstream_status" < 200 OR "upstream_status" >= 600 THEN 1 ELSE 0 END ) AS "其他" FROM log WHERE TIME_PARSE ( time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ' ) IS NOT NULL GROUP BY "time" ORDER BY "time" ASC LIMIT 100000
```

----结束

7.3.3.2 CCE 仪表盘模板

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群。

CCE仪表盘模板支持[查看CCE日志节点操作](#)、[查看CCE日志K8s对象操作](#)、[查看CCE日志K8s事件查询](#)、[查看CCE日志K8s事件中心](#)、[查看CCE日志聚合检索](#)、[查看CCE日志账号操作审计](#)和[查看CCE日志审计中心](#)。

前提条件

- 已采集CCE日志，详情请参见[云容器引擎CCE应用日志接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CCE 日志节点操作

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志节点操作”，查看图表详情。

- 过滤节点名称，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.name")
```

- 过滤操作用户，所关联的查询分析语句如下所示：

```
* | select distinct("user.username")
```


- 过滤状态码，所关联的查询分析语句如下所示：
* | select distinct("responseStatus.code")
- 过滤操作类型，所关联的查询分析语句如下所示：
* | select distinct("verb")
- 节点数趋势图表所关联的查询分析语句如下所示：
* | SELECT time_series(TIME_PARSE(LEFT(requestReceivedTimestamp, 23),'yyyy-MM-dd"T"HH:mm:ss.SSS'), 'PT1H', 'yyyy-MM-dd HH', '0' ,'+08:00') as "dt", count(DISTINCT("objectRef.name")) as "节点数" where "objectRef.resource" = 'nodes' and "objectRef.subresource" = 'status' and "verb" in ('update', 'patch') and "user.username" = 'system:node' group by "dt" order by "dt" desc limit 10000
- 非系统用户操作趋势图表所关联的查询分析语句如下所示：
* | SELECT time_series(TIME_PARSE(LEFT(requestReceivedTimestamp, 23),'yyyy-MM-dd"T"HH:mm:ss.SSS'), 'PT1H', 'yyyy-MM-dd HH', '0' ,'+08:00') as "dt", count(*) as "请求", "user.username" where "objectRef.resource" = 'nodes' and "user.username" not in ('kube-controller-manager', 'kube-apiserver-kubelet-client', 'apiserver') and "user.username" not like 'system:%' and "verb" in ('create','delete','update','patch') group by "dt", "user.username" order by "dt", "请求" desc limit 10000
- create操作状态码分布图表所关联的查询分析语句如下所示：
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'create' group by "状态码"
- delete操作状态码分布图表所关联的查询分析语句如下所示：
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'delete' group by "状态码"
- patch操作状态码分布图表所关联的查询分析语句如下所示：
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'patch' group by "状态码"
- update操作状态码分布图表所关联的查询分析语句如下所示：
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'update' group by "状态码"
- 节点封锁/解除封锁操作状态码分布图表所关联的查询分析语句如下所示：
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "requestObject" in ('{"spec":{"unschedulable":false}}', '{"spec":{"unschedulable":true}}') group by "状态码"
- Label操作状态码分布图表所关联的查询分析语句如下所示：
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" in ('patch','update') and CONTAINS_STRING("requestObject", 'labels') and CONTAINS_STRING("requestObject", 'metadata') group by "状态码"
- Taint操作状态码分布图表所关联的查询分析语句如下所示：
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" in ('patch','update') and "requestObject" = 'taints' group by "状态码"
- 驱逐操作状态码分布图表所关联的查询分析语句如下所示：
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.subresource" = 'eviction' and "objectRef.resource" = 'pods' and "verb" = 'create' group by "状态码"
- 节点增删操作列表图表所关联的查询分析语句如下所示：
* | select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "verb" AS "操作动作", "stageTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "objectRef.resource" = 'nodes' and "verb" in ('create','delete')
- Taint操作列表图表所关联的查询分析语句如下所示：
* | select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "requestObject" AS "Taints", "requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "objectRef.resource" = 'nodes' and "verb" = 'patch' and "requestObject" = 'taints'
- 驱逐操作列表图表所关联的查询分析语句如下所示：
* | select "auditID" AS "Audit ID", "objectRef.name" AS "pod", "sourceIPs" AS "源地址", "requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "objectRef.resource" = 'pods' and "verb" = 'create' and "objectRef.subresource" = 'eviction'

- **Label操作列表**图表所关联的查询分析语句如下所示：

```
* | select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "requestObject" AS "Label",  
"requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code"  
AS "状态码" where "objectRef.resource" = 'nodes' and "verb" = 'patch' and "requestObject" = 'labels'
```

- **封锁操作列表**图表所关联的查询分析语句如下所示：

```
* | select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "requestReceivedTimestamp" AS "操  
作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "verb" = 'patch'  
and "objectRef.resource" = 'nodes' and CONTAINS_STRING("requestObject",true) and  
CONTAINS_STRING("requestObject",unschedulable)
```

- **取消封锁操作列表**图表所关联的查询分析语句如下所示：

```
* | select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "requestReceivedTimestamp" AS "操  
作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "verb" = 'patch'  
and "objectRef.resource" = 'nodes' and "requestObject" not in ('true','taints','unschedulable')
```

----结束

查看 CCE 日志 K8s 对象操作

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志K8s对象操作”，查看图表详情。

- 过滤命名空间，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.namespace")
```

- 过滤操作类型，所关联的查询分析语句如下所示：

```
* | select distinct("verb")
```

- 过滤状态码，所关联的查询分析语句如下所示：

```
* | select distinct("responseStatus.code")
```

- 过滤资源对象，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.name")
```

- 过滤资源类型，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.resource")
```

- 过滤操作用户，所关联的查询分析语句如下所示：

```
* | select distinct("user.username")
```

- **重要操作趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT REPLACE(LEFT(requestReceivedTimestamp, 16),'T',' ') AS "dt", "verb" as "操作类型",  
count(*) as "count" where "verb" in ('create','delete','update','patch') and "objectRef.resource" in  
( 'deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','config  
maps','persistentvolumeclaims') group by "dt","操作类型" order by "dt" limit 10000
```

- **非系统用户操作趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT REPLACE(LEFT(requestReceivedTimestamp, 16),'T',' ') AS "dt", count(*) as "请求次数",  
"user.username" WHERE "user.username" not in ('kube-controller-manager','kube-apiserver-kubelet-  
client','apiserver') and "user.username" not like 'system:%' and "verb" in  
( 'create','delete','update','patch') and "objectRef.resource" in  
( 'deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','secrets',  
'pvcs') group by "dt","user.username" order by "dt" limit 10000
```

- **create操作资源类型分布**图表所关联的查询分析语句如下所示：

```
* | select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'create' and  
"objectRef.resource" in  
( 'deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','config  
maps','persistentvolumeclaims') group by "objectRef.resource"
```

- **delete操作资源类型分布**图表所关联的查询分析语句如下所示：

```
* | select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'delete' and  
"objectRef.resource" in  
( 'deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','config  
maps','persistentvolumeclaims') group by "objectRef.resource"
```

- **update操作资源类型分布**图表所关联的查询分析语句如下所示：

```
* | select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'update' and  
"objectRef.resource" in
```

```
('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```

- **patch操作资源类型分布**图表所关联的查询分析语句如下所示：

```
* | select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```

- **create操作用户分布**图表所关联的查询分析语句如下所示：

```
* | select "user.username" as "操作用户", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```

- **delete操作用户分布**图表所关联的查询分析语句如下所示：

```
* | select "user.username" as "操作用户", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```

- **update操作用户分布**图表所关联的查询分析语句如下所示：

```
* | select "user.username" as "操作用户", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```

- **patch操作用户分布**图表所关联的查询分析语句如下所示：

```
* | select "user.username" as "操作用户", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```

- **create操作状态码分布**图表所关联的查询分析语句如下所示：

```
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```

- **delete操作状态码分布**图表所关联的查询分析语句如下所示：

```
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```

- **update操作状态码分布**图表所关联的查询分析语句如下所示：

```
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```

- **patch操作状态码分布**图表所关联的查询分析语句如下所示：

```
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```

- **create操作趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT REPLACE(LEFT(stageTimestamp, 16), 'T', ' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```

- **delete操作趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT REPLACE(LEFT(stageTimestamp, 16), 'T', ' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```

- **update操作趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT REPLACE(LEFT(stageTimestamp, 16), 'T', ' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in
```

```
('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```

- **patch操作趋势**图表所关联的查询分析语句如下所示:

```
* | SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```

----结束

查看 CCE 日志 K8s 事件查询

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志K8s对象操作”，查看图表详情。

- 事件等级分为Warning、Normal、Error。

- 过滤事件类型，所关联的查询分析语句如下所示:

```
* | select distinct("name")
```

- 过滤集群ID，所关联的查询分析语句如下所示:

```
* | select distinct("cluster_id")
```

- 过滤命名空间，所关联的查询分析语句如下所示:

```
* | select distinct("namespace")
```

- 过滤名称，所关联的查询分析语句如下所示:

```
* | select distinct("resource_name")
```

- **事件总数**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( "_time_", 'yyyy-MM-dd HH:mm:ss', " ) as "_time_", "total" FROM ( SELECT TIME_CEIL ( __time, 'PT1H' ) AS "_time_", count( 1 ) AS "total" FROM log GROUP BY "_time_" ) order by "_time_"
```

- **普通事件数**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( "_time_", 'yyyy-MM-dd HH:mm:ss', " ) as "_time_", "total" FROM ( SELECT TIME_CEIL ( __time, 'PT1H' ) AS "_time_", count( 1 ) AS "total" FROM log where "type" = 'Normal' GROUP BY "_time_" ) order by "_time_"
```

- **警告事件数**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( "_time_", 'yyyy-MM-dd HH:mm:ss', " ) as "_time_", "total" FROM ( SELECT TIME_CEIL ( __time, 'PT1H' ) AS "_time_", count( 1 ) AS "total" FROM log where "type" = 'Warning' GROUP BY "_time_" ) order by "_time_"
```

- **错误事件数**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( "_time_", 'yyyy-MM-dd HH:mm:ss', " ) as "_time_", "total" FROM ( SELECT TIME_CEIL ( __time, 'PT1H' ) AS "_time_", count( 1 ) AS "total" FROM log where "type" = 'Error' GROUP BY "_time_" ) order by "_time_"
```

- **重要事件分布**图表所关联的查询分析语句如下所示:

```
* | select "name",count(1) as 'total' from log group by "name"
```

- **重要事件统计**图表所关联的查询分析语句如下所示:

```
* | SELECT "type" as "等级", "name" as "类型", count(1) as "总数", "reason" as "详细内容" from log where "type" != 'Normal' group by "type", "name", "reason" order by "总数" desc
```

- **Pod重要事件统计**图表所关联的查询分析语句如下所示:

```
* | SELECT "type" as "等级", "name" as "类型", count(1) as "总数", "reason" as "详细内容" from log where "type" != 'Normal' and "resource_kind" = 'Pod' group by "type", "name", "reason" order by "总数" desc
```

- **最近100条事件**图表所关联的查询分析语句如下所示:

```
* | select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as "Time", "type" as "等级", "name" as "类型", "reason" as "详细内容", "namespace" as "命名空间" from log order by __time desc limit 100
```

----结束

查看 CCE 日志 K8s 事件中心

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志K8s事件中心”，查看图表详情。

- 事件等级分为Warning、Normal、Error。

- 过滤事件类型，所关联的查询分析语句如下所示：

```
* | select distinct("name")
```

- 过滤集群ID，所关联的查询分析语句如下所示：

```
* | select distinct("cluster_id")
```

- 过滤命名空间，所关联的查询分析语句如下所示：

```
* | select distinct("namespace")
```

- 过滤名称，所关联的查询分析语句如下所示：

```
* | select distinct("resource_name")
```

- Contrack Full**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='ContrackFull' ) )
```

- 事件同步异常**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='NTPIsDown' ) )
```

- 节点Pid不足**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name" in ('PIDPressure','NodeHasPIDPressure') ) )
```

- 节点FD不足**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='NodeHasFDPressure' ) )
```

- 节点磁盘空间不足**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='NodeHasDiskPressure' ) )
```

- Pod OOM**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name" in ('OOMKilling','PodOOMKilling') ) )
```

- DockerHung**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='Failed' and "reason"='DockerHung' ) )
```

- 节点重启**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='NodeRebooted' ) )
```

- 镜像拉取失败**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='Failed' and "reason"='ImagePullBackOff' ) )
```

- 节点OOM**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='SystemOOM' ) )
```

- Pod启动失败**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select
compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"= 'Failed' and
"resource_kind" = 'Pod' and "reason" = 'ImagePullBackOff' ) )
```

- 事件分布图表所关联的查询分析语句如下所示：

```
* | select "type", count(*) as "事件数" group by "type"
```

- **Warning事件趋势**图表所关联的查询分析语句如下所示：

```
* | select time_series(__time, 'PT1H', 'yyyy-MM-dd HH', '0') as "dt",count(1) as "count" from log
where "type" = 'Warning' group by "dt" order by "dt"
```

- **Error事件趋势**图表所关联的查询分析语句如下所示：

```
* | select time_series(__time, 'PT1H', 'yyyy-MM-dd HH', '0') as "dt",count(1) as "count" from log
where "type" = 'Error' group by "dt" order by "dt"
```

- **Pod OOM事件列表**所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "Time", "resource_kind" as "事
件目标", "name" as "类型", "resource_name" as "目标名", "reason" as "详细内容" from log where
"name" in ('OOMKilling','PodOOMKilling') order by __time desc limit 100
```

- **Pod驱动事件列表**所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "Time", "resource_kind" as "事
件目标", "name" as "类型", "resource_name" as "目标名", "reason" as "详细内容" from log where
"name" = 'NodeControllerEviction' order by __time desc limit 100
```

- **重要事件列表**所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "Time", "type" as "等级",
"resource_kind" as "事件目标", "name" as "类型", "resource_name" as "目标名", "reason" as "详细内
容" from log where "type" in ('Warning','Error') order by __time desc limit 100
```

----结束

查看 CCE 日志聚合检索

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志聚合检索”，查看图表详情。

- 过滤命名空间，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.namespace")
```

- 过滤操作用户，所关联的查询分析语句如下所示：

```
* | select distinct("user.username")
```

- 过滤状态码，所关联的查询分析语句如下所示：

```
* | select distinct("responseStatus.code")
```

- 过滤操作类型，所关联的查询分析语句如下所示：

```
* | select distinct("verb")
```

- 过滤资源对象，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.name")
```

- 过滤资源类型，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.resource")
```

- 过滤请求URL，所关联的查询分析语句如下所示：

```
* | select distinct("requestURI")
```

- 过滤userAgent，所关联的查询分析语句如下所示：

```
* | select distinct("userAgent")
```

- **操作用户分布趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "user.username" as "操作用户", count(*)
as "count" group by dt, "user.username" order by dt limit 10000
```

- **命名空间分布趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.namespace" as "命名空间",
count(*) as "count" group by dt, "objectRef.namespace" order by dt limit 10000
```

- **操作类型分布趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "verb" as "操作类型", count(*) as "count"
group by dt, verb order by dt limit 10000
```

- **状态码分布趋势**图表所关联的查询分析语句如下所示：
`* | SELECT REPLACE(LEFT(stageTimestamp, 16), 'T', ' ') AS dt, cast("responseStatus.code" as varchar) as "返回码", count(*) as "count" group by dt, "返回码" order by dt limit 10000`
- **资源类型分布趋势**图表所关联的查询分析语句如下所示：
`* | SELECT REPLACE(LEFT(stageTimestamp, 16), 'T', ' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" group by dt, "objectRef.resource" order by dt limit 10000`
- **重要操作列表**所关联的查询分析语句如下所示：
`* | select "auditID" AS "Audit ID", "verb" AS "操作类型", "requestReceivedTimestamp" AS "开始时间", "stageTimestamp" AS "结束时间", "user.username" AS "操作账号", "sourceIPs" AS "操作源", "userAgent", "objectRef.namespace" AS "命名空间", CONCAT(CONCAT("objectRef.resource", '/'), "objectRef.subresource") AS "操作对象", "objectRef.name" AS "资源名", "responseStatus.code" AS "返回码"`

----结束

查看 CCE 日志账号操作审计

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志账号操作审计”，查看图表详情。

- **过滤用户名**，所关联的查询分析语句如下所示：
`* | select distinct("user.username")`
- **过滤命名空间**，所关联的查询分析语句如下所示：
`* | select distinct("objectRef.namespace")`
- **过滤状态码**，所关联的查询分析语句如下所示：
`* | select distinct("responseStatus.code")`
- **资源创建数**图表所关联的查询分析语句如下所示：
`* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( total, 86400) as diff from( select count(1) as total from log where "verb" = 'create' ) )`
- **资源修改数**图表所关联的查询分析语句如下所示：
`* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( total, 86400) as diff from( select count(*) as "total" from log where "verb" in ('update','patch')) )`
- **资源删除数**图表所关联的查询分析语句如下所示：
`* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( total, 86400) as diff from( select count(*) as "total" from log where "verb" = 'delete' ) )`
- **操作命名空间分布**图表所关联的查询分析语句如下所示：
`* | select case when "objectRef.namespace" is null then '_all_' else "objectRef.namespace" end as ns, count(1) as total group by ns limit 10000`
- **删除资源分布**图表所关联的查询分析语句如下所示：
`* | SELECT "objectRef.resource" as "resource", count(1) as "count" where "verb" = 'delete' group by "resource"`
- **操作轨迹**图表所关联的查询分析语句如下所示：
`* | select case when "操作" is null then '无' else "操作" end as "操作", "时间", v from (select concat(CASE WHEN "objectRef.subresource" is null then "objectRef.resource" else "objectRef.subresource" end, '[', verb, ']') as "操作", time_series(__time, 'PT1H', 'yyyy-MM-dd HH', '0') as "时间", count(1) as v from log where "verb" in ('create', 'patch', 'update', 'delete') group by "操作", "时间" order by "时间" desc limit 10000) order by "时间"`
- **资源操作分布**图表所关联的查询分析语句如下所示：
`* | select CASE WHEN "objectRef.subresource" is null then "objectRef.resource" else "objectRef.subresource" end as "资源", verb as "操作", count(1) as total where "verb" in ('create','update','patch','delete') group by "资源", "操作" limit 10000`
- **创建资源列表**所关联的查询分析语句如下所示：
`* | SELECT "auditID" as "事件ID", time_format("__time", 'yyyy-MM-dd HH:mm:ss') as "操作时间", "requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码", "sourceIPs" as "源地址", "requestObject" as "详细内容" where "verb" = 'create' order by __time desc limit 1000`
- **修改资源列表**所关联的查询分析语句如下所示：

```
* | SELECT auditID as "事件ID", time_format("__time",'yyyy-MM-dd HH:mm:ss') as "操作时间",
"requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码",
"sourceIPs" as "源地址", requestObject as "详细内容" where "verb" in ('update','patch') order by
__time desc limit 1000
```

- 资源访问列表所关联的查询分析语句如下所示：

```
* | SELECT auditID as "事件ID", time_format("__time",'yyyy-MM-dd HH:mm:ss') as "操作时间",
"requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码",
"sourceIPs" as "源地址", requestObject as "详细内容" where "verb" in ('get','list') order by __time
desc limit 1000
```

- 资源删除列表所关联的查询分析语句如下所示：

```
* | SELECT auditID as "事件ID", time_format("__time",'yyyy-MM-dd HH:mm:ss') as "操作时间",
"requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码",
"sourceIPs" as "源地址", requestObject as "详细内容" where "verb" = 'delete' order by __time desc
limit 1000
```

----结束

查看 CCE 日志审计中心

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志审计中心”，查看图表详情。

- 过滤命名空间，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.namespace")
```

- 过滤操作用户，所关联的查询分析语句如下所示：

```
* | select distinct("user.username")
```

- 过滤操作类型，所关联的查询分析语句如下所示：

```
* | select distinct("verb")
```

- 过滤状态码，所关联的查询分析语句如下所示：

```
* | select distinct("responseStatus.code")
```

- 过滤资源对象，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.name")
```

- 过滤资源类型，所关联的查询分析语句如下所示：

```
* | select distinct("objectRef.resource")
```

- 过滤请求URL，所关联的查询分析语句如下所示：

```
* | select distinct("requestURI")
```

- 过滤UserAgent，所关联的查询分析语句如下所示：

```
* | select distinct("userAgent")
```

- 总审计记录数图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select
compare( total , 86400) as diff from( select count(1) as total from log ) )
```

- 操作用户数图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select
compare( total , 86400) as diff from( select count(distinct("user.username")) as total from log ) )
```

- 活跃节点数图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select
compare( total , 86400) as diff from( select count(DISTINCT "user.username") as total from log
where "objectRef.resource" = 'nodes' and "objectRef.subresource" = 'status' and "verb" in
('update','put','patch') and "user.username" in ('node','system')) )
```

- 异常访问次数图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select
compare( total , 86400) as diff from( select count(1) as total from log where "responseStatus.code"
>= 400) )
```

- 敏感操作次数图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select
compare( total , 86400) as diff from( select count(1) as "total" from log where ("verb" = 'create'
AND "objectRef.subresource" = 'exec') OR ("verb" = 'create' AND "objectRef.subresource" = 'attach'
AND "objectRef.resource" = 'pods') OR ("objectRef.resource" = 'secrets' AND "verb" = 'get' AND
```



```
("user.username" != 'apiserver') AND ("user.username" not like 'system:node:%')) OR ("verb" = 'delete' AND ("user.username" not like 'system:node:%') AND ("user.username" not like 'system:serviceaccount:kube-system:%') AND ("user.username" != 'system:apiserver') AND ("user.username" != 'system:apiserver') AND ("user.username" != 'system:kube-scheduler') AND ("user.username" != 'system:kube-controller-manager'))))
```

- **创建操作次数图表所关联的查询分析语句如下所示：**

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( total, 86400) as diff from( select count(1) as total from log where verb = 'create' ) )
```

- **更新操作次数图表所关联的查询分析语句如下所示：**

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( total, 86400) as diff from( select count(1) as total from log where verb in ('update','patch')) )
```

- **删除操作次数图表所关联的查询分析语句如下所示：**

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( total, 86400) as diff from( select count(1) as total from log where verb = 'delete' ) )
```

- **操作用户分布图表所关联的查询分析语句如下所示：**

```
* | select "user.username" as "用户名", count(*) as "count" group by "用户名" order by "count" desc
```

- **命名空间分布图表所关联的查询分析语句如下所示：**

```
* | select "objectRef.namespace" as "命名空间", count(*) as "count" group by "命名空间"
```

- **资源类型分布图表所关联的查询分析语句如下所示：**

```
* | select "objectRef.resource" as "资源类型", count(*) as "count" group by "资源类型" order by "count" desc limit 20
```

- **操作类型分布图表所关联的查询分析语句如下所示：**

```
* | select verb as "操作类型", count(*) as "count" group by "操作类型" order by "count" desc
```

- **节点操作分布图表所关联的查询分析语句如下所示：**

```
* | select "verb" as "操作类型", count(*) as "count" where "objectRef.resource" = 'nodes' AND ("verb" in ('create','delete')) group by "操作类型" order by "count" desc
```

- **状态码分布图表所关联的查询分析语句如下所示：**

```
* | select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" group by "状态码" order by "count" desc
```

- **工作负载操作分布图表所关联的查询分析语句如下所示：**

```
* | select "verb" as "操作类型", count(*) as "count" where "verb" in ('create', 'delete') and "objectRef.resource" in ('deployments','statefulsets','daemonsets','jobs','cronjobs') group by "操作类型" order by "count" desc
```

- **Service/Ingress操作分布图表所关联的查询分析语句如下所示：**

```
* | select "verb" as "操作类型", count(*) as "count" where "verb" in ('create', 'delete') and "objectRef.resource" in ('ingresses','services') group by "verb" order by "count" desc
```

- **重要操作趋势图表所关联的查询分析语句如下所示：**

```
* | SELECT REPLACE(LEFT("stageTimestamp", 13), 'T', ' ') AS "dt", "verb", count(*) as "count" where "verb" in ('create','delete','update','patch') group by "dt", "verb" order by "dt" limit 10000
```

- **非系统用户操作趋势图表所关联的查询分析语句如下所示：**

```
* | SELECT REPLACE(LEFT("stageTimestamp", 13), 'T', ' ') AS "dt", count(*) as "count", "user.username" as "用户名称" where "user.username" not in ('kube-controller-manager','kube-apiserver-kubelet-client','system','apiserver') group by "dt", "用户名称" order by "dt" limit 10000
```

----结束

7.3.3.3 CDN 仪表盘模板

CDN（Content Delivery Network，内容分发网络）记录了所有域名（包括已删除域名，如果您开通了企业项目，则已删除域名不支持此功能）被网络用户访问的详细日志，您可以将日志接入LTS，对您的业务资源被访问情况进行详细分析。

CDN仪表盘模板支持[查看CDN错误分析](#)、[查看CDN基础数据](#)、[查看CDN用户分析](#)和[查看CDN热门资源](#)。

前提条件

- 已采集CDN日志，详情请参见[内容分发网络CDN接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CDN 错误分析

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“CDN仪表盘模板 > CDN错误分析”，查看图表详情。

- 错误域名访问Top5**图表所关联的查询分析语句如下所示：

```
* | select domain , count(*) as c where http_code > 400 group by domain order by c desc limit 5
```

- 错误URI访问Top5**图表所关联的查询分析语句如下所示：

```
* | select uri , count(*) as c where http_code > 400 group by uri order by c desc limit 5
```

- 请求错误百分比**图表所关联的查询分析语句如下所示：

```
* | select time_floor(__time,'PT5M') as axisData,sum(case when http_code >= 400 and http_code < 500 then 1 else 0 end) * 1.0 / count(*) as "4xx",sum(case when http_code >= 500 then 1 else 0 end) * 1.0 / count(*) as "5xx" group by axisData order by axisData
```

- 错误请求状态分布**图表所关联的查询分析语句如下所示：

```
* | select http_code , count(*) as c where http_code > 400 group by http_code order by c desc
```

- 错误按运营商统计**图表所关联的查询分析语句如下所示：

```
* | select ip_to_provider(client_ip) as isp , count(*) as c where http_code > 400 group by isp having ip_to_provider(client_ip) != "" order by c desc limit 10
```

- 错误按客户端统计**图表所关联的查询分析语句如下所示：

```
* | select user_agent as "客户端版本" , count(*) as "错误次数" where http_code > 400 group by user_agent order by "错误次数" desc limit 10
```

- 错误按省份统计**图表所关联的查询分析语句如下所示：

```
* | select ip_to_province(client_ip) as province , count(*) as c where http_code > 400 group by province order by c desc limit 50
```

- 4XX错误详情**图表所关联的查询分析语句如下所示：

```
* | SELECT province AS "省份", isp AS "运营商", c AS "错误次数", round( c * 100.0 / sum( c ), 2 ) AS "错误比率(%)" FROM ( SELECT ip_to_province ( client_ip ) AS province, ip_to_provider ( client_ip ) AS isp, count(*) AS c FROM log WHERE http_code >= 400 AND http_code < 500 GROUP BY province, isp HAVING ( ip_to_provider ( client_ip )) != "" ORDER BY c DESC ) GROUP BY province, isp, c
```

- 5XX错误详情**图表所关联的查询分析语句如下所示：

```
* | SELECT province AS "省份", isp AS "运营商", c AS "错误次数", round( c * 100.0 / sum( c ), 2 ) AS "错误比率(%)" FROM ( SELECT ip_to_province ( client_ip ) AS province, ip_to_provider ( client_ip ) AS isp, count(*) AS c FROM log WHERE http_code >= 500 GROUP BY province, isp HAVING ( ip_to_provider ( client_ip )) != "" ORDER BY c DESC ) GROUP BY province, isp, c
```

----结束

查看 CDN 基础数据

步骤1 在仪表盘模板下方，选择“CDN仪表盘模板 > CDN基础数据”，查看图表详情。

- 缓存命中率**图表所关联的查询分析语句如下所示：

```
* | select round(diff[1],2) as Hit_ratio, round(diff[2],2) as diff, round((diff[3]-1)*100, 2) from (select compare(Hit_ratio, 86400) as diff from (select sum(s) * 100.0/count(*) as Hit_ratio from (select case when hit_info = 'HIT' then 1 else 0 end as s from log)))
```

- 下载速度**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN diff [ 1 ] < 1024 THEN concat( cast( diff [ 1 ] AS decimal(10,2) ), ' KB/sec' ) WHEN diff [ 1 ] < 1024 * 1024 and diff [ 1 ] >= 1024 THEN concat( cast( diff [ 1 ] / 1024 AS decimal(10,2) ), ' MB/sec' ) WHEN diff [ 1 ] < 1024 * 1024 * 1024 and diff [ 1 ] >= 1024 * 1024 THEN concat( cast( diff [ 1 ] / 1024 * 1024.0 AS decimal(10,2) ), ' TB/sec' ) ELSE concat( cast( diff [ 1 ] / 1024.0 / 1024 / 1024 AS decimal(10,2) ), ' PB/sec' ) END AS "speed", round( diff [ 2 ], 2 ) AS diff, round(( diff [ 3 ] - 1 ) * 100, 2 ) as parallel_period FROM ( SELECT compare ( speed, 86400 ) AS diff FROM ( SELECT sum( cast( response_size AS BIGINT ) ) * 1.0 / sum( cast( response_time AS BIGINT ) ) AS speed FROM log ) )
```

- **访问状态图表所关联的查询分析语句如下所示：**

```
* | select http_code , count(*) as c group by http_code order by c desc
```

- **访问延时分布图表所关联的查询分析语句如下所示：**

```
* | select case when cast(response_time as INTEGER)< 100 then '1~100ms' when cast(response_time as INTEGER)< 500 and cast(response_time as INTEGER)>= 100 then '100~500ms' when cast(response_time as INTEGER)< 1000 and cast(response_time as INTEGER)>= 500 then '500ms~1s' when cast(response_time as INTEGER)< 5000 and cast(response_time as INTEGER)>= 1000 then '1~5s' when cast(response_time as INTEGER)< 6000 and cast(response_time as INTEGER)>= 5000 then '5~6s' when cast(response_time as INTEGER)< 7000 and cast(response_time as INTEGER)>= 6000 then '6~7s' when cast(response_time as INTEGER)< 8000 and cast(response_time as INTEGER)>= 7000 then '7~8s' when cast(response_time as INTEGER)< 10000 and cast(response_time as INTEGER)>= 8000 then '8~10s' when cast(response_time as INTEGER)< 15000 and cast(response_time as INTEGER)>= 10000 then '10~15s' else '15s~' end as latency , count(*) as cnt group by latency order by cnt
```

- **请求带宽图表所关联的查询分析语句如下所示：**

```
* | select TIME_FORMAT (TIME_FLOOR(__time,'PT1M'), 'HH:mm', '+08:00') as thisdate, sum(cast(response_size as bigint)) * 8/1000000000.0 as "带宽Gbit/min" group by TIME_FLOOR(__time,'PT1M') order by TIME_FLOOR(__time,'PT1M')
```

- **访问次数/人数图表所关联的查询分析语句如下所示：**

```
* | select TIME_FORMAT (TIME_FLOOR(__time,'PT1M'), 'HH:mm', '+08:00') as thisdate, count(*) as pv, APPROX_COUNT_DISTINCT(client_ip) as uv group by TIME_FLOOR(__time,'PT1M') order by TIME_FLOOR(__time,'PT1M')
```

- **访问平均延时图表所关联的查询分析语句如下所示：**

```
* | select TIME_FORMAT (TIME_FLOOR(__time,'PT1M'), 'HH:mm', '+08:00') as thisdate, avg(cast(response_time as bigint)) as "平均延时(ms)" group by TIME_FLOOR(__time,'PT1M') order by TIME_FLOOR(__time,'PT1M')
```

- **请求命中率图表所关联的查询分析语句如下所示：**

```
* | select TIME_FORMAT (TIME_FLOOR(m_time,'PT1M'), 'HH:mm', '+08:00') as thisdate , sum(is_hit)*100.0/count(*) as hit_ratio from (select TIME_FLOOR(__time,'PT1M') as m_time , case when hit_info = 'HIT' then 1 else 0 end as is_hit from log ) group by m_time order by m_time
```

----结束

查看 CDN 用户分析

步骤1 在仪表盘模板下方，选择“CDN仪表盘模板 > CDN用户分析”，查看图表详情。

- **访问次数图表所关联的查询分析语句如下所示：**

```
* | SELECT CASE WHEN diff [ 1 ] < 1000 THEN concat( cast( diff [ 1 ] AS decimal(10,2) ), ' 次' ) WHEN diff [ 1 ] < 1000 * 1000 THEN concat( cast( diff [ 1 ] / 1000 AS decimal(10,2) ), ' 千次' ) WHEN diff [ 1 ] < 1000000000 THEN concat( cast( diff [ 1 ] / 1000000.0 AS decimal(10,2) ), ' 百万次' ) WHEN diff [ 1 ] / 1000.0 < 1000000000 THEN concat( cast( diff [ 1 ] / 1000.0 / 1000000 AS decimal(10,2) ), ' 十亿次' ) ELSE concat( cast( diff [ 1 ] / 1000.0 / 1000 / 1000 / 1000 AS decimal(10,2) ), ' 万亿次' ) END AS "pv", diff [ 2 ] AS diff, round( 100 * ( diff [ 3 ] - 1 ) , 2 ) AS parallel_period FROM ( SELECT compare ( pv, 86400 ) AS diff FROM ( SELECT count(*) AS pv FROM log ) )
```

- **访问人数图表所关联的查询分析语句如下所示：**

```
* | select diff[1] as uv, diff[2] as diff, round((diff[3]-1)*100, 2) from (select compare(uv , 86400) as diff from (select APPROX_COUNT_DISTINCT(client_ip) as uv from log))
```

- **访问客户端统计图表所关联的查询分析语句如下所示：**

```
* | select ua as "客户端", sum(c) as "访问次数" from (select case when strpos(ua, 'iphone') > 1 then 'iphone' when strpos(ua, 'ipad') > 1 then 'ipad' when strpos(ua, 'android') > 1 then 'android' when strpos(ua, 'windows') > 1 then 'windows' when strpos(ua, 'mac') > 1 then 'mac' when strpos(ua, 'linux') > 1 then 'linux' else ua end as ua, c from (select count(*) as c, lower(user_agent) as ua from log group by ua order by c desc limit 2000) ) group by "客户端" order by "访问次数" desc limit 100
```

- **运营商次数统计图表所关联的查询分析语句如下所示：**

```
* | select ip_to_provider(client_ip) as isp ,count(*) as "访问次数" group by isp order by "访问次数" desc limit 100
```

- **访问地区分布图表所关联的查询分析语句如下所示：**

```
* | select ip_to_province(client_ip) as province , count(*) as cnt group by province order by cnt desc limit 100
```

- **有效访问用户TOP图表所关联的查询分析语句如下所示：**

```
* |  
SELECT  
CASE WHEN ip_to_country(client_ip)='香港' THEN concat(client_ip, ' ( Hong Kong )') WHEN  
ip_to_province(client_ip)='' THEN concat(client_ip, ' ( Unknown IP )') WHEN ip_to_provider(client_ip)='  
内网IP' THEN concat(client_ip, ' ( Private IP )') ELSE concat(client_ip, ' ( ', ip_to_country(client_ip), '/',  
ip_to_province(client_ip), '/',  
CASE WHEN ip_to_city(client_ip)='-1' THEN 'Unknown city' ELSE ip_to_city(client_ip) END,  
' ',ip_to_provider(client_ip), ' )' ) END AS "用户",  
pv as "总访问数",  
(pv - success_count) as "错误访问数",  
throughput as "下载总量(GB)"  
from (  
select  
client_ip ,  
count(*) as pv,  
round(sum(cast(response_size as bigint))/1024.0/1024/1024.0, 1) AS throughput ,  
sum(CASE WHEN http_code < 400 THEN 1 ELSE 0 END) AS success_count  
from log  
group by client_ip  
order by success_count desc  
limit 100)
```

- **下载量TOP用户图表所关联的查询分析语句如下所示：**

```
* | SELECT  
CASE WHEN ip_to_country(client_ip)='香港' THEN concat(client_ip, ' ( Hong Kong )') WHEN  
ip_to_province(client_ip)='' THEN concat(client_ip, ' ( Unknown IP )') WHEN ip_to_provider(client_ip)='  
内网IP' THEN concat(client_ip, ' ( Private IP )') ELSE concat(client_ip, ' ( ', ip_to_country(client_ip), '/',  
ip_to_province(client_ip), '/',  
CASE WHEN ip_to_city(client_ip)='-1' THEN 'Unknown city' ELSE ip_to_city(client_ip) END,  
' ',ip_to_provider(client_ip), ' )' ) END AS "用户",  
pv as "总访问数",  
error_count as "错误访问数",  
throughput as "下载总量(GB)" from (  
select client_ip ,  
count(*) as pv,  
round(sum(cast(response_size as bigint))/1024.0/1024/1024.0, 1) AS throughput ,  
sum(CASE WHEN http_code > 400 THEN 1 ELSE 0 END) AS error_count  
from log  
group by client_ip  
order by throughput desc  
limit 100)
```

----结束

查看 CDN 热门资源

步骤1 在仪表盘模板下方，选择“CDN仪表盘模板 > CDN热门资源”，查看图表详情。

- **域名访问次数Top5图表所关联的查询分析语句如下所示：**

```
* | select domain ,count(*) as cnt group by domain order by cnt desc limit 5
```

- **域名下载流量Top5图表所关联的查询分析语句如下所示：**

```
* | select domain , sum(cast(response_size as bigint)) as "下载总量" group by domain order by "下载总  
量" desc limit 5
```

- **热门访问（URI）** 图表所关联的查询分析语句如下所示：
* | select URI ,count(*) as "访问次数", APPROX_COUNT_DISTINCT(client_ip) as "访问人数", round(sum(cast(response_size as bigint))/1024.0/1024.0/1024.0, 2) as "下载总量(GB)" where http_code < 400 group by uri order by "访问次数" desc limit 100
- **热门访问（来源）** 图表所关联的查询分析语句如下所示：
* | select ip_to_province(client_ip) as province , count(*) as cnt group by province order by cnt desc limit 1000
- **全国访问次数分布统计** 图表所关联的查询分析语句如下所示：
* | select ip_to_province(client_ip) as province , count(*) as cnt group by province order by cnt desc limit 1000
- **全国下载网速统计** 图表所关联的查询分析语句如下所示：
* | select ip_to_province(client_ip) as province , sum(cast(response_size as bigint))* 1.0 / (sum(cast(response_time as bigint))+1) as speed , count(*) as c group by province order by c desc limit 40
- **省份统计** 图表所关联的查询分析语句如下所示：
* | select ip_to_province(client_ip) as "省份" ,count(*) as "访问次数", sum(cast(response_size as bigint))/1024.0/1024.0/1024.0 as "下载流量(GB)", sum(cast(response_size as bigint)) * 1.0 / sum(cast(response_time as bigint)) as "下载速度(KB/s)" group by "省份" having ip_to_province(client_ip) != " " order by "下载流量(GB)" desc limit 200
- **运营商流量和速度** 图表所关联的查询分析语句如下所示：
* | select ip_to_provider(client_ip) as isp , sum(cast(response_size as bigint))* 1.0 / (sum(cast(response_time as bigint))+1) as "下载速度(KB/s)", sum(cast(response_size as bigint))/1024.0/1024.0/1024.0 as "下载总量(GB)", count(*) as c group by isp having ip_to_provider(client_ip) != " " order by c desc limit 10
- **运营商统计** 图表所关联的查询分析语句如下所示：
* | select ip_to_provider(client_ip) as "运营商" ,count(*) as "访问次数", round(sum(cast(response_size as bigint))/1024.0/1024.0/1024.0,2) as "下载流量(GB)", round(sum(cast(response_size as bigint)) * 1.0 / sum(cast(response_time as bigint)) ,2) as "下载速度(KB/s)" group by "运营商" having ip_to_provider(client_ip) != " " order by "下载流量(GB)" desc limit 200

----结束

7.3.3.4 CFW 仪表盘模板

云防火墙（Cloud Firewall，CFW）是新一代的云原生防火墙，提供云上互联网边界和VPC边界的防护，包括实时入侵检测与防御、全局统一访问控制、全流量分析可视化、日志审计与溯源分析等，同时支持按需弹性扩容、AI提升智能防御能力、灵活扩展满足云上业务的变化和扩张需求，极简应用让用户快速灵活应对威胁。云防火墙可以通过攻击事件日志查看检测到的危险流量的危险等级、受影响的端口、命中的规则、攻击事件类型等信息；通过访问控制日志查看根据访问控制策略放行或阻断的所有流量，以便更好的调整访问控制策略。

CFW仪表盘模板支持[查看CFW访问日志中心](#)、[查看CFW流量日志中心](#)和[查看CFW攻击日志中心](#)。

前提条件

- 已采集CFW日志，详情请参见[云防火墙CFW接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CFW 访问日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“日志管理”。

步骤3 在“日志应用”模块中，单击“CFW日志中心”，选择“进入仪表盘”。

步骤4 在仪表盘模板下方，选择“CFW仪表盘模板 > CFW访问日志中心”，查看图表详情。

- **互联网访问-拦截趋势**图展示互联网访问-拦截趋势的变化情况，所关联的查询分析语句如下所示：

```
* | select time_series(MILLIS_TO_TIMESTAMP(hit_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' group by t_time order by t_time limit 100000
```
- **主动外联-拦截趋势**图展示主动外联-拦截趋势的变化情况，所关联的查询分析语句如下所示：

```
* | select time_series(MILLIS_TO_TIMESTAMP(hit_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' group by t_time order by t_time limit 100000
```
- **互联网阻断应用TOP5**图展示互联网阻断应用TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT app, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' GROUP BY app ORDER BY frequency DESC LIMIT 5
```
- **互联网阻断目的TOP5**图展示互联网阻断目的TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT dst_ip, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' GROUP BY dst_ip ORDER BY frequency DESC LIMIT 5
```
- **互联网阻断来源TOP5**图展示互联网阻断来源TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT src_ip, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' GROUP BY src_ip ORDER BY frequency DESC LIMIT 5
```
- **主动外联阻断应用TOP5**图展示主动外联阻断应用TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT app, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' GROUP BY app ORDER BY frequency DESC LIMIT 5
```
- **主动外联阻断目的TOP5**图展示主动外联阻断目的TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT dst_ip, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' GROUP BY dst_ip ORDER BY frequency DESC LIMIT 5
```
- **主动外联阻断来源TOP5**图展示主动外联阻断来源TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT src_ip, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' GROUP BY src_ip ORDER BY frequency DESC LIMIT 5
```

----结束

查看 CFW 流量日志中心

步骤1 在“日志应用”模块中，单击“CFW日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“CFW仪表盘模板 > CFW流量日志中心”，查看图表详情。

- **互联网访问流量趋势**图展示互联网访问流量趋势的变化情况，所关联的查询分析语句如下所示：

```
* | select time_series(MILLIS_TO_TIMESTAMP(start_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, SUM(to_s_bytes) AS '入流量', SUM(to_c_bytes) AS '出流量' WHERE direction='out2in' group by t_time order by t_time limit 100000
```
- **互联网访问流入地域分布(中国)**图展示互联网访问流入地域分布(中国)的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT count(*) AS PV, ip_to_province(src_ip) AS province WHERE direction='out2in' and IP_TO_COUNTRY (src_ip) = '中国' GROUP BY province HAVING province not in ('', '保留地址', '*') ORDER BY PV DESC
```
- **互联网访问流入地域分布(世界)**图展示互联网访问流入地域分布(世界)的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT count(*) AS PV, ip_to_country(src_ip) AS country WHERE direction='out2in' GROUP BY country HAVING country not in ('','保留地址','*') ORDER BY PV DESC
```

- **互联网访问应用分布图**展示互联网访问应用分布的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT app, COUNT(*) AS num WHERE direction='out2in' GROUP BY app ORDER BY num DESC
```
- **互联网访问源IP TOP5图**展示互联网访问源IP TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | select src_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='out2in' GROUP BY src_ip ORDER BY sum_bytes DESC LIMIT 5
```
- **互联网访问目的IP TOP5图**展示互联网访问目的IP TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | select dst_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='out2in' GROUP BY dst_ip ORDER BY sum_bytes DESC LIMIT 5
```
- **主动外联流量趋势图**展示主动外联流量趋势的变化情况，所关联的查询分析语句如下所示：

```
* | select time_series(MILLIS_TO_TIMESTAMP(start_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, SUM(to_c_bytes) AS '入流量', SUM(to_s_bytes) AS '出流量' WHERE direction='in2out' group by t_time order by t_time limit 100000
```
- **主动外联目的地域分布(中国)图**展示主动外联目的地域分布(中国)的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT count(*) AS PV, ip_to_province(dst_ip) AS province WHERE direction='in2out' and IP_TO_COUNTRY (dst_ip) = '中国' GROUP BY province HAVING province not in ('','保留地址','*') ORDER BY PV DESC
```
- **目的地域分布(世界)图**展示目的地域分布(世界)的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT count(*) AS PV, ip_to_country(dst_ip) AS country WHERE direction='in2out' GROUP BY country HAVING country not in ('','保留地址','*') ORDER BY PV DESC
```
- **主动外联-应用分布图**展示主动外联-应用分布的变化情况，所关联的查询分析语句如下所示：

```
* | SELECT app, COUNT(*) AS num WHERE direction='in2out' GROUP BY app ORDER BY num DESC
```
- **主动外联源IP TOP5图**展示主动外联源IP TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | select src_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='in2out' GROUP BY src_ip ORDER BY sum_bytes DESC LIMIT 5
```
- **主动外联目的IP TOP5图**展示主动外联目的IP TOP5的变化情况，所关联的查询分析语句如下所示：

```
* | select dst_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='in2out' GROUP BY dst_ip ORDER BY sum_bytes DESC LIMIT 5
```

----结束

查看 CFW 攻击日志中心

步骤1 在“日志应用”模块中，单击“CFW日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“CFW仪表盘模板 > CFW攻击日志中心”，查看图表详情。

- **攻击趋势图**所关联的查询分析语句如下所示：

```
* | select time_series(MILLIS_TO_TIMESTAMP(event_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, count(*) as frequency group by t_time order by t_time limit 100000
```
- **攻击来源分布(中国)图**表所关联的查询分析语句如下所示：

```
* | SELECT count(*) as PV,ip_to_province(src_ip) as province WHERE IP_TO_COUNTRY (src_ip) = '中国' GROUP BY province HAVING province not in ('','保留地址','*')
```
- **攻击来源分布(世界)图**表所关联的查询分析语句如下所示：

```
* | SELECT count(*) AS PV,ip_to_country(src_ip) AS country GROUP BY country HAVING country not in ('','保留地址','*')
```


- **攻击类型分布**图表所关联的查询分析语句如下所示：

```
* | SELECT attack_type, COUNT(*) as num GROUP BY attack_type ORDER BY num
```

- **攻击目的TOP5**图表所关联的查询分析语句如下所示：

```
* | SELECT dst_ip, COUNT(*) as frequency GROUP BY dst_ip ORDER BY frequency DESC LIMIT 5
```

- **攻击来源TOP5**图表所关联的查询分析语句如下所示：

```
* | SELECT src_ip, COUNT(*) as frequency GROUP BY src_ip ORDER BY frequency DESC LIMIT 5
```

----结束

7.3.3.5 CSE 仪表盘模板

微服务引擎（Cloud Service Engine，CSE）是用于微服务应用的云中间件。用户可结合其他云服务，快速构建云原生微服务体系，实现微服务应用的快速开发和高可用运维。

CSE仪表盘模板支持[查看CSE层访问中心](#)、[查看CSE层监控中心](#)和[查看CSE层秒级监控](#)。

前提条件

- 已采集CSE日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CSE 层访问中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“CSE仪表盘模板 > CSE层访问中心”，查看图表详情。

- **过滤上游IP**，所关联的查询分析语句如下所示：

```
* | select distinct(upstream_host)
```

- **过滤调用链trace_id**，所关联的查询分析语句如下所示：

```
* | select distinct(trace_id)
```

- **PV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv", 86400) as diff from (select count(1) as "pv" from log))
```

- **PV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv", 604800) as diff from (select count(1) as "pv" from log))
```

- **UV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv", 86400) as diff from (select APPROX_COUNT_DISTINCT(authority) as "uv" from log))
```

- **UV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv", 604800) as diff from (select APPROX_COUNT_DISTINCT(authority) as "uv" from log))
```

- **访问量PV分布(中国)**图表所关联的查询分析语句如下所示：

```
* | select ip_to_province(authority) as province, sum(ori_pv) as pv from (select authority, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) where IP_TO_COUNTRY (authority) = '中国' group by province HAVING province not in ('', '保留地址', '*')
```

- **访问量PV分布(世界)**图表所关联的查询分析语句如下所示：

```
* | SELECT ip_to_country(authority) as country, sum(ori_pv) as PV from (select authority, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) GROUP BY country HAVING country not in ('', '保留地址', '*')
```


- **平均时延分布（中国）** 图表所关联的查询分析语句如下所示：

```
* | SELECT province, round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" FROM (SELECT ip_to_province(authority) as province, sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select authority, sum(duration) as rt, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) WHERE IP_TO_COUNTRY (authority) = '中国' GROUP BY province ) where province not in ('','保留地址','*')
```

- **平均时延分布（世界）** 图表所关联的查询分析语句如下所示：

```
* | SELECT country, round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)" FROM (SELECT ip_to_country(authority) as country, sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select authority, sum(duration) as rt, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) GROUP BY country ) where country not in ('','保留地址','*')
```

- **今日PV/UV** 图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss') as _time_ , PV, UV FROM (select TIME_CEIL(TIME_PARSE(start_time, 'yyyy-MM-dd"T"HH:mm:ss.SSS"Z")', 'PT600S') AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(authority) as UV from log WHERE _time_ >= DATE_TRUNC( 'DAY', CURRENT_TIMESTAMP) group by _time_ order by _time_ ) order by _time_ limit 10000
```

- **区域访问TOP10(省份)** 图表所关联的查询分析语句如下所示：

```
* | select ip_to_province(authority) as "province", sum(ori_pv) as "访问次数" from (select authority, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) group by "province" HAVING "province" <> '-1' order by "访问次数" desc limit 10
```

- **区域访问TOP10(城市)** 图表所关联的查询分析语句如下所示：

```
* | select ip_to_city(authority) as "city", sum(ori_pv) as "访问次数" from (select authority, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) group by "city" HAVING "city" <> '-1' order by "访问次数" desc limit 10
```

- **Host访问TOP10** 图表所关联的查询分析语句如下所示：

```
* | select upstream_host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```

- **UserAgent访问TOP10** 图表所关联的查询分析语句如下所示：

```
* | select user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc limit 10
```

- **设备占比(终端)** 图表所关联的查询分析语句如下所示：

```
* | select case when regexp_like(lower(user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type , count(1) as total group by type
```

- **设备占比(系统)** 图表所关联的查询分析语句如下所示：

```
* | select case when regexp_like(lower(user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total group by type HAVING type != 'other'
```

- **TOP URL** 图表所关联的查询分析语句如下所示：

```
* | select path , count(1) as pv, APPROX_COUNT_DISTINCT(authority) as UV, round(sum( case when response_code < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group by path ORDER BY pv desc
```

- **TOP 访问IP** 图表所关联的查询分析语句如下所示：

```
* | select authority as "来源IP", ip_to_country(authority) as "国家", ip_to_province(authority) as "省份", ip_to_city(authority) as "城市", ip_to_provider(authority) as "运营商", count(1) as "PV" group by authority ORDER BY "PV" desc limit 100
```

----结束

查看 CSE 层监控中心

步骤1 在仪表盘模板下方，选择“CSE仪表盘模板 > CSE层监控中心”，查看图表详情。

- **过滤上游IP**，所关联的查询分析语句如下所示：

```
* | select distinct(upstream_host)
```

- **过滤调用链trace_id**，所关联的查询分析语句如下所示：

```
* | select distinct(trace_id)
```

- **访问量PV** 图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss') as _time_ , PV FROM ( SELECT TIME_CEIL  
( TIME_PARSE(start_time,'yyyy-MM-dd"T"HH:mm:ss.SSS"Z"') , 'PT300S' ) AS _time_ , count( 1 ) AS PV  
FROM log GROUP BY _time_ ) order by _time_
```

- **请求成功率**图表所关联的查询分析语句如下所示:

```
* | select ROUND(sum(case when response_code < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```

- **平均延迟**图表所关联的查询分析语句如下所示:

```
* | select round(avg(duration) * 1000, 3) as cnt
```

- **4XX请求数**图表所关联的查询分析语句如下所示:

```
* | SELECT COUNT(1) as cnt WHERE "response_code" >= 400 and "response_code" < 500
```

- **404请求数**图表所关联的查询分析语句如下所示:

```
* | SELECT COUNT(1) as cnt WHERE "response_code" = 404
```

- **429请求数**图表所关联的查询分析语句如下所示:

```
* | SELECT COUNT(1) as cnt WHERE "response_code" = 429
```

- **504请求数**图表所关联的查询分析语句如下所示:

```
* | SELECT COUNT(1) as cnt WHERE "response_code" = 504
```

- **5XX请求数**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss') as _time_ , cnt FROM ( SELECT TIME_CEIL  
( TIME_PARSE(start_time,'yyyy-MM-dd"T"HH:mm:ss.SSS"Z"') , 'PT300S' ) AS _time_ , count( 1 ) AS cnt  
FROM log where "response_code" >= 500 GROUP BY _time_ ) order by _time_
```

- **状态码分布**图表所关联的查询分析语句如下所示:

```
* | SELECT response_code, COUNT(1) AS rm GROUP BY response_code
```

- **访问量UV**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss') as _time_ , UV FROM (select  
TIME_CEIL(TIME_PARSE(start_time,'yyyy-MM-dd"T"HH:mm:ss.SSS"Z"') , 'PT600S') AS _time_ ,  
APPROX_COUNT_DISTINCT(authority) as UV from log group by _time_ ) order by _time_
```

- **流量**图表所关联的查询分析语句如下所示:

```
* | select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss') AS _time_ , round( CASE WHEN "入流量" > 0  
THEN "入流量" ELSE 0 END, 2 ) AS "入流量" , round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0  
END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(TIME_PARSE(start_time,'yyyy-MM-  
dd"T"HH:mm:ss.SSS"Z"') , 'PT600S') AS _time_ , sum(bytes_received) / 1024.0 AS "入流量"  
 , sum(bytes_sent) / 1024.0 AS "出流量" group by _time_ ) order by _time_
```

- **访问失败率**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss') as _time_ , round( CASE WHEN "失败率" >  
0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率" , round( CASE WHEN "5XX比例" > 0 THEN "5XX比例"  
ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(TIME_PARSE(start_time,'yyyy-MM-  
dd"T"HH:mm:ss.SSS"Z"') , 'PT600S') AS _time_ , sum(case when response_code >= 400 then 1 else 0  
end) * 100.0 / count(1) as '失败率' , sum(case when response_code >= 500 THEN 1 ELSE 0 END) * 100.0 /  
COUNT(1) as '5XX比例' group by _time_ ) order by _time_
```

- **延迟**图表所关联的查询分析语句如下所示:

```
* | select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss') as _time_ , round( CASE WHEN "平均" > 0  
THEN "平均" ELSE 0 END, 2 ) AS "平均" , round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 )  
AS "P50" , round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90" , round( CASE WHEN  
"P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99" , round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE  
0 END, 2 ) AS "P9999" from (select TIME_CEIL(TIME_PARSE(start_time,'yyyy-MM-  
dd"T"HH:mm:ss.SSS"Z"') , 'PT600S') as _time_ , avg(duration) * 1000 as "平均",  
APPROX_QUANTILE_DS("duration", 0.50) * 1000 as "P50", APPROX_QUANTILE_DS("duration",  
0.90) * 1000 as "P90", APPROX_QUANTILE_DS("duration", 0.99) * 1000 as  
"P99", APPROX_QUANTILE_DS("duration", 0.9999) * 1000 as "P9999" group by _time_ ) order by _time_
```

- **Host请求TOP**图表所关联的查询分析语句如下所示:

```
* | SELECT "upstream_host" , pv , uv , round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)"  
ELSE 0 END, 2 ) AS "访问成功率(%)" , round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)"  
ELSE 0 END, 3 ) AS "平均延迟(ms)" , round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0  
END, 3 ) AS "入流量(KB)" , round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 )  
AS "出流量(KB)" FROM ( SELECT "upstream_host" , count( 1 ) AS pv , APPROX_COUNT_DISTINCT  
( authority ) AS uv , sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 /  
count( 1 ) AS "访问成功率(%)" , avg( duration ) * 1000 AS "平均延迟(ms)" , sum( bytes_received ) /  
1024.0 AS "入流量(KB)" , sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "upstream_host" != ""  
GROUP BY "upstream_host" ) ORDER BY pv DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "upstream_host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "upstream_host", count( 1 ) AS pv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " " GROUP BY "upstream_host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "upstream_host", pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "upstream_host", count( 1 ) AS pv, sum( CASE WHEN "response_code" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " " GROUP BY "upstream_host" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT path, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT path, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( authority ) AS uv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", sum( bytes_received ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "upstream_host" != " " GROUP BY path ) ORDER BY pv desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT path, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT path, count( 1 ) AS pv, sum( CASE WHEN "response_code" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " " GROUP BY path ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT authority as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( authority ) AS uv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", sum( bytes_received ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "upstream_host" != " " GROUP BY addr having length(authority) > 2 ) ORDER BY "pv" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT authority as addr, count( 1 ) AS pv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " " and "authority" != '-' GROUP BY addr ) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)"
```

```
延迟(ms)" FROM ( SELECT authority as addr, count( 1 ) AS pv, sum( CASE WHEN "response_code" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " and "authority" != '-' GROUP BY addr) ORDER BY "访问失败率(%)" desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT path, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT path, count( 1 ) AS pv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " GROUP BY path ) ORDER BY "平均延迟(ms)" desc
```

----结束

查看 CSE 层秒级监控

步骤1 在仪表盘模板下方，选择“CSE仪表盘模板 > CSE层秒级监控”，查看图表详情。

- 过滤上游IP，所关联的查询分析语句如下所示：

```
* | select distinct(upstream_host)
```

- 过滤调用链trace_id，所关联的查询分析语句如下所示：

```
* | select distinct(trace_id)
```

- **QPS**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time,'yyyy-MM-dd"T"HH:mm:ss.SSS"Z"'),'PT5S'),'yyyy-MM-dd HH:mm:ss') AS _time_, COUNT(*) as QPS from log group by _time_ order by _time_ limit 10000
```

- **成功率**图表所关联的查询分析语句如下所示：

```
* | select _time_, round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time,'yyyy-MM-dd"T"HH:mm:ss.SSS"Z"'),'PT5S'),'yyyy-MM-dd HH:mm:ss') as _time_, sum(case when response_code < 400 then 1 else 0 end) * 100.0 / count(1) as "成功率" from log group by _time_) order by _time_ limit 10000
```

- **延迟**图表所关联的查询分析语句如下所示：

```
* | select _time_, round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟", round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time,'yyyy-MM-dd"T"HH:mm:ss.SSS"Z"'),'PT5S'),'yyyy-MM-dd HH:mm:ss') as _time_, avg(duration)* 1000 as '访问延迟', avg(upstream_service_time)* 1000 as 'Upstream延迟' from log group by _time_) order by _time_ limit 10000
```

- **流量**图表所关联的查询分析语句如下所示：

```
* | select _time_, round( CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 3 ) AS "请求流量", round( CASE WHEN "返回body流量" > 0 THEN "返回body流量" ELSE 0 END, 3 ) AS "返回body流量" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time,'yyyy-MM-dd"T"HH:mm:ss.SSS"Z"'),'PT5S'),'yyyy-MM-dd HH:mm:ss') as _time_, sum("bytes_received") / 1024.0 as "请求流量", sum("bytes_sent") / 1024.0 as "返回body流量" group by _time_) order by _time_ limit 10000
```

- **状态码**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_CEIL ( TIME_PARSE ( start_time , 'yyyy-MM-dd"T"HH:mm:ss.SSS"Z"') , 'PT5S' ) AS "time", SUM( CASE WHEN "response_code" >= 200 AND "response_code" < 300 THEN 1 ELSE 0 END ) AS "2XX", SUM( CASE WHEN "response_code" >= 300 AND "response_code" < 400 THEN 1 ELSE 0 END ) AS "3XX", SUM( CASE WHEN "response_code" >= 400 AND "response_code" < 500 THEN 1 ELSE 0 END ) AS "4XX", SUM( CASE WHEN "response_code" >= 500 AND "response_code" < 600 THEN 1 ELSE 0 END ) AS "5XX", SUM( CASE WHEN "response_code" < 200 OR "response_code" >= 600 THEN 1 ELSE 0 END ) AS "其他" FROM log WHERE TIME_PARSE ( start_time , 'yyyy-MM-dd"T"HH:mm:ss.SSS"Z"') IS NOT NULL GROUP BY "time" ORDER BY "time" ASC LIMIT 100000
```

----结束

7.3.3.6 CoreDNS 仪表盘模板

CoreDNS作为集群内的DNS服务器，您可以通过CoreDNS仪表盘模板查看CoreDNS日志来分析CoreDNS解析慢、访问高危请求域名等问题。

前提条件

- 已采集CoreDNS日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CoreDNS 日志分析

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“CoreDNS仪表盘模板 > CoreDNS日志分析”，查看图表详情。

- 过滤域名，所关联的查询分析语句如下所示：
`* | select name group by name limit 10000`
- 过滤客户端IP，所关联的查询分析语句如下所示：
`* | select remote group by remote limit 10000`
- 过滤状态码，所关联的查询分析语句如下所示：
`* | select rcode group by rcode limit 10000`
- 总请求次数图表所关联的查询分析语句如下所示：
`* | select diff[1] as nowCount, diff[1] - diff[2] as delta from (select compare(total, 86400) as diff from( select count(1) as total from log ) )`
- NXDOMAIN次图表所关联的查询分析语句如下所示
`(* and rcode: NXDOMAIN) | select diff[1] as nowCount, diff[1] - diff[2] as delta from (select compare(total, 86400) as diff from( select count(1) as total from log ) )`
- 请求成功率图表所关联的查询分析语句如下所示：
`* | select diff[1] as nowCount, round(diff[1] - diff[2], 2) as delta from (select compare(total, 86400) as diff from( select round( count_if(rcode = 'NOERROR' or rcode = 'NXDOMAIN') * 100.0 / (case when count(1) = 0 then 1 else count(1) end), 2 ) as total from log ) )`
- 域名数图表所关联的查询分析语句如下所示：
`* | select diff[1] as nowCount, diff[1] - diff[2] as delta from (select compare(total, 86400) as diff from( select approx_distinct(name) as total from log ) )`
- 平均延迟图表所关联的查询分析语句如下所示：
`* | select diff[1] as nowCount, round(diff[1] - diff[2], 3) as delta from (select compare(total, 86400) as diff from( select round(avg(duration * 1000), 3) as total from log ) )`
- Timeout次数图表所关联的查询分析语句如下所示：
`(level : ERROR) | select diff[1] as nowCount, round(diff[1] - diff[2], 3) as delta from (select compare(total, 86400) as diff from( select count(1) as "total" from log where errmsg like '%timeout%' ) )`
- P95延迟图表所关联的查询分析语句如下所示：
`* | select diff[1] as nowCount, round(diff[1] - diff[2], 3) as delta from (select compare(total, 86400) as diff from( select round(approx_percentile(duration * 1000, 0.95), 3) as total from log ) )`
- P99延迟图表所关联的查询分析语句如下所示：
`* | select diff[1] as nowCount, round(diff[1] - diff[2], 3) as delta from (select compare(total, 86400) as diff from( select round(approx_percentile(duration * 1000, 0.99), 3) as total from log ) )`
- 请求QPS图表所关联的查询分析语句如下所示：
`* | select date_format(t, '%Y-%m-%d %H:%i') as time , d[1] as "今天", d[2] as "昨天", d[3] as "上周" from(select t, ts_compare(pv, 86400, 604800) as d from(select from_unixtime(__time - __time % 60000) as t, round(count(1)/ 60.0, 2) as pv from log group by t order by t) group by t) where date_format(t, '%Y-%m-%d %H:%i') is not null order by time limit 10000`

- 请求平均延迟(ms)图表所关联的查询分析语句如下所示：

```
*| select date_format(t, '%Y-%m-%d %H:%i') as time, d[1] as "今天", d[2] as "昨天", d[3] as "上周"
from( select t, ts_compare(pv, 86400,604800 ) as d from(select from_unixtime(__time - __time %
60000) as t, round(avg(duration * 1000.0), 3 ) as pv from log where level = 'INFO' group by t order by
t ) group by t ) where date_format(t, '%Y-%m-%d %H:%i') is not null order by time limit 10000
```
- 成功率(%)图表所关联的查询分析语句如下所示：

```
*| select date_format(t, '%Y-%m-%d %H:%i') as time, d[1] as "今天", d[2] as "昨天", d[3] as "上周"
from( select t, ts_compare(pv, 86400,604800 ) as d from(select from_unixtime(__time - __time %
60000) as t, round( count_if(rcode = 'NOERROR') * 100.0 / (case when count(1) = 0 then 1 else
count(1) end), 2 ) as pv from log group by t order by t ) group by t ) where date_format(t, '%Y-%m-
%d %H:%i') is not null order by time limit 10000
```
- P99延迟(ms)图表所关联的查询分析语句如下所示：

```
*| select date_format(t, '%Y-%m-%d %H:%i') as time, d[1] as "今天", d[2] as "昨天", d[3] as "上周"
from( select t, ts_compare(pv, 86400,604800 ) as d from(select from_unixtime(__time - __time %
60000) as t, round(approx_percentile(duration * 1000.0, 0.99), 3 ) as pv from log where level = 'INFO'
group by t order by t ) group by t ) where date_format(t, '%Y-%m-%d %H:%i') is not null order by
time limit 10000
```
- Top请求域名图表所关联的查询分析语句如下所示：

```
*| SELECT name, COUNT(*) as total where name is not null group by name order by total desc
```
- 状态码分布图表所关联的查询分析语句如下所示：

```
*| SELECT rcode, COUNT(*) as total group by rcode order by total desc
```
- TOP域名图表所关联的查询分析语句如下所示：

```
*| SELECT name as "域名", rcode as "状态码", round(sum(size/ 1024.0), 3) as "请求流量KB",
round(sum(rsize / 1024.0),3 ) as "返回流量KB", COUNT(*) as "请求次数", round(avg(duration * 1000),
3) as "平均延迟(ms)" where name is not null group by name, rcode order by "请求次数" desc
```
- TOP NXDOMAIN域名图表所关联的查询分析语句如下所示：

```
* and rcode: NXDOMAIN| SELECT name as "域名", rcode as "状态码", round(sum(size/ 1024.0), 3) as
"请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB", COUNT(*) as "请求次数",
round(avg(duration * 1000), 3) as "平均延迟(ms)" group by name, rcode order by "请求次数" desc
```
- TOP 错误域名 (非NXDOMAIN) 图表所关联的查询分析语句如下所示：

```
(* not rcode: NXDOMAIN not rcode : NOERROR)| SELECT name as "域名", rcode as "状态码",
round(sum(size/ 1024.0), 3) as "请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB",
COUNT(*) as "请求次数", round(avg(duration * 1000), 3) as "平均延迟(ms)" where name is not null
group by name, rcode order by "请求次数" desc
```
- TOP请求Pod图表所关联的查询分析语句如下所示：

```
*| SELECT remote as "IP", arbitrary(name) as "请求域名采样", round(sum(size/ 1024.0), 3) as "请求流量
KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB", COUNT(*) as "请求次数", round(avg(duration *
1000), 3) as "平均延迟(ms)" where remote is not null group by remote order by "请求次数" desc
```
- TOP NXDOMAIN Pod图表所关联的查询分析语句如下所示：

```
(* and rcode : NXDOMAIN)| SELECT remote as "IP", arbitrary(name) as "请求域名采样",
round(sum(size/ 1024.0), 3) as "请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB",
COUNT(*) as "请求次数", round(avg(duration * 1000), 3) as "平均延迟(ms)" where remote is not null
group by remote order by "请求次数" desc
```
- TOP错误Pod (非NXDOMAIN) 图表所关联的查询分析语句如下所示：

```
(* not rcode : NXDOMAIN not rcode : NOERROR)| SELECT remote as "IP", arbitrary(name) as "请求域
名采样", round(sum(size/ 1024.0), 3) as "请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量
KB", COUNT(*) as "请求次数", round(avg(duration * 1000), 3) as "平均延迟(ms)" where remote is not
null group by remote order by "请求次数" desc
```
- Timeout域名图表所关联的查询分析语句如下所示：

```
(level : ERROR)| select name as "域名", count(1) as "次数" from log where errmsg like '%timeout%'
group by name order by "次数" desc limit 100
```
- 近100条请求APIServier的错误图表所关联的查询分析语句如下所示：

```
(content : reflector.go)| select date_format(from_unixtime(__time), '%Y-%m-%d %H:%i:%S') as "时间"
, errmsg as "错误" from log order by __time desc limit 100
```
- 解析日志图表所关联的查询分析语句如下所示：

```
(* and level: INFO and duration > 0.010)| SELECT name as "域名", round( duration * 1000.0, 3) as "延迟
(ms)", remote as "请求者", port as "端口", type as "类型", rcode as "结果" from log order by duration
desc limit 100
```

----结束

7.3.3.7 DCS 仪表盘模板

分布式缓存服务（Distributed Cache Service，简称DCS）是一款内存数据库服务，兼容了Redis内存数据库引擎，满足用户高并发及数据快速访问的业务诉求。云日志服务提供对各种云资源操作记录的收集、存储和查询功能，可用于支撑安全分析、合规审计、资源跟踪和问题定位等常见应用场景。

云日志服务支持日志采集向导一站式采集DCS日志，DCS审计日志中心仪表盘支持展示访问用户数、访问客户端数、审计日志条数等图表。

前提条件

- 已采集DCS日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

DCS 审计日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“DCS仪表盘模板 > DCS审计日志中心”，查看图表详情。

- **访问用户数**图表所关联的查询分析语句如下所示：
* | select count(distinct(user)) as user_num
- **访问客户端数**图表所关联的查询分析语句如下所示：
* | select count(distinct(client_addr)) as client_num
- **审计日志条数**图表所关联的查询分析语句如下所示：
* | select count(1) as log_num
- **平均响应时间**图表所关联的查询分析语句如下所示：
* | select avg(use_time) as avg_time
- **平均QPS**图表所关联的查询分析语句如下所示：
* | select count(*) / CAST((TIMESTAMPDIFF (minute, MIN(_time),MAX(_time))+1) as FLOAT)
- **TOP5 用户**图表所关联的查询分析语句如下所示：
* | select user, count(1) as 'user_count' group by user order by count(1) desc LIMIT 5
- **TOP5 客户端**图表所关联的查询分析语句如下所示：
* | select client_addr, count(1) as 'remote_count' group by client_addr order by count(1) desc limit 5
- **TOP5 执行命令**图表所关联的查询分析语句如下所示：
* | select command_name, count(1) as 'command_count' group by command_name order by count(1) desc
- **热Key**图表所关联的查询分析语句如下所示：
* | SELECT count(*) as 'count', command_keys GROUP by command_keys order by count(*) desc limit 5
- **热Key详情**图表所关联的查询分析语句如下所示：
* | SELECT server_addr as "服务节点地址", CAST(db AS VARCHAR) as 'DB', command_type as "数据类型", CAST(count(*) AS VARCHAR) as '访问频次', command_keys GROUP by command_keys, db, server_addr, command_type order by count(*) desc
- **审计日志详情**图表所关联的查询分析语句如下所示：
* | select "client_addr" as '客户端IP',"client_type" as '客户端类型',"server_addr" as '服务端IP',"command_name" as '执行命令',"command_keys" as '命令KEYS',"command_param" as '命令内容',"command_type" as '命令类型',"use_time" as '执行耗时',"time" as '执行时间',"db" as 'DB',"user" as '账号名',"instance_id" as '实例ID',"role" as '节点角色',"extend" as '扩展信息'

----结束

7.3.3.8 DDS 仪表盘模板

文档数据库服务（Document Database Service）完全兼容MongoDB协议，提供安全、高可用、高可靠、弹性伸缩和易用的数据库服务，同时提供一键部署、弹性扩容、容灾、备份、恢复、监控和告警等功能。云日志服务进行分析日志、搜索日志、日志可视化、下载日志和查看实时日志等操作。

云日志服务支持日志采集向导一站式采集DDS日志，DDS审计日志中心展示审计日志条数、访问用户数、访问客户端数等图表。

前提条件

- 已采集DDS日志，详情请参见[文档数据库服务DDS接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

DDS 审计日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“DDS仪表盘模板 > DDS审计日志中心”，查看图表详情。

- 过滤操作类型，所关联的查询分析语句如下所示：
`* | select distinct(optype)`
- 过滤客户端IP，所关联的查询分析语句如下所示：
`* | select distinct(user_ip)`
- 过滤数据库名，所关联的查询分析语句如下所示：
`* | select distinct(db)`
- 过滤用户名，所关联的查询分析语句如下所示：
`* | select distinct(user)`
- 过滤集合名，所关联的查询分析语句如下所示：
`* | select distinct(coll)`
- 审计日志条数图表所关联的查询分析语句如下所示：
`* | select count(1) as log_num`
- 访问用户数图表所关联的查询分析语句如下所示：
`* | select count(distinct(user)) as user_num`
- 访问客户端数图表所关联的查询分析语句如下所示：
`* | select count(distinct(user_ip)) as client_num`
- TOP5 执行命令图表所关联的查询分析语句如下所示：
`* | select optype, count(1) as 'command_count' group by optype order by count(1) desc LIMIT 5`
- TOP5 用户图表所关联的查询分析语句如下所示：
`* | select user, count(1) as 'user_count' group by user order by count(1) desc LIMIT 5`
- TOP5 客户端图表所关联的查询分析语句如下所示：
`* | select user_ip, count(1) as 'remote_count' group by user_ip order by count(1) desc LIMIT 5`
- 审计日志详情图表所关联的查询分析语句如下所示：
`* | select "time" as "执行时间","user" as "用户名","param" as "查询语句","instanceid" as "实例ID","db" as "DB","coll" as "集合名","user_ip" as "客户端IP"`

----结束

7.3.3.9 DMS 仪表盘模板

分布式消息服务Kafka版（Distributed Message Service for Kafka）是一款基于开源社区版Kafka提供的消息队列服务，向用户提供计算、存储和带宽资源独占式的Kafka

专享实例。重平衡日志记录Rebalance的详情，包括Rebalance时间、原因和触发Rebalance的客户端等。

云日志服务支持日志采集向导一站式采集DMS重平衡日志，支持多维度分析，并为DMS-Rebalance日志配置结构化和仪表盘。该仪表盘主要展示DMS重平衡日志的重平衡消费组个数、重平衡次数、消费组重平衡次数、重平衡原因及组详情。

前提条件

- 已采集DMS日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

DMS 重平衡日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“DMS仪表盘模板 > DMS重平衡日志中心”，查看图表详情。

- 消费组ID，所关联的查询分析语句如下所示：

```
* | select distinct("message.groupId")
```
- 过滤重平衡原因。
- 重平衡消费组个数图表所关联的查询分析语句如下所示：

```
* | select count(distinct("message.groupId")) as "total" from log where ("message.type" != 'RESPONSE' and "message.type" != 'REQUEST')
```
- 重平衡次数图表所关联的查询分析语句如下所示：

```
* | select count(*) as 'total' where ("message.type" != 'RESPONSE' and "message.type" != 'REQUEST')
```
- 消费组重平衡次数图表所关联的查询分析语句如下所示：

```
* | select "message.groupId" as 'GroupId', count(*) as 'Count' where ('message.type' != 'RESPONSE' and 'message.type' != 'REQUEST') group by "message.groupId"
```
- 重平衡原因及组详情图表所关联的查询分析语句如下所示：

```
* | select __time as 'Time', "message.type" as 'Type', "message.groupId" as 'GroupId', "message.reason" as 'Reason', "message.group" as 'Group' where ('message.type' != 'RESPONSE' and 'message.type' != 'REQUEST')
```

----结束

7.3.3.10 DSL 仪表盘模板

DSL加工是LTS为您提供的一站式日志加工平台，基于领域自定义的脚本语言和200多个内置函数，您可以在LTS控制台实现端到端的日志规整、富化、流转、脱敏、过滤等加工任务。

云日志服务支持DSL（Domain Specific Language）仪表盘模板，DSL加工任务监控中心主要展示加工任务ID、加工任务名称、输入行数、输出行数等信息。

前提条件

- 已创建DSL加工任务。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

DSL 加工任务监控中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“DSL仪表盘模板 > DSL加工任务监控中心”，查看图表详情。

- 过滤加工任务ID，所关联的查询分析语句如下所示：

```
* | select distinct(task_id)
```

- 过滤加工任务名称，所关联的查询分析语句如下所示：

```
* | select distinct(task_name)
```

- 输入行数图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "input" < 1000 THEN concat( cast( "input" AS VARCHAR ), '行' ) WHEN "input" < 1000 * 1000 THEN concat( cast( round( "input"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "input" < 1000000000 THEN concat( cast( round( "input"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "input"/ 1000.0 < 1000000000 THEN concat( cast( round( "input"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "input"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("process.accept") as "input")
```

- 输出行数图表所关联的查询分析语句如下所示

```
* | SELECT CASE WHEN "delivered" < 1000 THEN concat( cast( "delivered" AS VARCHAR ), '行' ) WHEN "delivered" < 1000 * 1000 THEN concat( cast( round( "delivered"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "delivered" < 1000000000 THEN concat( cast( round( "delivered"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "delivered"/ 1000.0 < 1000000000 THEN concat( cast( round( "delivered"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "delivered"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("process.delivered") as "delivered")
```

- 过滤行数图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "drop" < 1000 THEN concat( cast( "drop" AS VARCHAR ), '行' ) WHEN "drop" < 1000 * 1000 THEN concat( cast( round( "drop"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "drop" < 1000000000 THEN concat( cast( round( "drop"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "drop"/ 1000.0 < 1000000000 THEN concat( cast( round( "drop"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "drop"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("process.drop") as "drop")
```

- 失败行数图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "failed" < 1000 THEN concat( cast( "failed" AS VARCHAR ), '行' ) WHEN "failed" < 1000 * 1000 THEN concat( cast( round( "failed"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "failed" < 1000000000 THEN concat( cast( round( "failed"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "failed"/ 1000.0 < 1000000000 THEN concat( cast( round( "failed"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "failed"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("process.failed") as "failed")
```

- 执行记录图表所关联的查询分析语句如下所示：

```
* | select MILLIS_TO_TIMESTAMP("start") as "统计开始时间", MILLIS_TO_TIMESTAMP("end") as "统计结束时间", "process.accept" as "输入行数", "process.delivered" as "输出行数", "process.drop" as "过滤行数", "process.failed" as "失败行数" limit 1000
```

----结束

7.3.3.11 ELB 仪表盘模板（共享型）

弹性负载均衡（Elastic Load Balance，简称ELB）是将访问流量根据分配策略分发到后端多台服务器的流量分发控制服务。弹性负载均衡可以通过流量分发扩展应用系统对外的服务能力，同时通过消除单点故障提升应用系统的可用性，支持查看和分析对七层负载均衡HTTP和HTTPS进行请求的详细访问日志记录，包括请求时间、客户端IP地址、请求路径和服务器响应等。

ELB仪表盘模板支持[查看ELB7层访问中心](#)、[查看ELB7层监控中心](#)和[查看ELB7层秒级监控](#)。

- upstream_addr_priv (该参数适用于共享型负载均衡)：后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。
- upstream_addr (该参数适用于独享型负载均衡)：后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。

更多信息请参考[独享型负载均衡与共享型弹性负载均衡的区别](#)。

前提条件

- 已采集ELB日志，详情请参见[弹性负载均衡ELB接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 ELB7 层访问中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“日志管理”。

步骤3 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤4 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层访问中心”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
* | select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
* | select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
* | select distinct(upstream_addr_priv)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
* | select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **PV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv", 86400) as diff from (select count(1) as "pv" from log))
```

- **PV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv", 604800) as diff from (select count(1) as "pv" from log))
```

- **UV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv", 86400) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log))
```

- **UV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select  
compare( "uv" , 604800) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from  
log))
```

- **访问量PV分布(中国)**图表所关联的查询分析语句如下所示:

```
* | select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY  
(remote_addr) = '中国' group by province HAVING province not in ('','保留地址','')
```

- **访问量PV分布(世界)**图表所关联的查询分析语句如下所示:

```
* | SELECT ip_to_country(remote_addr) as country, COUNT(1) as PV GROUP BY country HAVING  
country not in ('','保留地址','')
```

- **访问量UV分布(中国)**图表所关联的查询分析语句如下所示:

```
* | select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as uv  
where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址',  
'')
```

- **访问量UV分布(世界)**图表所关联的查询分析语句如下所示:

```
* | select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv  
group by country HAVING country not in ('','保留地址','')
```

- **平均时延分布(中国)**图表所关联的查询分析语句如下所示:

```
* | SELECT province, round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS  
"平均延迟(ms)" FROM (SELECT ip_to_province(remote_addr) as province, avg(request_time) * 1000 AS  
"平均延迟(ms)" WHERE IP_TO_COUNTRY (remote_addr) = '中国' GROUP BY province HAVING province  
not in ('','保留地址',''))
```

- **平均时延分布(世界)**图表所关联的查询分析语句如下所示:

```
* | SELECT country, round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS  
"平均延迟(ms)" FROM (SELECT ip_to_country(remote_addr) as country, avg(request_time) * 1000 AS  
"平均延迟(ms)" GROUP BY country HAVING country not in ('','保留地址',''))
```

- **今日PV/UV**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , PV, UV FROM (select  
TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), PT600S) AS _time_ , count(1)  
as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE  
TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ') <= CURRENT_TIMESTAMP and  
TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ') >= DATE_TRUNC( 'DAY',  
(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR group by _time_ order by  
_time_)
```

- **7日PV/UV**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , PV, UV FROM (select  
TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), PT600S) AS _time_ , count(1)  
as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE  
TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ') <= CURRENT_TIMESTAMP and  
TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ') >= DATE_TRUNC( 'DAY',  
(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR - INTERVAL '7' DAY group by  
_time_ order by _time_)
```

- **区域访问TOP10(省份)**图表所关联的查询分析语句如下所示:

```
* | select ip_to_province(remote_addr) as "province", count(1) as "访问次数" group by "province"  
HAVING "province" <> '-1' order by "访问次数" desc limit 10
```

- **区域访问TOP10(城市)**图表所关联的查询分析语句如下所示:

```
* | select ip_to_city(remote_addr) as "city", count(1) as "访问次数" group by "city" HAVING "city" <>  
'-1' order by "访问次数" desc limit 10
```

- **Host访问TOP10**图表所关联的查询分析语句如下所示:

```
* | select host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```

- **UserAgent访问TOP10**图表所关联的查询分析语句如下所示:

```
* | select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc  
limit 10
```

- **设备占比(终端)**图表所关联的查询分析语句如下所示:

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else  
'PC端' end as type , count(1) as total group by type
```

- **设备占比(系统)**图表所关联的查询分析语句如下所示:

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when  
regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as  
total group by type HAVING type != 'other'
```

- **TOP URL**图表所关联的查询分析语句如下所示：

```
* | select CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN LPAD(router_request_uri,
CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE router_request_uri END as
"router_request_uri" , count(1) as pv, APPROX_COUNT_DISTINCT(remote_addr) as uv,
round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group
by "router_request_uri" ORDER by pv desc
```

- **TOP 访问IP**图表所关联的查询分析语句如下所示：

```
* | select remote_addr as "来源IP",ip_to_country(remote_addr) as "国家",ip_to_province(remote_addr)
as "省份",ip_to_city(remote_addr) as "城市",ip_to_provider(remote_addr) as "运营商",count(1) as
"PV",http_user_agent as "UserAgent采样",CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN
LPAD(router_request_uri, CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE
router_request_uri END as "URL采样" group by remote_addr,http_user_agent,"URL采样" ORDER by
"PV" desc
```

----结束

查看 ELB7 层监控中心

步骤1 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层监控中心”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
* | select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
* | select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
* | select distinct(upstream_addr_priv)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
* | select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **访问量PV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , PV FROM (select
TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS _time_ , count(1)
as PV from log group by _time_ order by _time_ ) order by _time_
```

- **请求成功率**图表所关联的查询分析语句如下所示：

```
* | select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```

- **4XX请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
```

- **499请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 499
```

- **平均延迟**图表所关联的查询分析语句如下所示：

```
* | select round(avg(request_time) * 1000, 3) as cnt
```

- **404请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 404
```

- **504请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 504
```

- **5XX请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_, cnt FROM ( SELECT TIME_CEIL ( TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT300S' ) AS _time_ , count( 1 ) AS cnt FROM log where "status" >= 500 GROUP BY _time_ ) order by _time_
```

- **状态码分布**图表所关联的查询分析语句如下所示：

```
* | SELECT status, COUNT(1) AS rm GROUP BY status
```

- **访问量UV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , UV FROM ( select TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S' ) AS _time_ , APPROX_COUNT_DISTINCT(remote_addr) as UV from log group by _time_ ) order by _time_
```

- **流量**图表所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) AS _time_ , round( CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量" , round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2 ) AS "出流量" FROM ( SELECT TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S' ) AS _time_ , sum(request_length) / 1024.0 AS "入流量" , sum(bytes_sent) / 1024.0 AS "出流量" group by _time_ ) order by _time_
```

- **访问失败率**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , round( CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率" , round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2 ) AS "5XX比例" from ( select TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S' ) AS _time_ , sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率' , sum(case when status >= 500 then 1 else 0 end) * 100.0 / COUNT(1) as '5XX比例' group by _time_ ) order by _time_
```

- **延迟**图表所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , round( CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均" , round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 ) AS "P50" , round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90" , round( CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99" , round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2 ) AS "P9999" from ( select TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S' ) as _time_ , avg(request_time) * 1000 as "平均" , APPROX_QUANTILE_DS("request_time", 0.50) * 1000 as "P50" , APPROX_QUANTILE_DS("request_time", 0.90) * 1000 as "P90" , APPROX_QUANTILE_DS("request_time", 0.99) * 1000 as "P99" , APPROX_QUANTILE_DS("request_time", 0.9999) * 1000 as "P9999" group by _time_ ) order by _time_
```

- **Host请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT host, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)" , round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" , round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)" , round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)" , avg( request_time ) * 1000 AS "平均延迟(ms)" , sum( request_length ) / 1024.0 AS "入流量(KB)" , sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY "host" ) ORDER BY pv DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)" , round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" , round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)" , round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)" , avg( request_time ) * 1000 AS "平均延迟(ms)" , APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)" , APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY "host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)" , round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" , round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)" , round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE
```

```
0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"host" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP图表所关联的查询分析语句如下所示：**

```
* | SELECT router_request_uri, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)"  
ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)"  
ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0  
END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 )  
AS "出流量(KB)" FROM ( SELECT CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN  
LPAD(router_request_uri, CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE  
router_request_uri END as "router_request_uri", count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 /  
count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)"  
WHERE "host" != " " GROUP BY "router_request_uri" ) ORDER BY pv desc
```

- **URL延迟TOP图表所关联的查询分析语句如下所示：**

```
* | SELECT router_request_uri, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)"  
ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)"  
ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)"  
ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0  
END, 3 ) AS "P99延迟(ms)" FROM (SELECT CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN  
LPAD(router_request_uri, CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE  
router_request_uri END as "router_request_uri", count( 1 ) AS pv, sum( CASE WHEN "status" < 400  
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延  
迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"router_request_uri" ) ORDER BY "平均延迟(ms)" desc
```

- **URL失败率TOP图表所关联的查询分析语句如下所示：**

```
* | SELECT router_request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)"  
ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)"  
ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)"  
ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE  
0 END, 3 ) AS "P99延迟(ms)" FROM( SELECT CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN  
LPAD(router_request_uri, CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE  
router_request_uri END as "router_request_uri", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400  
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延  
迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"router_request_uri" ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP图表所关联的查询分析语句如下所示：**

```
* | SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,  
2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,  
3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS  
"入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量  
(KB)" FROM ( SELECT upstream_addr_priv as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 /  
count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)"  
WHERE "host" != " " GROUP BY addr having length(upstream_addr_priv) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP图表所关联的查询分析语句如下所示：**

```
* | SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )  
AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM (SELECT upstream_addr_priv as addr,count( 1 ) AS pv,sum( CASE WHEN "status" <  
400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平  
均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟  
(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != "  
GROUP BY addr having length(upstream_addr_priv) > 2) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP图表所关联的查询分析语句如下所示：**

```
* | SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 )  
AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99
```

```
延迟(ms)" FROM ( SELECT upstream_addr_priv as addr, count( 1 ) AS pv, sum( CASE WHEN "status"
>= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS
"平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " GROUP BY
addr having length(upstream_addr_priv) > 2) ORDER BY "访问失败率(%)" desc
```

----结束

查看 ELB7 层秒级监控

步骤1 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层秒级监控”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- QPS图展示QPS的变化情况，所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT1S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') AS __time_, COUNT(*) as QPS from log
group by __time_
```

- 成功率图展示成功率的变化情况，所关联的查询分析语句如下所示：

```
* | select __time, round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end, 2) as "成功率" from (select
TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') as __time, sum(case when status < 400 then 1 else 0 end) * 100.0 /
count(1) as '成功率' from log group by __time)
```

- 延迟图展示延迟的变化情况，所关联的查询分析语句如下所示：

```
* | select __time, round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end, 2) as "访问延迟",
round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end, 2) as "Upstream延迟"
from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') as __time, avg(request_time) * 1000 as
'访问延迟', avg(upstream_response_time) * 1000 as 'Upstream延迟' from log group by __time)
```

- 流量图展示流量的变化情况，所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(__time, 'yyyy-MM-dd HH:mm:ss', '+08:00') AS __time, round(CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 2) AS "请求流量", round(CASE WHEN "返回body流量" > 0
THEN "返回body流量" ELSE 0 END, 2) AS "返回body流量" from (select
TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as __time,
sum("request_length") / 1024.0 as "请求流量", sum("body_bytes_sent") / 1024.0 as "返回body流量"
group by __time )
```

- 状态码图展示状态码的变化情况，所关联的查询分析语句如下所示：


```
* | select TIME_FORMAT(t.t, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "time", CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX", CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX", CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX", CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX", CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他" from (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t from log group by t order by t asc ) t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "status" >= 200 and "status" < 300 group by t order by t asc ) a on t.t = a.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "status" >= 300 and "status" < 400 group by t order by t asc ) b on t.t = b.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "status" >= 400 and "status" < 500 group by t order by t asc ) c on t.t = c.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "status" >= 500 and "status" < 600 group by t order by t asc ) d on t.t = d.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "status" < 200 or "status" >= 600 group by t order by t asc ) e on t.t = e.t
```

- 后端响应码图展示后端响应码的变化情况，所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(t.t, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "time", CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX", CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX", CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX", CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX", CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他" from (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t from log group by t order by t asc ) t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "upstream_status" >= 200 and "upstream_status" < 300 group by t order by t asc ) a on t.t = a.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "upstream_status" >= 300 and "upstream_status" < 400 group by t order by t asc ) b on t.t = b.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "upstream_status" >= 400 and "upstream_status" < 500 group by t order by t asc ) c on t.t = c.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "upstream_status" >= 500 and "upstream_status" < 600 group by t order by t asc ) d on t.t = d.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "upstream_status" < 200 or "upstream_status" >= 600 group by t order by t asc ) e on t.t = e.t
```

----结束

7.3.3.12 ELB 仪表盘模板（独享型）

弹性负载均衡（Elastic Load Balance，简称ELB）是将访问流量根据分配策略分发到后端多台服务器的流量分发控制服务。弹性负载均衡可以通过流量分发扩展应用系统对外的服务能力，同时通过消除单点故障提升应用系统的可用性，支持查看和分析对七层负载均衡HTTP和HTTPS进行请求的详细访问日志记录，包括请求时间、客户端IP地址、请求路径和服务器响应等。

ELB仪表盘模板支持[查看ELB7层访问中心](#)、[查看ELB7层监控中心](#)和[查看ELB7层秒级监控](#)。

- upstream_addr_priv（该参数适用于共享型负载均衡）：后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。
- upstream_addr（该参数适用于独享型负载均衡）：后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。

更多信息请参考[独享型负载均衡与共享型弹性负载均衡的区别](#)。

前提条件

- 已采集ELB日志，详情请参见[弹性负载均衡ELB接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 ELB7 层访问中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“日志管理”。

步骤3 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤4 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层访问中心”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
* | select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
* | select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
* | select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
* | select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- PV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv", 86400) as diff from (select count(1) as "pv" from log))
```

- PV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv", 604800) as diff from (select count(1) as "pv" from log))
```

- UV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv", 86400) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log))
```

- UV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv", 604800) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log))
```

- 访问量PV分布(中国)**图表所关联的查询分析语句如下所示：

```
* | select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY(remote_addr) = '中国' group by province HAVING province not in ('','保留地址','*')
```

- 访问量PV分布(世界)**图表所关联的查询分析语句如下所示：

```
* | SELECT ip_to_country(remote_addr) as country,COUNT(1) as PV GROUP BY country HAVING country not in ('','保留地址','*')
```

- **访问量UV分布(中国)**图表所关联的查询分析语句如下所示:

```
* | select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as uv
where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','*')
```
- **访问量UV分布(世界)**图表所关联的查询分析语句如下所示:

```
* | select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv
group by country HAVING country not in ('','保留地址','*')
```
- **平均时延分布(中国)**图表所关联的查询分析语句如下所示:

```
* | SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS
"平均延迟(ms)"FROM (SELECT ip_to_province(remote_addr) as province,avg(request_time) * 1000 AS
"平均延迟(ms)"WHERE IP_TO_COUNTRY (remote_addr) = '中国'GROUP BY province HAVING province
not in ('','保留地址','*'))
```
- **平均时延分布(世界)**图表所关联的查询分析语句如下所示:

```
* | SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS
"平均延迟(ms)"FROM (SELECT ip_to_country(remote_addr) as country,avg(request_time) * 1000 AS
"平均延迟(ms)" GROUP BY country HAVING country not in ('','保留地址','*'))
```
- **今日PV/UV**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select
TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS _time_ , count(1)
as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE
TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ') <= CURRENT_TIMESTAMP and
TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ') >= DATE_TRUNC( 'DAY',
(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR group by _time_ order by
_time_)
```
- **7日PV/UV**图表所关联的查询分析语句如下所示:

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select
TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS _time_ , count(1)
as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE
TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ') <= CURRENT_TIMESTAMP and
TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ') >= DATE_TRUNC( 'DAY',
(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR - INTERVAL '7' DAY group by
_time_ order by _time_)
```
- **区域访问TOP10(省份)**图表所关联的查询分析语句如下所示:

```
* | select ip_to_province(remote_addr) as "province", count(1) as "访问次数" group by "province"
HAVING "province" <> '-1' order by "访问次数" desc limit 10
```
- **区域访问TOP10(城市)**图表所关联的查询分析语句如下所示:

```
* | select ip_to_city(remote_addr) as "city", count(1) as "访问次数" group by "city" HAVING "city" <>
'-1' order by "访问次数" desc limit 10
```
- **Host访问TOP10**图表所关联的查询分析语句如下所示:

```
* | select host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```
- **UserAgent访问TOP10**图表所关联的查询分析语句如下所示:

```
* | select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc
limit 10
```
- **设备占比(终端)**图表所关联的查询分析语句如下所示:

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else
'PC端' end as type , count(1) as total group by type
```
- **设备占比(系统)**图表所关联的查询分析语句如下所示:

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when
regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as
total group by type HAVING type != 'other'
```
- **TOP URL**图表所关联的查询分析语句如下所示:

```
* | select CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN LPAD(router_request_uri,
CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE router_request_uri END as
"router_request_uri" , count(1) as pv, APPROX_COUNT_DISTINCT(remote_addr) as uv,
round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group
by "router_request_uri" ORDER by pv desc
```
- **TOP 访问IP**图表所关联的查询分析语句如下所示:

```
* | select remote_addr as "来源IP",ip_to_country(remote_addr) as "国家",ip_to_province(remote_addr)
as "省份",ip_to_city(remote_addr) as "城市",ip_to_provider(remote_addr) as "运营商",count(1) as
```

```
"PV",http_user_agent as "UserAgent采样",CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN  
LPAD(router_request_uri, CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE  
router_request_uri END as "URL采样" group by remote_addr,http_user_agent,"URL采样" ORDER by  
"PV" desc
```

----结束

查看 ELB7 层监控中心

步骤1 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层监控中心”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **访问量PV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , PV FROM (select  
TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ' ) , 'PT600S' ) AS _time_ , count(1)  
as PV from log group by _time_ order by _time_ ) order by _time_
```

- **请求成功率**图表所关联的查询分析语句如下所示：

```
* | select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```

- **4XX请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
```

- **499请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 499
```

- **平均延迟**图表所关联的查询分析语句如下所示：

```
* | select round(avg(request_time) * 1000, 3) as cnt
```

- **404请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 404
```

- **504请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 504
```

- **5XX请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , cnt FROM ( SELECT  
TIME_CEIL ( TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ' ) , 'PT300S' ) AS _time_ ,  
count( 1 ) AS cnt FROM log where "status" >= 500 GROUP BY _time_ ) order by _time_
```

- **状态码分布**图表所关联的查询分析语句如下所示：

```
* | SELECT status, COUNT(1) AS rm GROUP BY status
```

- **访问量UV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , UV FROM (select TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS _time_ , APPROX_COUNT_DISTINCT(remote_addr) as UV from log group by _time_ ) order by _time_
```

- **流量**图表所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) AS _time_ , round( CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量" , round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS _time_ , sum(request_length) / 1024.0 AS "入流量" , sum(bytes_sent) / 1024.0 AS "出流量" group by _time_ ) order by _time_
```

- **访问失败率**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , round( CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率" , round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS _time_ , sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率' , sum(case when status >= 500 THEN 1 ELSE 0 END) * 100.0 / COUNT(1) as '5XX比例' group by _time_ ) order by _time_
```

- **延迟**图表所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , round( CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均" , round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 ) AS "P50" , round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90" , round( CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99" , round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(TIME_PARSE(time_iso8601 , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') as _time_ , avg(request_time) * 1000 as "平均" , APPROX_QUANTILE_DS("request_time", 0.50) * 1000 as "P50" , APPROX_QUANTILE_DS("request_time", 0.90) * 1000 as "P90" , APPROX_QUANTILE_DS("request_time", 0.99) * 1000 as "P99" , APPROX_QUANTILE_DS("request_time", 0.9999) * 1000 as "P9999" group by _time_ ) order by _time_
```

- **Host请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT host, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)" , round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" , round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)" , round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)" , avg( request_time ) * 1000 AS "平均延迟(ms)" , sum( request_length ) / 1024.0 AS "入流量(KB)" , sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != "" GROUP BY "host" ) ORDER BY pv DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)" , round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" , round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)" , round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)" , avg( request_time ) * 1000 AS "平均延迟(ms)" , APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)" , APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != "" GROUP BY "host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)" , round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" , round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)" , round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)" , avg( request_time ) * 1000 AS "平均延迟(ms)" , APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)" , APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != "" GROUP BY "host" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT router_request_uri, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)" , round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" , round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)" , round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN
```

```
LPAD(router_request_uri, CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE  
router_request_uri END as "router_request_uri", count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 /  
count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)"  
WHERE "host" != " " GROUP BY "router_request_uri" ) ORDER BY pv desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示:

```
* | SELECT router_request_uri, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)"  
ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)"  
ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE  
0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0  
END, 3 ) AS "P99延迟(ms)" FROM( SELECT CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN  
LPAD(router_request_uri, CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE  
router_request_uri END as "router_request_uri", count( 1 ) AS pv, sum( CASE WHEN "status" < 400  
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延  
迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"router_request_uri" ) ORDER BY "平均延迟(ms)" desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示:

```
* | SELECT router_request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)"  
ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)"  
ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)"  
ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE  
0 END, 3 ) AS "P99延迟(ms)" FROM( SELECT CASE WHEN STRPOS(router_request_uri, '?')!=0 THEN  
LPAD(router_request_uri, CAST(STRPOS(router_request_uri, '?') as INTEGER)-1, '?') ELSE  
router_request_uri END as "router_request_uri", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400  
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延  
迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"router_request_uri" ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示:

```
* | SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,  
2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,  
3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS  
"入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量  
(KB)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 /  
count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)"  
WHERE "host" != " " GROUP BY addr having length(upstream_addr) > 2 ) ORDER BY "pv" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示:

```
* | SELECT addr, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )  
AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" < 400  
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延  
迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟  
(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " "  
GROUP BY addr having length(upstream_addr) > 2 ) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示:

```
* | SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 )  
AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >=  
400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平  
均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
addr having length(upstream_addr) > 2 ) ORDER BY "访问失败率(%)" desc
```

----结束

查看 ELB7 层秒级监控

步骤1 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层秒级监控”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
* | select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
* | select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
* | select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
* | select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **QPS图**展示QPS的变化情况，所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT1S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') AS _time_, COUNT(*) as QPS from log group by _time_
```

- **成功率图**展示成功率的变化情况，所关联的查询分析语句如下所示：

```
* | select __time, round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end, 2) as "成功率" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') as __time, sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group by __time)
```

- **延迟图**展示延迟的变化情况，所关联的查询分析语句如下所示：

```
* | select __time, round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end, 2) as "访问延迟", round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end, 2) as "Upstream延迟" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') as __time, avg(request_time) * 1000 as '访问延迟', avg(upstream_response_time) * 1000 as 'Upstream延迟' from log group by __time)
```

- **流量图**展示流量的变化情况，所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(__time, 'yyyy-MM-dd HH:mm:ss', '+08:00') AS __time, round(CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 2) AS "请求流量", round(CASE WHEN "返回body流量" > 0 THEN "返回body流量" ELSE 0 END, 2) AS "返回body流量" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') as __time, sum("request_length") / 1024.0 as "请求流量", sum("body_bytes_sent") / 1024.0 as "返回body流量" group by __time)
```

- **状态码图**展示状态码的变化情况，所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(t, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "time", CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX", CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX", CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX", CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX", CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他" from (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t from log group by t order by t asc) t left join (select TIME_CEIL(TIME_PARSE(time_iso8601, 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "status" >= 200 and "status" < 300
```



```
group by t order by t asc ) a on t.t = a.t left join ( select
TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S' ) as t,
CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "status" >= 300 and "status" < 400
group by t order by t asc ) b on t.t = b.t left join (select
TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S' ) as t,
CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "status" >= 400 and "status" < 500
group by t order by t asc ) c on t.t = c.t left join ( select
TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S' ) as t,
CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "status" >= 500 and "status" < 600
group by t order by t asc ) d on t.t = d.t left join ( select
TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S' ) as t,
CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "status" < 200 or "status" >= 600
group by t order by t asc ) e on t.t = e.t
```

- 后端响应码图展示后端响应码的变化情况，所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(t.t,'yyyy-MM-dd HH:mm:ss','+08:00') as "time",CASE WHEN a."2XX" IS NOT
NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",CASE WHEN b."3XX" IS NOT NULL THEN
CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",CASE WHEN c."4XX" IS NOT NULL THEN
CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",CASE WHEN d."5XX" IS NOT NULL THEN
CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其
他" AS BIGINT) ELSE 0 END as "其他" from (select TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-MM-
dd"T"HH:mm:ssZZ'),'PT5S') as t from log group by t order by t asc) t left join(select
TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-MM-dd"T"HH:mm:ssZZ'),'PT5S') as t , CAST(COUNT(1)
as VARCHAR) as "2XX" from log WHERE "upstream_status" >= 200 and "upstream_status" < 300
group by t order by t asc) a on t.t = a.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-
MM-dd"T"HH:mm:ssZZ'),'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE
"upstream_status" >= 300 and "upstream_status" < 400 group by t order by t asc) b on t.t =b.t left
join (select TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-MM-dd"T"HH:mm:ssZZ'),'PT5S') as t ,
CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "upstream_status" >= 400 and
"upstream_status" < 500 group by t order by t asc) c on t.t =c.t left join (select
TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-MM-dd"T"HH:mm:ssZZ'),'PT5S') as t , CAST(COUNT(1)
as VARCHAR) as "5XX" from log WHERE "upstream_status" >= 500 and "upstream_status" < 600
group by t order by t asc) d on t.t =d.t left join (select TIME_CEIL(TIME_PARSE(time_iso8601,'yyyy-
MM-dd"T"HH:mm:ssZZ'),'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE
"upstream_status" < 200 or "upstream_status" >= 600 group by t order by t asc) e on t.t =e.t
```

----结束

7.3.3.13 ER 仪表盘模板

企业路由器（Enterprise Router, ER）可以连接虚拟私有云（Virtual Private Cloud, VPC）或本地网络来构建中心辐射型组网，是云上大规格、高带宽、高性能的集中路由器。企业路由器使用边界网关协议（Border Gateway Protocol, BGP），支持路由学习、动态选路以及链路切换，极大的提升网络的可扩展性及运维效率，从而保证业务的连续性。

云日志服务支持日志采集向导一站式采集ER日志，支持多维度分析，并为ER日志配置结构化和仪表盘。该仪表盘主要展示ER日志的TOP20包数统计、TOP20流量统计、流日志条数等信息。

前提条件

- 已采集ER流量日志，详情请参见[企业路由器ER接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

ER 流量日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“ER仪表盘模板 > ER流量日志中心”，查看图表详情。

- 过滤实例ID图，所关联的查询分析语句如下所示：
* | SELECT DISTINCT(instance_id)
- 过滤连接ID图，所关联的查询分析语句如下所示：
* | SELECT DISTINCT(resource_id)
- 流量方向图展示为静态过滤器，可按照入方向和出方向的流量进行过滤。
- 过滤源IP图，所关联的查询分析语句如下所示：
* | SELECT DISTINCT(srcaddr)
- 过滤目的IP图，所关联的查询分析语句如下所示：
* | SELECT DISTINCT(dstaddr)
- 过滤协议类型图，所关联的查询分析语句如下所示：
* | SELECT DISTINCT(protocol)
- **TOP20包数统计**图表所关联的查询分析语句如下所示：
* | SELECT "srcaddr" as "源地址", "dstaddr" as "目的地址", sum("packets") as "包数", "resource_id" as "连接ID", "instance_id" as "实例ID" group by "instance_id", "resource_id", "srcaddr", "dstaddr" order by "包数" desc limit 20
- **TOP20流量统计**图表所关联的查询分析语句如下所示：
* | SELECT "srcaddr" as "源地址", "dstaddr" as "目的地址", sum("bytes") as "字节数", "resource_id" as "连接ID", "instance_id" as "实例ID" group by "instance_id", "resource_id", "srcaddr", "dstaddr" order by "字节数" desc limit 20
- **流日志条数**图表所关联的查询分析语句如下所示：
* | select time_series(_time, 'PT1H', 'yyyy-MM-dd HH:mm:ss', '0', '+08:00') as "时间", count(*) as "流日志条数" group by "时间" order by "时间"
- **流日志详情**图表所关联的查询分析语句如下所示：
* | SELECT "instance_id" as "实例ID", "resource_id" as "连接ID", "project_id" as "项目ID", "srcaddr" as "源IP", "dstaddr" as "目的IP", "srcport" as "源端口", "dstport" as "目的端口", "protocol" as "协议类型", "direct" as "流量方向", "packets" as "包数", "bytes" as "字节数", TIME_FORMAT(MILLIS_TO_TIMESTAMP("start"*1000), 'yyyy-MM-dd HH:mm:ss', '+08:00') as "开始时间", TIME_FORMAT(MILLIS_TO_TIMESTAMP("end"*1000), 'yyyy-MM-dd HH:mm:ss', '+08:00') as "结束时间"

----结束

7.3.3.14 LTS 仪表盘模板

LTS仪表盘模板主要介绍日志生成指标任务监控中心、OBS导入任务监控仪表盘、资源用量明细仪表盘、用量Top排序仪表盘、DSL加工任务监控中心。

- **日志生成指标任务监控中心**：用户在LTS页面只需按照业务需要创建指标规则即可生成自己的统计报表，设置单个日志过滤条件或通过添加关联关系和添加组设置多个日志过滤条件，保留符合条件的日志，对用户特定时间范围内已结构化的日志进行动态统计，并将统计结果动态呈现到aom的Prometheus实例，操作简单且功能强大。
- **OBS导入任务监控仪表盘**：支持查看OBS导入LTS的任务详情，例如任务ID、任务名称、成功条数、失败条数、处理速率等关键指标，帮助用户实时监控数据导入状态，快速定位异常问题，确保数据高效、完整导入LTS。
- **资源用量明细仪表盘**：需要开通“可观测For LTS”功能，详细操作请参考[设置LTS日志采集配额和使用量预警](#)。支持按时间范围、日志组ID或日志流ID筛选，查看日志流的资源用量明细（如原始日志流量、索引流量、存储量等），帮助用户快速识别资源消耗情况，及时发现异常流量，优化资源配置，提升资源利用率。
- **用量Top排序仪表盘**：需要开通“可观测For LTS”功能，详细操作请参考[设置LTS日志采集配额和使用量预警](#)。支持按时间范围、日志组ID或日志流ID筛选，查看TOP N日志组或日志流的资源用量（如原始日志流量、索引流量、存储量等），帮助用户快速识别高资源消耗情况，及时发现异常流量，优化日志采集策略与资源配置，降低无效成本。

- **DSL加工任务监控中心**：DSL加工是LTS为您提供的一站式日志加工平台，基于领域自定义的脚本语言和200多个内置函数，您可以在LTS控制台实现端到端的日志规整、富化、流转、脱敏、过滤等加工任务。DSL加工任务监控中心主要展示加工任务ID、加工任务名称、输入行数、输出行数等信息。

说明

请先开通管道符搜索方式才能使用OBS导入任务监控仪表盘、资源用量明细仪表盘、用量Top排序仪表盘。如有需要请[提交工单](#)申请开通。

日志生成指标任务监控中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“METRIC仪表盘模板 > 日志生成指标任务监控中心”，查看图表详情。

- 过滤规则ID，所关联的查询分析语句如下所示：

```
* | select distinct(task_set)
```

- 输入行数图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "input" < 1000 THEN concat( cast( "input" AS VARCHAR ), '行' ) WHEN "input" < 1000 * 1000 THEN concat( cast( round( "input"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN "input" < 10000000000.0 THEN concat( cast( round( "input"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "input"/ 1000.0 < 10000000000.0 THEN concat( cast( round( "input"/ 1000.0 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "input"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("input") as "input")
```

- 输出行数图表所关联的查询分析语句如下所示

```
* | SELECT CASE WHEN "output" < 1000 THEN concat( cast( "output" AS VARCHAR ), '行' ) WHEN "output" < 1000 * 1000 THEN concat( cast( round( "output"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN "output" < 10000000000 THEN concat( cast( round( "output"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "\"output\""/ 1000.0 < 10000000000 THEN concat( cast( round( "output"/ 1000.0 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "output"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("output") as "output")
```

- 满足过滤条件行数图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "filters" < 1000 THEN concat( cast( "filters" AS VARCHAR ), '行' ) WHEN "filters" < 1000 * 1000 THEN concat( cast( round( "filters"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN "filters" < 10000000000 THEN concat( cast( round( "filters"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "filters"/ 1000.0 < 10000000000 THEN concat( cast( round( "filters"/ 1000.0 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "filters"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("filters") as "filters")
```

- 不满足过滤条件行数图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "filter_drops" < 1000 THEN concat( cast( "filter_drops" AS VARCHAR ), '行' ) WHEN "filter_drops" < 1000 * 1000 THEN concat( cast( round( "filter_drops"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN "filter_drops" < 10000000000 THEN concat( cast( round( "filter_drops"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "filter_drops"/ 1000.0 < 10000000000 THEN concat( cast( round( "filter_drops"/ 1000.0 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "filter_drops"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("filter_drops") as "filter_drops")
```

- 采样行数图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "samples" < 1000 THEN concat( cast( "samples" AS VARCHAR ), '行' ) WHEN "samples" < 1000 * 1000 THEN concat( cast( round( "samples"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN "samples" < 10000000000 THEN concat( cast( round( "samples"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "samples"/ 1000.0 < 10000000000 THEN concat( cast( round( "samples"/ 1000.0 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "samples"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("samples") as "samples")
```

- 采样丢弃行数图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "sample_drops" < 1000 THEN concat( cast( "sample_drops" AS VARCHAR ), '行' ) WHEN "sample_drops" < 1000 * 1000 THEN concat( cast( round( "sample_drops"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN "sample_drops" < 10000000000 THEN
```

```
concat( cast( round( "sample_drops"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN  
"sample_drops"/ 1000.0 < 1000000000 THEN concat( cast( round( "sample_drops"/ 1000.0 /  
1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "sample_drops"/ 1000.0 / 1000 /  
1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum( "sample_drops" ) as  
"sample_drops")
```

- **超过日志时间范围行数**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "out_of_bound" < 1000 THEN concat( "out_of_bounds" AS  
VARCHAR ), '行' ) WHEN "out_of_bounds" < 1000 * 1000 THEN  
concat( cast( round( "out_of_bounds"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN "out_of_bounds" <  
1000000000 THEN concat( cast( round( "out_of_bounds"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )  
WHEN "out_of_bounds"/ 1000.0 < 1000000000 THEN concat( cast( round( "out_of_bounds"/ 1000.0 /  
1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "out_of_bounds"/ 1000.0 / 1000 /  
1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("out_of_bounds") as  
"out_of_bounds")
```

- **执行记录**图表所关联的查询分析语句如下所示：

```
* | select from_unixtime( " __time" ) as "统计时间", sum("input") as "输入行数", sum("output") as "输出  
行数", sum("filters") as "满足过滤条件行数", sum("filter_drops") as "不满足过滤条件行数  
", sum("samples") as "采样行数", sum("sample_drops") as "采样丢弃行数", sum("out_of_bounds") as "超  
过日志时间范围行数" group by "统计时间" order by "统计时间" desc limit 1000
```

----结束

OBS 导入任务监控仪表盘

步骤1 在仪表盘模板下方，选择“LTS仪表盘模板”。

步骤2 在仪表盘列表页面，单击“OBS导入任务监控仪表盘”仪表盘名称，查看图表详情。

- **过滤任务ID**，所关联的查询分析语句如下所示：

```
* | select task_set_id group by task_set_id
```

- **过滤任务名称**，所关联的查询分析语句如下所示：

```
* | select task_set_name group by task_set_name
```

- **成功条数**图表所关联的查询分析语句如下所示：

```
* | select sum(max_read_line) as "成功条数" from (select max_by(read_lines, __time) as max_read_line  
group by task_set_id, period_times limit 10000)
```

- **失败条数**图表所关联的查询分析语句如下所示：

```
* | select sum(max_failure_lines) as "失败条数" from (select max_by(failure_lines, __time) as  
max_failure_lines group by task_set_id, period_times limit 10000)
```

- **丢弃条数**图表所关联的查询分析语句如下所示：

```
* | select sum(max_ignore_lines) as "丢弃条数" from (select max_by(ignore_lines, __time) as  
max_ignore_lines group by task_set_id, period_times limit 10000)
```

- **读流量**图表所关联的查询分析语句如下所示：

```
* | select sum(max_in_flow)/1024.0/1024 as "读流量" from (select max_by(in_flow, __time) as  
max_in_flow group by task_set_id, period_times limit 10000)
```

- **写流量**图表所关联的查询分析语句如下所示：

```
* | select sum(max_out_flow)/1024.0/1024 as "写流量" from (select max_by(out_flow, __time) as  
max_out_flow group by task_set_id, period_times limit 10000)
```

- **导入进度**图表所关联的查询分析语句如下所示：

```
* | select max_by(case when total_files = 0 or total_files is null then 'NA' else cast(complete_files*1.0/  
total_files*100 as VARCHAR) end, __time ) as progress
```

- **处理速率**图表所关联的查询分析语句如下所示：

```
* | SELECT t AS "统计时间", ROUND(( CASE WHEN diff [ 1 ] IS NOT NULL AND diff [ 2 ] IS NOT  
NULL AND diff [ 1 ] - diff [ 2 ] > 0 THEN diff [ 1 ] - diff [ 2 ] WHEN diff [ 1 ] IS NOT NULL  
AND diff [ 2 ] IS NULL THEN diff [ 1 ] ELSE 0 END ) / 300, 0) AS "成功条数/s" FROM  
( SELECT t, ts_compare ( max_read_lines, 300 ) AS diff FROM ( SELECT t, sum( max_read_lines )  
AS max_read_lines FROM ( SELECT from_unixtime( __time - __time % 300000 ) AS t, max_by  
( read_lines, __time ) AS max_read_lines, task_set_id, period_times GROUP BY t, task_set_id,  
period_times LIMIT 10000 ) GROUP BY t ORDER BY t ) GROUP BY t ORDER BY t LIMIT  
10000 ) LIMIT 10000
```

- 运行异常图表所关联的查询分析语句如下所示：

```
* | select task_set_name as "任务名称", task_set_id as "任务ID", max_by(task_set_state, __time) as "任务状态", from_unixtime(max(__time)) as "采集时间", max_by(failure_lines, __time) as "失败条数" group by task_set_id, task_set_name HAVING "任务状态" = 'UNHEALTHY' OR "任务状态" = 'FAILURE' limit 10000
```
- 运行状态图表所关联的查询分析语句如下所示：

```
* | select task_set_name as "任务名称", task_set_id as "任务ID", max_by(task_set_state, __time) as "任务状态", max_by(read_lines, __time) as "成功条数", max_by(failure_lines, __time) as "失败条数", max_by(in_flow, __time)/1024/1024 as "读流量 MB", max_by(out_flow, __time)/1024/1024 as "写流量 MB", period_times as "周期" group by task_set_id, task_set_name, period_times order by max(__time), task_set_id, period_times limit 10000
```

----结束

资源用量明细仪表盘

步骤1 在仪表盘模板下方，选择“LTS仪表盘模板”。

步骤2 在仪表盘列表页面，单击“资源用量明细”仪表盘名称，查看图表详情。

若仪表盘中的部分图表展示异常，您可在索引配置页面单击“自动配置”更新索引字段，详细操作请参考[创建LTS日志索引](#)。索引更新完成后，返回仪表盘页面，查看图表显示正常。

- 过滤日志组ID，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" | select log_group_id from log_group by log_group_id limit 100000
```
- 过滤日志组名称，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" | select log_group_name_alias from log_group by log_group_name_alias limit 100000
```
- 过滤日志流ID，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" | select log_stream_id from log_group by log_stream_id limit 100000
```
- 过滤日志流名称，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" | select log_stream_name_alias from log_group by log_stream_name_alias limit 100000
```
- 原始日志流量/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" and metric_name:"storage_traffic" | select sum(counter)/1024/1024 as "原始日志流量/MB"
```
- 读流量(压缩后)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" and metric_name:"read" | select sum(counter)/1024/1024/5 as "读流量/MB"
```
- 写流量(压缩后)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" and metric_name:"write_traffic" | select sum(counter)/1024/1024 as "写流量/MB"
```
- 索引流量(标准型)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" and metric_name:"index_traffic" | select sum(counter)/1024/1024 as "索引流量/MB"
```
- 索引流量(搜索型)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" and metric_name:"search_index_traffic" | select sum(counter)/1024/1024 as "索引流量/MB"
```
- 日志DSL加工流量/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" and metric_name:"dsl_traffic" && counter:* | select sum(counter)/1024/1024 as "日志DSL加工流量/MB"
```
- 日志转指标流量/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__*" and not log_group_name_alias:"__default__*" and metric_name:"log2Metric" | select sum(counter)/1024/1024 as "日志转指标流量/MB"
```

- 标准存储量(最新值)/MB图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"storage\" && counter:* | select sum(max_storage) as \"标准存储量 (最新值) /MB
\" from (select max_by(counter,__time)/1024/1024 as max_storage group by log_stream_id limit
10000)`
- 冷存储量(最新值)/MB图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"cold_storage\" && counter:* | select sum(max_cold_storage) as \"冷存储量 (最新
值) /MB\" from (select max_by(counter,__time)/1024/1024 as max_cold_storage group by
log_stream_id limit 10000)`
- 基础转储流量/MB图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
(metric_name:\"obs_transfer\" or metric_name:\"dms_transfer\" or metric_name:\"dis_transfer\") |
select sum(counter)/1024.0/1024 as \"基础转储流量/MB\"`
- 高级转储流量/MB图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
(metric_name:\"dws_transfer\" or metric_name:\"dli_transfer\" or metric_name:\"obs_orc_transfer\") |
select sum(counter)/1024/1024 as \"高级转储流量/MB\"`
- 原始日志流量图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"storage_traffic\" | select from_unixtime(__time - __time%60000) as \"时间
\",sum(counter/1024/1024) as \"原始日志流量/MB\" group by \"时间\" order by \"时间\" limit 10000`
- 读流量(压缩后)图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"read\" | select from_unixtime(__time - __time%60000) as \"时间\",sum(counter)/
1024/1024 as \"读流量/MB\" group by \"时间\" order by \"时间\" limit 10000`
- 写流量(压缩后)图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"write_traffic\" | select from_unixtime(__time - __time%60000) as \"时间
\",sum(counter)/1024/1024 as \"写流量/MB\" group by \"时间\" order by \"时间\" limit 10000`
- 索引流量(标准型)图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"index_traffic\" | select from_unixtime(__time - __time%60000) as \"时间
\",sum(counter)/1024/1024 as \"索引流量/MB\" group by \"时间\" order by \"时间\" limit 10000`
- 索引流量(搜索型)图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"search_index_traffic\" | select from_unixtime(__time - __time%60000) as \"时间
\",sum(counter)/1024/1024 as \"索引流量/MB\" group by \"时间\" order by \"时间\" limit 10000`
- 日志DSL加工流量图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"dsl_traffic\" | select from_unixtime(__time - __time%60000) as \"时间\",sum(counter)/
1024/1024 as \"日志DSL加工流量/MB\" group by \"时间\" order by \"时间\" limit 10000`
- 日志转指标流量图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"log2Metric\" | select from_unixtime(__time - __time%60000) as \"时间\",sum(counter)/
1024/1024 as \"日志转指标流量/MB\" group by \"时间\" order by \"时间\" limit 10000`
- 标准存储量(最新值)图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"storage\" | select from_unixtime(__time - __time%60000) as \"时间
\",max_by(counter,__time)/1024/1024 as \"标准存储量 (最新值) /MB\" group by \"时间\" order by \"时
间\" limit 10000`
- 冷存储量(最新值)图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
metric_name:\"cold_storage\" | select from_unixtime(__time - __time%60000) as \"时间
\",max_by(counter,__time)/1024/1024 as \"冷存储量 (最新值) /MB\" group by \"时间\" order by \"时
间\" limit 10000`
- 基础转储流量图表所关联的查询分析语句如下所示:
`not log_group_id:\"__default_\" and not log_group_name_alias:\"__default_\" and
(metric_name:\"obs_transfer\" or metric_name:\"dms_transfer\" or metric_name:\"dis_transfer\") |`

```
select from_unixtime(__time - __time%60000) as \"时间\",sum(counter)/1024/1024 as \"基础转储流量/MB\" group by \"时间\" order by \"时间\" limit 10000
```

- 高级转储流量图表所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" and  
(metric_name:\"dws_transfer\" or metric_name:\"dli_transfer\" or metric_name:\"obs_orc_transfer\") |  
select from_unixtime(__time - __time%60000) as \"时间\",sum(counter)/1024/1024 as \"高级转储流量/MB\" group by \"时间\" order by \"时间\" limit 10000
```

----结束

用量 Top 排序仪表盘

步骤1 在仪表盘模板下方，选择“LTS仪表盘模板”。

步骤2 在仪表盘列表页面，单击“用量Top排序”仪表盘名称，查看图表详情。

若仪表盘中的部分图表展示异常，您可在索引配置页面单击“自动配置”更新索引字段，详细操作请参考[创建LTS日志索引](#)。索引更新完成后，返回仪表盘页面，查看图表显示正常。

- 过滤日志组ID，所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" | select log_group_id  
from log_group by log_group_id limit 100000
```

- 过滤日志组名称，所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" | select  
log_group_name_alias from log_group by log_group_name_alias limit 100000
```

- 过滤日志流ID，所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" | select log_stream_id  
from log_group by log_stream_id limit 100000
```

- 过滤日志流名称，所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" | select  
log_stream_name_alias from log_group by log_stream_name_alias limit 100000
```

- 原始日志流量/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" and  
metric_name:\"storage_traffic\" | select sum(counter)/1024/1024 as \"原始日志流量/MB\"
```

- 读流量(压缩后)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" and metric_name:\"read\"  
| select sum(counter)/1024/1024 as \"读流量/MB\"
```

- 写流量(压缩后)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" and  
metric_name:\"write_traffic\" | select sum(counter)/1024/1024 as \"写流量/MB\"
```

- 索引流量(标准型)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" and  
metric_name:\"index_traffic\" | select sum(counter)/1024/1024 as \"索引流量/MB\"
```

- 索引流量(搜索型)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" and  
metric_name:\"search_index_traffic\" | select sum(counter)/1024/1024 as \"索引流量/MB\"
```

- 日志DSL加工流量/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" and  
metric_name:\"dsl_traffic\" && counter:* | select sum(counter)/1024/1024 as \"日志DSL加工流量/MB\"
```

- 日志转指标流量/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" and  
metric_name:\"log2Metric\" | select sum(counter)/1024/1024 as \"日志转指标流量/MB\"
```

- 标准存储量(最新值)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:\"__default__\" and not log_group_name_alias:\"__default__\" and  
metric_name:\"storage\" && counter:* | select sum(max_storage) as \"标准存储量(最新值)/MB\" from  
(select max_by(counter,__time)/1024/1024 as max_storage group by log_stream_id limit 10000 )
```


- 冷存储量(最新值)/MB图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" and not log_group_name_alias:"__default__" and metric_name:"cold_storage" && counter:* | select sum(max_cold_storage) as "冷存储量 (最新值) /MB" from (select max_by(counter,_time)/1024/1024 as max_cold_storage group by log_stream_id limit 10000 )`
- 基础转储流量/MB图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" and not log_group_name_alias:"__default__" and (metric_name:"obs_transfer" or metric_name:"dms_transfer" or metric_name:"dis_transfer") | select sum(counter)/1024/1024 as "基础转储流量/MB"`
- 高级转储流量/MB图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" and not log_group_name_alias:"__default__" and (metric_name:"dws_transfer" or metric_name:"dli_transfer" or metric_name:"obs_orc_transfer") | select sum(counter)/1024/1024 as "高级转储流量/MB"`
- TOP日志组用量图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" | select log_group_id as "日志组ID", sum(case when metric_name = 'storage_traffic' then counter else 0 end)*1.0/1024/1024 as "原始日志流量/MB", sum(case when metric_name = 'index_traffic' then counter else 0 end)*1.0/1024/1024 as "索引流量 (标准型) /MB", sum(case when metric_name = 'search_index_traffic' then counter else 0 end)*1.0/1024/1024 as "索引流量 (搜索型) /MB", sum(case when metric_name = 'write_traffic' then counter else 0 end)*1.0/1024/1024 as "写流量 (压缩后) /MB", sum(case when metric_name = 'storage' then counter else 0 end)*1.0/1024/1024 as "标准存储量 (最新值) /MB", sum(case when metric_name = 'cold_storage' then counter else 0 end)*1.0/1024/1024 as "冷存储量 (最新值) /MB" group by log_group_id order by "原始日志流量/MB" desc`
- TOP日志流用量图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" | select log_stream_id as "日志流ID", sum(case when metric_name = 'storage_traffic' then counter else 0 end)*1.0/1024/1024 as "原始日志流量/MB", sum(case when metric_name = 'index_traffic' then counter else 0 end)*1.0/1024/1024 as "索引流量 (标准型) /MB", sum(case when metric_name = 'search_index_traffic' then counter else 0 end)*1.0/1024/1024 as "索引流量 (搜索型) /MB", sum(case when metric_name = 'write_traffic' then counter else 0 end)*1.0/1024/1024 as "写流量 (压缩后) /MB",sum(case when metric_name = 'storage' then counter else 0 end)*1.0/1024/1024 as "标准存储量 (最新值) /MB",sum(case when metric_name = 'cold_storage' then counter else 0 end)*1.0/1024/1024 as "冷存储量 (最新值) /MB" group by log_stream_id order by "原始日志流量/MB" desc`

----结束

DSL 加工任务监控中心

步骤1 在仪表盘模板下方，选择“DSL仪表盘模板 > DSL加工任务监控中心”，查看图表详情。

- 过滤加工任务ID，所关联的查询分析语句如下所示:
`* | select distinct(task_id)`
- 过滤加工任务名称，所关联的查询分析语句如下所示:
`* | select distinct(task_name)`
- 输入行数图表所关联的查询分析语句如下所示:
`* | SELECT CASE WHEN "input" < 1000 THEN concat(cast("input" AS VARCHAR), '行') WHEN "input" < 1000 * 1000 THEN concat(cast(round("input"/ 1000, 1) AS VARCHAR), '千行') WHEN "input" < 1000000000 THEN concat(cast(round("input"/ 1000000.0, 1) AS VARCHAR), '百万行') WHEN "input"/ 1000.0 < 1000000000 THEN concat(cast(round("input"/ 1000 / 1000000.0, 1) AS VARCHAR), '十亿行') ELSE concat(cast(round("input"/ 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR), '万亿行') END AS "total" from (select sum("process.accept") as "input")`
- 输出行数图表所关联的查询分析语句如下所示:
`* | SELECT CASE WHEN "delivered" < 1000 THEN concat(cast("delivered" AS VARCHAR), '行') WHEN "delivered" < 1000 * 1000 THEN concat(cast(round("delivered"/ 1000, 1) AS VARCHAR), '千行') WHEN "delivered" < 1000000000 THEN concat(cast(round("delivered"/ 1000000.0, 1) AS VARCHAR), '百万行') WHEN "delivered"/ 1000.0 < 1000000000 THEN concat(cast(round("delivered"/ 1000 / 1000000.0, 1) AS VARCHAR), '十亿行') ELSE concat(cast(round("delivered"/ 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR), '万亿行') END AS "total" from (select sum("process.delivered") as "delivered")`
- 过滤行数图表所关联的查询分析语句如下所示:

```
* | SELECT CASE WHEN "drop" < 1000 THEN concat( cast( "drop" AS VARCHAR ), '行' ) WHEN
"drop" < 1000 * 1000 THEN concat( cast( round( "drop"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN
"drop" < 1000000000 THEN concat( cast( round( "drop"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )
WHEN "drop"/ 1000.0 < 1000000000 THEN concat( cast( round( "drop"/ 1000 / 1000 / 1000, 1 ) AS
VARCHAR ), '十亿行' ) ELSE concat( cast( round( "drop"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS
VARCHAR ), '万亿行' ) END AS "total" from (select sum("process.drop") as "drop")
```

- **失败行数图表所关联的查询分析语句如下所示：**

```
* | SELECT CASE WHEN "failed" < 1000 THEN concat( cast( "failed" AS VARCHAR ), '行' ) WHEN
"failed" < 1000 * 1000 THEN concat( cast( round( "failed"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN
"failed" < 1000000000 THEN concat( cast( round( "failed"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )
WHEN "failed"/ 1000.0 < 1000000000 THEN concat( cast( round( "failed"/ 1000 / 1000000.0, 1 ) AS
VARCHAR ), '十亿行' ) ELSE concat( cast( round( "failed"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS
VARCHAR ), '万亿行' ) END AS "total" from (select sum("process.failed") as "failed")
```

- **执行记录图表所关联的查询分析语句如下所示：**

```
* | select MILLIS_TO_TIMESTAMP("start") as "统计开始时间", MILLIS_TO_TIMESTAMP("end") as "统计
结束时间", "process.accept" as "输入行数", "process.delivered" as "输出行数", "process.drop" as "过滤行
数", "process.failed" as "失败行数" limit 1000
```

----结束

7.3.3.15 METRIC 仪表盘模板

用户在LTS页面只需按照业务需要创建指标规则即可生成自己的统计报表，设置单个日志过滤条件或通过添加关联关系和添加组设置多个日志过滤条件，保留符合条件的日志，对用户特定时间范围内已结构化的日志进行动态统计，并将统计结果动态呈现到aom的Prometheus实例，操作简单且功能强大。

前提条件

- 已创建日志生成指标，详情请参见[日志生成指标（邀测）](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

日志生成指标任务监控中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“METRIC仪表盘模板 > 日志生成指标任务监控中心”，查看图表详情。

- **过滤规则ID，所关联的查询分析语句如下所示：**

```
* | select distinct(task_set)
```

- **输入行数图表所关联的查询分析语句如下所示：**

```
* | SELECT CASE WHEN "input" < 1000 THEN concat( cast( "input" AS VARCHAR ), '行' ) WHEN
"input" < 1000 * 1000 THEN concat( cast( round( "input"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN
"input" < 1000000000.0 THEN concat( cast( round( "input"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )
WHEN "input"/ 1000.0 < 1000000000.0 THEN concat( cast( round( "input"/ 1000.0 / 1000000.0, 1 )
AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "input"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS
VARCHAR ), '万亿行' ) END AS "total" from (select sum("input") as "input")
```

- **输出行数图表所关联的查询分析语句如下所示：**

```
* | SELECT CASE WHEN "output" < 1000 THEN concat( cast( "output" AS VARCHAR ), '行' ) WHEN
"output" < 1000 * 1000 THEN concat( cast( round( "output"/ 1000.0, 1 ) AS VARCHAR ), '千行' )
WHEN "output" < 1000000000 THEN concat( cast( round( "output"/ 1000000.0, 1 ) AS VARCHAR ), '百
万行' ) WHEN "output"/ 1000.0 < 1000000000 THEN concat( cast( round( "output"/ 1000.0 /
1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "output"/ 1000.0 / 1000 / 1000 /
1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("output") as "output")
```

- **满足过滤条件行数图表所关联的查询分析语句如下所示：**

```
* | SELECT CASE WHEN "filters" < 1000 THEN concat( cast( "filters" AS VARCHAR ), '行' ) WHEN
"filters" < 1000 * 1000 THEN concat( cast( round( "filters"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN
```



```
"filters" < 1000000000 THEN concat( cast( round( "filters"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )  
WHEN "filters"/ 1000.0 < 1000000000 THEN concat( cast( round( "filters"/ 1000.0 / 1000000.0, 1 ) AS  
VARCHAR ), '十亿行' ) ELSE concat( cast( round( "filters"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS  
VARCHAR ), '万亿行' ) END AS "total" from (select sum("filters") as "filters")
```

- **不满足过滤条件行数**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "filter_drops" < 1000 THEN concat( cast( "filter_drops" AS VARCHAR ), '行' )  
WHEN "filter_drops" < 1000 * 1000 THEN concat( cast( round( "filter_drops"/ 1000.0, 1 ) AS  
VARCHAR ), '千行' ) WHEN "filter_drops" < 1000000000 THEN concat( cast( round( "filter_drops"/  
1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "filter_drops"/ 1000.0 < 1000000000 THEN  
concat( cast( round( "filter_drops"/ 1000.0 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE  
concat( cast( round( "filter_drops"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END  
AS "total" from (select sum("filter_drops") as "filter_drops")
```

- **采样行数**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "samples" < 1000 THEN concat( cast( "samples" AS VARCHAR ), '行' )  
WHEN "samples" < 1000 * 1000 THEN concat( cast( round( "samples"/ 1000.0, 1 ) AS VARCHAR ), '千  
行' ) WHEN "samples" < 1000000000 THEN concat( cast( round( "samples"/ 1000000.0, 1 ) AS  
VARCHAR ), '百万行' ) WHEN "samples"/ 1000.0 < 1000000000 THEN  
concat( cast( round( "samples"/ 1000.0 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE  
concat( cast( round( "samples"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS  
"total" from (select sum("samples") as "samples")
```

- **采样丢弃行数**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "sample_drops" < 1000 THEN concat( cast( "sample_drops" AS VARCHAR ),  
'行' ) WHEN "sample_drops" < 1000 * 1000 THEN concat( cast( round( "sample_drops"/ 1000.0, 1 )  
AS VARCHAR ), '千行' ) WHEN "sample_drops" < 1000000000 THEN  
concat( cast( round( "sample_drops"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN  
"sample_drops"/ 1000.0 < 1000000000 THEN concat( cast( round( "sample_drops"/ 1000.0 /  
1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "sample_drops"/ 1000.0 / 1000 /  
1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum( "sample_drops" ) as  
"sample_drops")
```

- **超过日志时间范围行数**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN "out_of_bounds" < 1000 THEN concat( cast( "out_of_bounds" AS  
VARCHAR ), '行' ) WHEN "out_of_bounds" < 1000 * 1000 THEN  
concat( cast( round( "out_of_bounds"/ 1000.0, 1 ) AS VARCHAR ), '千行' ) WHEN "out_of_bounds" <  
1000000000 THEN concat( cast( round( "out_of_bounds"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )  
WHEN "out_of_bounds"/ 1000.0 < 1000000000 THEN concat( cast( round( "out_of_bounds"/ 1000.0 /  
1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "out_of_bounds"/ 1000.0 / 1000 /  
1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("out_of_bounds") as  
"out_of_bounds")
```

- **执行记录**图表所关联的查询分析语句如下所示：

```
* | select from_unixtime( "__time" ) as "统计时间", sum("input") as "输入行数", sum("output") as "输出  
行数", sum("filters") as "满足过滤条件行数", sum("filter_drops") as "不满足过滤条件行数"  
", sum("samples") as "采样行数", sum("sample_drops") as "采样丢弃行数", sum("out_of_bounds") as "超  
过日志时间范围行数" group by "统计时间" order by "统计时间" desc limit 1000
```

----结束

7.3.3.16 NGINX-INGRESS 仪表盘模板

日志服务支持采集NGINX-INGRESS日志，并进行多维度分析。云日志服务支持日志采集向导一站式采集NGINX-INGRESS日志，并为NGINX-INGRESS日志配置结构化和仪表盘。Nginx Ingress是专为Kubernetes集群设计的Ingress控制器，用于管理外部访问到集群内服务的路由规则。

NGINX-INGRESS仪表盘模板支持[查看NGINX-INGRESS秒级监控](#)、[查看NGINX-INGRESS访问中心](#)和[查看NGINX-INGRESS监控中心](#)。

前提条件

- 已采集NGINX-INGRESS日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 NGINX-INGRESS 秒级监控

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX秒级监控”，查看图表详情。

- 过滤appName，所关联的查询分析语句如下所示：

```
* | select appName group by appName
```

- QPS图表所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(TIME_CEIL(__time,'PT1S'),'yyyy-MM-dd HH:mm:ss','+08:00') AS __time_,  
COUNT(*) as QPS from log where req_id is not null group by __time_
```

- 成功率图表所关联的查询分析语句如下所示：

```
* | select __time,round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select  
TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, sum(case when  
status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log where req_id is not null group  
by __time)
```

- 延迟图表所关联的查询分析语句如下所示：

```
* | select __time,round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟",  
round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟"  
from (select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time,  
avg(request_time)* 1000 as '访问延迟',avg(upstream_response_time)* 1000 as 'Upstream延迟' from  
log where req_id is not null group by __time)
```

- 流量图表所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time ,  
sum("request_length") as "请求流量", sum("body_bytes_sent") as "返回body流量" where req_id is not  
null group by __time
```

- 状态码图表所关联的查询分析语句如下所示：

```
* | select t.t as "time",  
CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",  
CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",  
CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",  
CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",  
CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他"  
from (select TIME_CEIL(__time,'PT5S') as t from log where req_id is not null group by t order by t  
asc ) t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from  
log WHERE "status" >= 200 and "status" < 300 and req_id is not null group by t order by t asc ) a on  
t.t =a.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from  
log WHERE "status" >= 300 and "status" < 400 and req_id is not null group by t order by t asc ) b on  
t.t =b.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from  
log WHERE "status" >= 400 and "status" < 500 and req_id is not null group by t order by t asc ) c on  
t.t =c.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from  
log WHERE "status" >= 500 and "status" < 600 and req_id is not null group by t order by t asc ) d on  
t.t =d.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from  
log WHERE "status" < 200 or "status" >= 600 and req_id is not null group by t order by t asc ) e on t.t  
=e.t
```

- 后端响应码图表所关联的查询分析语句如下所示：

```
* | select t.t as "time",  
CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",  
CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",  
CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",  
CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",  
CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他"  
from (  
select TIME_CEIL(__time,'PT5S') as t from log where req_id is not null group by t order by t asc  
) t  
left join(  
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE  
"upstream_status" >= 200 and "upstream_status" < 300 and req_id is not null group by t order by t  
asc) a  
on t.t = a.t  
left join (  
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE
```

```
"upstream_status" >= 300 and "upstream_status" < 400 and req_id is not null group by t order by t asc) b
    on t.t = b.t
    left join (
        select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE
"upstream_status" >= 400 and "upstream_status" < 500 and req_id is not null group by t order by t asc) c
    on t.t = c.t
    left join (
        select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE
"upstream_status" >= 500 and "upstream_status" < 600 and req_id is not null group by t order by t asc) d
    on t.t = d.t
    left join (
        select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE
"upstream_status" < 200 or "upstream_status" >= 600 and req_id is not null group by t order by t asc) e
    on t.t = e.t
```

----结束

查看 NGINX-INGRESS 访问中心

步骤1 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX访问中心”，查看图表详情。

- 过滤appName，所关联的查询分析语句如下所示：

```
* | select appName group by appName
```

- PV对比昨日图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select
compare( "pv" , 86400) as diff from (select count(1) as "pv" from log where req_id is not null))
```

- PV对比上周图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select
compare( "pv" , 604800) as diff from (select count(1) as "pv" from log where req_id is not null))
```

- UV对比昨日图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select
compare( "uv" , 86400) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from
log where req_id is not null))
```

- UV对比上周图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select
compare( "uv" , 604800) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from
log where req_id is not null))
```

- 访问量PV分布(中国)图表所关联的查询分析语句如下所示：

```
* | select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY
(remote_addr) = '中国' and req_id is not null group by province HAVING province not in ('','保留地址',*)
```

- 访问量PV分布(世界)图表所关联的查询分析语句如下所示：

```
* | SELECT ip_to_country(remote_addr) as country,COUNT(1) as PV where req_id is not null GROUP BY
country HAVING country not in ('','保留地址',*)
```

- 访问量UV分布(中国)图表所关联的查询分析语句如下所示：

```
* | select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as UV
where IP_TO_COUNTRY (remote_addr) = '中国' and req_id is not null group by province HAVING
province not in ('','保留地址',*)
```

- 访问量UV分布(世界)图表所关联的查询分析语句如下所示：

```
* | select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv
where req_id is not null group by country HAVING country not in ('','保留地址',*)
```

- 平均时延分布（中国）图表所关联的查询分析语句如下所示：

```
* | SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS
"平均延迟(ms)"FROM (SELECT ip_to_province(remote_addr) as province,avg(request_time) * 1000 AS
"平均延迟(ms)" WHERE IP_TO_COUNTRY (remote_addr) = '中国' and req_id is not null GROUP BY
province HAVING province not in ('','保留地址',*))
```

- **平均时延分布（世界）**图表所关联的查询分析语句如下所示：

```
* | SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)" FROM (SELECT ip_to_country(remote_addr) as country,avg(request_time) * 1000 AS "平均延迟(ms)" where req_id is not null GROUP BY country HAVING country not in ('','保留地址',''))
```

- **今日PV/UV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( __time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(__time_,PT600S) AS _time_, count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE __time_ <= CURRENT_TIMESTAMP and __time_ >= DATE_TRUNC( 'DAY', (CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR and req_id is not null group by _time_ order by _time_) WHERE _time_ <= CURRENT_TIMESTAMP order by _time_ LIMIT 100000 OFFSET 1
```

- **7日PV/UV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( __time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(__time_,PT600S) AS _time_, count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE __time_ <= CURRENT_TIMESTAMP and __time_ >= DATE_TRUNC( 'DAY', (CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR - INTERVAL '7' DAY and req_id is not null group by _time_ order by _time_) WHERE _time_ <= CURRENT_TIMESTAMP order by _time_ LIMIT 100000 OFFSET 1
```

- **区域访问TOP10(省份)**图表所关联的查询分析语句如下所示：

```
* | select ip_to_province(remote_addr) as "province", count(1) as "访问次数" where req_id is not null group by "province" HAVING "province" <> '-1' order by "访问次数" asc limit 10
```

- **区域访问TOP10(城市)**图表所关联的查询分析语句如下所示：

```
* | select ip_to_city(remote_addr) as "city", count(1) as "访问次数" where req_id is not null group by "city" HAVING "city" <> '-1' order by "访问次数" asc limit 10
```

- **Host访问TOP10**图表所关联的查询分析语句如下所示：

```
* | select remote_addr as "Host", count(1) as "PV" where req_id is not null group by "Host" order by "PV" asc limit 10
```

- **UserAgent访问TOP10**图表所关联的查询分析语句如下所示：

```
* | select http_user_agent as "UserAgent", count(1) as "PV" where req_id is not null group by "UserAgent" order by "PV" asc limit 10
```

- **设备占比(终端)**图表所关联的查询分析语句如下所示：

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type , count(1) as total where req_id is not null group by type
```

- **设备占比(系统)**图表所关联的查询分析语句如下所示：

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total where req_id is not null group by type HAVING type != 'other'
```

- **TOP URL**图表所关联的查询分析语句如下所示：

```
* | select request_uri , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV, round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" where req_id is not null group by request_uri ORDER by PV desc
```

- **TOP 访问IP**图表所关联的查询分析语句如下所示：

```
* | select remote_addr as "来源IP",ip_to_country(remote_addr) as "国家",ip_to_province(remote_addr) as "省份",ip_to_city(remote_addr) as "城市",ip_to_provider(remote_addr) as "运营商",count(1) as "PV",http_user_agent as "UserAgent采样",request_uri as "URL采样" where req_id is not null group by remote_addr,http_user_agent,request_uri ORDER by "PV" desc
```

----结束

查看 NGINX-INGRESS 监控中心

步骤1 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX监控中心”，查看图表详情。

- **过滤appName**，所关联的查询分析语句如下所示：

```
* | select appName group by appName
```

- **访问量PV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( __time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV FROM ( SELECT TIME_CEIL ( __time_, 'PT300S' ) AS _time_, count( 1 ) AS PV FROM log where req_id is not null GROUP BY _time_ ) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **请求成功率**图表所关联的查询分析语句如下所示：
* | select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt where req_id is not null
- **平均延迟**图表所关联的查询分析语句如下所示：
* | select round(avg(request_time) * 1000, 3) as cnt where req_id is not null
- **4XX请求数**图表所关联的查询分析语句如下所示：
* | SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500 and req_id is not null
- **404请求数**图表所关联的查询分析语句如下所示：
* | SELECT COUNT(1) as cnt WHERE "status" = 404 and req_id is not null
- **429请求数**图表所关联的查询分析语句如下所示：
* | SELECT COUNT(1) as cnt WHERE "status" = 429 and req_id is not null
- **504请求数**图表所关联的查询分析语句如下所示：
* | SELECT COUNT(1) as cnt WHERE "status" = 504 and req_id is not null
- **5XX请求数**图表所关联的查询分析语句如下所示：
* | SELECT TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time_, cnt FROM (SELECT TIME_CEIL(_time_, 'PT300S') AS _time_, count(1) AS cnt FROM log where "status" >= 500 and req_id is not null GROUP BY _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- **状态码分布**图表所关联的查询分析语句如下所示：
* | SELECT status, COUNT(1) AS rm where req_id is not null GROUP BY status
- **访问量UV**图表所关联的查询分析语句如下所示：
* | SELECT TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time_, UV FROM (select TIME_CEIL(_time_, 'PT600S') AS _time_, APPROX_COUNT_DISTINCT(remote_addr) as UV from log where req_id is not null group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- **流量**图表所关联的查询分析语句如下所示：
* | select TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') AS _time_, round(CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2) AS "入流量", round(CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2) AS "出流量" FROM (SELECT TIME_CEIL(_time_, 'PT600S') AS _time_, sum(request_length) / 1024.0 AS "入流量", sum(body_bytes_sent) / 1024.0 AS "出流量" where req_id is not null group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- **访问失败率**图表所关联的查询分析语句如下所示：
* | SELECT TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time_, round(CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2) AS "失败率", round(CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2) AS "5XX比例" from (select TIME_CEIL(_time_, 'PT600S') AS _time_, sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率', sum(case when status >= 500 then 1 else 0 end) * 100.0 / COUNT(1) as '5XX比例' where req_id is not null group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- **延迟**图表所关联的查询分析语句如下所示：
* | select TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time_, round(CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2) AS "平均", round(CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2) AS "P50", round(CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2) AS "P90", round(CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2) AS "P99", round(CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2) AS "P9999" from (select TIME_CEIL(_time_, 'PT600S') as _time_, avg(request_time) * 1000 as "平均", APPROX_QUANTILE_DS("request_time", 0.50) * 1000 as "P50", APPROX_QUANTILE_DS("request_time", 0.90) * 1000 as "P90", APPROX_QUANTILE_DS("request_time", 0.99) * 1000 as "P99", APPROX_QUANTILE_DS("request_time", 0.9999) * 1000 as "P9999" where req_id is not null group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- **Host请求TOP**图表所关联的查询分析语句如下所示：
* | SELECT "remote_addr", pv, uv, round(CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2) AS "访问成功率(%)", round(CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3) AS "平均延迟(ms)", round(CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3) AS "入流量(KB)", round(CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3) AS "出流量(KB)" FROM (SELECT "remote_addr", count(1) AS pv, APPROX_COUNT_DISTINCT (remote_addr) AS uv, sum(CASE WHEN "status" < 400 THEN 1 ELSE 0 END) * 100.0 / count(1) AS "访问成功率(%)", avg(request_time) * 1000 AS "平均延迟(ms)", sum(request_length) / 1024.0 AS "入流量(KB)", sum(body_bytes_sent) / 1024.0 AS "出流量(KB)" WHERE "remote_addr" != "" and req_id is not null GROUP BY "remote_addr") ORDER BY pv DESC

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "remote_addr", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "remote_addr", count( 1 ) AS pv, sum( CASE WHEN
"status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) *
1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != "" and
req_id is not null GROUP BY "remote_addr" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "remote_addr", pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0
END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "remote_addr", count( 1 ) AS pv, sum( CASE WHEN
"status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) *
1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != "" and
req_id is not null GROUP BY "remote_addr" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT request_uri, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END,
3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出
流量(KB)" FROM ( SELECT request_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr )
AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率
(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",
sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "remote_addr" != "" and req_id is not null
GROUP BY request_uri ) ORDER BY pv desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT request_uri, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" < 400
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延
迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != "" and
req_id is not null GROUP BY request_uri ) ORDER BY "平均延迟(ms)" desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0
END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status"
>= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS
"平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != "" and
req_id is not null GROUP BY request_uri ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,
2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,
3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS
"入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量
(KB)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT
( remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS
"访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS
"入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "remote_addr" != "" and
req_id is not null GROUP BY addr having length(upstream_addr) > 2 ) ORDER BY "pv" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )
AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99
延迟(ms)"
```

```
延迟(ms)" FROM (SELECT upstream_addr as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != " and req_id is not null GROUP BY addr having length(upstream_addr) > 2) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示:

```
* | SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != " and req_id is not null GROUP BY addr having length(upstream_addr) > 2) ORDER BY "访问失败率(%)" desc
```

----结束

7.3.3.17 NGINX 仪表盘模板

日志服务支持采集NGINX日志，并进行多维度分析。云日志服务支持日志采集向导一站式采集NGINX日志，并为NGINX日志配置结构化和仪表盘。Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。

NGINX仪表盘模板支持[查看NGINX秒级监控](#)、[查看NGINX访问中心](#)和[查看NGINX监控中心](#)。

前提条件

- 已采集NGINX日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 NGINX 秒级监控

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX秒级监控”，查看图表详情。

- **QPS**图表所关联的查询分析语句如下所示:

```
* | select TIME_FORMAT(TIME_CEIL(__time,'PT1S'),'yyyy-MM-dd HH:mm:ss','+08:00') AS __time_, COUNT(*) as QPS from log group by __time_
```

- **成功率**图表所关联的查询分析语句如下所示:

```
* | select __time,round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group by __time)
```

- **延迟**图表所关联的查询分析语句如下所示:

```
* | select __time,round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟",round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟" from (select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, avg(request_time)* 1000 as '访问延迟',avg(upstream_response_time)* 1000 as 'Upstream延迟' from log group by __time)
```

- **流量**图表所关联的查询分析语句如下所示:

```
* | select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, sum("request_length") as "请求流量", sum("body_bytes_sent") as "返回body流量" group by __time
```

- **状态码**图表所关联的查询分析语句如下所示:

```
* | select t.t as "time", CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",
```



```
CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",
CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",
CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",
CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他"
from (select TIME_CEIL(__time,'PT5S') as t from log group by t order by t asc ) t left join (select
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "status" >=
200 and "status" < 300 group by t order by t asc ) a on t.t =a.t left join (select
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "status" >=
300 and "status" < 400 group by t order by t asc) b on t.t =b.t left join (select
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "status" >=
400 and "status" < 500 group by t order by t asc) c on t.t =c.t left join (select
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "status" >=
500 and "status" < 600 group by t order by t asc) d on t.t =d.t left join (select
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "status" <
200 or "status" >= 600 group by t order by t asc) e on t.t =e.t
```

- 后端响应码图表所关联的查询分析语句如下所示：

```
* | select t.t as "time",
CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",
CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",
CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",
CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",
CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他"
from (
select TIME_CEIL(__time,'PT5S') as t from log group by t order by t asc
) t
left join(
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE
"upstream_status" >= 200 and "upstream_status" < 300 group by t order by t asc) a
on t.t = a.t
left join (
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE
"upstream_status" >= 300 and "upstream_status" < 400 group by t order by t asc) b
on t.t =b.t
left join (
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE
"upstream_status" >= 400 and "upstream_status" < 500 group by t order by t asc) c
on t.t =c.t
left join (
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE
"upstream_status" >= 500 and "upstream_status" < 600 group by t order by t asc) d
on t.t =d.t
left join (
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE
"upstream_status" < 200 or "upstream_status" >= 600 group by t order by t asc) e
on t.t =e.t
```

----结束

查看 NGINX 访问中心

步骤1 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX访问中心”，查看图表详情。

- **PV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select
compare( "pv" , 86400) as diff from (select count(1) as "pv" from log))
```

- **PV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select
compare( "pv" , 604800) as diff from (select count(1) as "pv" from log))
```

- **UV对比昨日**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select
compare( "uv" , 86400) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from
log))
```

- **UV对比上周**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select
compare( "uv" , 604800) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from
log))
```


- **访问量PV分布(中国)图表所关联的查询分析语句如下所示:**

```
* | select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','')
```
- **访问量PV分布(世界)图表所关联的查询分析语句如下所示:**

```
* | SELECT ip_to_country(remote_addr) as country,COUNT(1) as PV GROUP BY country HAVING country not in ('','保留地址','')
```
- **访问量UV分布(中国)图表所关联的查询分析语句如下所示:**

```
* | select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as UV where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','')
```
- **访问量UV分布(世界)图表所关联的查询分析语句如下所示:**

```
* | select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv group by country HAVING country not in ('','保留地址','')
```
- **平均时延分布(中国)图表所关联的查询分析语句如下所示:**

```
* | SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(remote_addr) as province,avg(request_time) * 1000 AS "平均延迟(ms)"WHERE IP_TO_COUNTRY (remote_addr) = '中国'GROUP BY province HAVING province not in ('','保留地址',''))
```
- **平均时延分布(世界)图表所关联的查询分析语句如下所示:**

```
* | SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(remote_addr) as country,avg(request_time) * 1000 AS "平均延迟(ms)" GROUP BY country HAVING country not in ('','保留地址',''))
```
- **今日PV/UV图表所关联的查询分析语句如下所示:**

```
* | SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(_time_,PT600S) AS _time_, count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE _time_ <= CURRENT_TIMESTAMP and _time_ >= DATE_TRUNC( 'DAY', (CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR group by _time_ order by _time_ ) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```
- **7日PV/UV图表所关联的查询分析语句如下所示:**

```
* | SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(_time_,PT600S) AS _time_, count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE _time_ <= CURRENT_TIMESTAMP and _time_ >= DATE_TRUNC( 'DAY', (CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR - INTERVAL '7' DAY group by _time_ order by _time_ ) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```
- **区域访问TOP10(省份)图表所关联的查询分析语句如下所示:**

```
* | select ip_to_province(remote_addr) as "province", count(1) as "访问次数" group by "province" HAVING "province" <> '-1' order by "访问次数" asc limit 10
```
- **区域访问TOP10(城市)图表所关联的查询分析语句如下所示:**

```
* | select ip_to_city(remote_addr) as "city", count(1) as "访问次数" group by "city" HAVING "city" <> '-1' order by "访问次数" asc limit 10
```
- **Host访问TOP10图表所关联的查询分析语句如下所示:**

```
* | select host as "Host", count(1) as "PV" group by "Host" order by "PV" asc limit 10
```
- **UserAgent访问TOP10图表所关联的查询分析语句如下所示:**

```
* | select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" asc limit 10
```
- **设备占比(终端)图表所关联的查询分析语句如下所示:**

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type , count(1) as total group by type
```
- **设备占比(系统)图表所关联的查询分析语句如下所示:**

```
* | select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total group by type HAVING type != 'other'
```
- **TOP URL图表所关联的查询分析语句如下所示:**

```
* | select request_uri , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV, round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group by request_uri ORDER by PV desc
```
- **TOP 访问IP图表所关联的查询分析语句如下所示:**

```
* | select remote_addr as "来源IP",ip_to_country(remote_addr) as "国家",ip_to_province(remote_addr) as "省份",ip_to_city(remote_addr) as "城市",ip_to_provider(remote_addr) as "运营商",count(1) as "PV",http_user_agent as "UserAgent采样",request_uri as "URL采样" group by remote_addr,http_user_agent,request_uri ORDER by "PV" desc
```

----结束

查看 NGINX 监控中心

步骤1 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX监控中心”，查看图表详情。

- **访问量PV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , PV FROM ( SELECT TIME_CEIL( _time_ , 'PT300S' ) AS _time_ , count( 1 ) AS PV FROM log GROUP BY _time_ ) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **请求成功率**图表所关联的查询分析语句如下所示：

```
* | select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```

- **平均延迟**图表所关联的查询分析语句如下所示：

```
* | select round(avg(request_time) * 1000, 3) as cnt
```

- **4XX请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
```

- **404请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 404
```

- **429请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 429
```

- **504请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT COUNT(1) as cnt WHERE "status" = 504
```

- **5XX请求数**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , cnt FROM ( SELECT TIME_CEIL( _time_ , 'PT300S' ) AS _time_ , count( 1 ) AS cnt FROM log where "status" >= 500 GROUP BY _time_ ) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **状态码分布**图表所关联的查询分析语句如下所示：

```
* | SELECT status, COUNT(1) AS rm GROUP BY status
```

- **访问量UV**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , UV FROM (select TIME_CEIL(_time_,'PT600S') AS _time_ , APPROX_COUNT_DISTINCT(remote_addr) as UV from log group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **流量**图表所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(_time_,'yyyy-MM-dd HH:mm:ss','+08:00') AS _time_ , round( CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量" , round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(_time_,'PT600S') AS _time_ , sum(request_length) / 1024.0 AS "入流量" , sum(bytes_sent) / 1024.0 AS "出流量" group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **访问失败率**图表所关联的查询分析语句如下所示：

```
* | SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , round( CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率" , round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(_time_,'PT600S') AS _time_ , sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率' , sum(case when status >= 500 THEN 1 ELSE 0 END)*100.0/COUNT(1) as '5XX比例' group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **延迟**图表所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , round( CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均" , round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 ) AS "P50" , round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90" , round( CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99" , round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(_time_,'PT600S') as _time_ , avg(request_time) * 1000 as "平均" , APPROX_QUANTILE_DS("request_time" , 0.50)*1000 as "P50" , APPROX_QUANTILE_DS("request_time" , 0.90)*1000 as "P90" , APPROX_QUANTILE_DS("request_time" , 0.99)*1000 as
```

```
'P99',APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as 'P9999' group by _time_) WHERE  
_time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **Host请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0  
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END,  
3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出  
流量(KB)" FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr ) AS  
uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",  
avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",  
sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY "host" ) ORDER BY pv  
DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,  
2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,  
3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 )  
AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS  
"P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1  
ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟  
(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT "host", pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 )  
AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE  
0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"host" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT request_uri, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0  
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END,  
3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出  
流量(KB)" FROM ( SELECT request_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr )  
AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率  
(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",  
sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY request_uri ) ORDER BY  
pv desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT request_uri, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0  
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0  
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END,  
3 ) AS "P99延迟(ms)" FROM ( SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" < 400  
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延  
迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
request_uri ) ORDER BY "平均延迟(ms)" desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0  
END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0  
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0  
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status"  
>= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS  
"平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
request_uri ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )  
AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )
```

```
AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM (SELECT upstream_addr as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400  
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延  
迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟  
(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != "  
GROUP BY addr having length(upstream_addr) > 2) ORDER BY "平均延迟(ms)" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,  
2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,  
3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS  
"入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量  
(KB)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS  
"访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS  
"入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " GROUP BY addr  
having length(upstream_addr) > 2) ORDER BY "pv" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示：

```
* | SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 )  
AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >=  
400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平  
均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " GROUP BY  
addr having length(upstream_addr) > 2)ORDER BY "访问失败率(%)" desc
```

----结束

7.3.3.18 WAF 仪表盘模板

Web应用防火墙（Web Application Firewall，WAF），通过对HTTP(S)请求进行检测，识别并阻断SQL注入、跨站脚本攻击、网页木马上传、命令/代码注入、文件包含、敏感文件访问、第三方应用漏洞攻击、CC攻击、恶意爬虫扫描、跨站请求伪造等攻击，保护Web服务安全稳定。

WAF仪表盘模板支持[查看WAF安全日志中心](#)和[查看WAF访问日志中心](#)。

前提条件

- 已采集WAF日志，详情请参考[Web应用防火墙WAF接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 WAF 安全日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“WAF仪表盘模板 > WAF安全日志中心”，查看图表详情。

- **CC攻击拦截**图表所关联的查询分析语句如下所示：

```
* |SELECT  
CASE  
  
WHEN  
diff [ 1 ] < 1000 THEN  
concat( cast( cast( diff [ 1 ] as DECIMAL(20,2)) AS VARCHAR ), ' 次' )  
WHEN diff [ 1 ] < 1000 * 1000 THEN  
concat( cast(cast( diff [ 1 ] / 1000 as DECIMAL(20,2)) AS VARCHAR ), ' 千次' )  
WHEN diff [ 1 ] < 1000000000 THEN
```

```
concat( cast( cast( diff [ 1 ]/ 1000000.0 as DECIMAL(20,2)) AS VARCHAR ), ' 百万次' )
WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
concat( cast( cast( diff [ 1 ]/ 1000.0 / 1000000 as DECIMAL(20,2)) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast ( cast( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000 as DECIMAL(20,2)) AS VARCHAR ), ' 万亿
次' )
END AS
"value"

,
CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio
FROM
( SELECT compare ( "data", 86400 ) AS diff FROM ( SELECT count( 1 ) AS "data" FROM log WHERE
attack != 'default' ) )
```

- **攻击峰值**图表所关联的查询分析语句如下所示：

```
* |SELECT diff [ 1 ] AS "VALUE", CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 )
END AS ratio FROM ( SELECT compare ( "DATA", 86400 ) AS diff FROM ( SELECT
COALESCE ( max( pv ), 0 ) AS "DATA" FROM ( SELECT from_unixtime( __time - __time%60000 )
AS dt, CEIL(count(1)/ 60.0) AS pv FROM log WHERE action != " GROUP BY dt LIMIT 10000 )))
```

- **攻击来源国家**图表所关联的查询分析语句如下所示：

```
* |SELECT
diff [ 1 ] AS
"VALUE"
,
COALESCE ( diff [ 1 ]- diff [ 2 ], 0 ) AS "BEFORE"
FROM
(
SELECT
compare ( "DATA", 86400 ) AS diff
FROM
( SELECT count( DISTINCT ip_to_country ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) ) AS
"DATA" FROM log
WHERE action != "
)
)
```

- **Web攻击拦截**图表所关联的查询分析语句如下所示：

```
* |SELECT
CASE
WHEN
diff [ 1 ] < 1000 THEN
concat( cast( cast( diff [ 1 ] as DECIMAL(20,2)) AS VARCHAR ), ' 次' )
WHEN diff [ 1 ] < 1000 * 1000 THEN
concat( cast( cast( diff [ 1 ]/ 1000 as DECIMAL(20,2)) AS VARCHAR ), ' 千次' )
WHEN diff [ 1 ] < 1000000000 THEN
concat( cast( cast( diff [ 1 ]/ 1000000.0 as DECIMAL(20,2)) AS VARCHAR ), ' 百万次' )
WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
concat( cast( cast( diff [ 1 ]/ 1000.0 / 1000000 as DECIMAL(20,2)) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast( cast( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000 as DECIMAL(20,2)) AS VARCHAR ), ' 万亿
次' )
END AS
"value"
,
CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio
FROM
( SELECT compare ( "data", 86400 ) AS diff FROM ( SELECT count( 1 ) AS "data" FROM log WHERE
action != " ) )
```

- **攻击者UV**图表所关联的查询分析语句如下所示：

```
* |SELECT
CASE
WHEN
diff [ 1 ] < 1000 THEN
concat( cast( cast ( diff [ 1 ] AS INTEGER ) AS VARCHAR ), ' 个' )
WHEN diff [ 1 ] < 1000 * 1000 THEN
concat( cast( cast( diff [ 1 ]/ 1000 as DECIMAL(20,2)) AS VARCHAR ), ' 千个' )
WHEN diff [ 1 ] < 1000000000 THEN
concat( cast( cast( diff [ 1 ]/ 1000000.0 as DECIMAL(20,2)) AS VARCHAR ), ' 百万个' )
WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
concat( cast( cast( diff [ 1 ]/ 1000.0 / 1000000 as DECIMAL(20,2)) AS VARCHAR ), ' 十亿' ) ELSE
```

```
concat( cast( cast( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000 as DECIMAL(20,2)) AS VARCHAR ), ' 万亿' )
END AS "value",
CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio
FROM
(
SELECT
compare ( "data", 86400 ) AS diff
FROM
( SELECT count( DISTINCT CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) AS "data"
FROM log
))
```

- **攻击拦截**图表所关联的查询分析语句如下所示：

```
* |SELECT
CASE
WHEN
diff [ 1 ] < 1000 THEN
concat( cast( cast( diff [ 1 ] as DECIMAL(20,2)) AS VARCHAR ), ' 次' )
WHEN diff [ 1 ] < 1000 * 1000 THEN
concat( cast( cast( diff [ 1 ]/ 1000 as DECIMAL(20,2)) AS VARCHAR ), ' 千次' )
WHEN diff [ 1 ] < 1000000000 THEN
concat( cast( cast( diff [ 1 ]/ 1000000.0 as DECIMAL(20,2)) AS VARCHAR ), ' 百万次' )
WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
concat( cast( cast( diff [ 1 ]/ 1000.0 / 1000000 as DECIMAL(20,2)) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast( cast( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000 as DECIMAL(20,2)) AS VARCHAR ), ' 万亿
次' )
END AS
"value",
CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS "ratio"
FROM
(
SELECT
compare ( "data", 86400 ) AS diff
FROM
( SELECT count( 1 ) AS "data" FROM log WHERE action != '' )
)
```

- **CC攻击**图表所关联的查询分析语句如下所示：

```
* |SELECT
ip_to_province (
CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS province,
count( 1 ) AS "攻击次数"
WHERE attack != 'default'
GROUP BY
province
```

- **Web攻击**图表所关联的查询分析语句如下所示：

```
* |SELECT
ip_to_province (
CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS province,
count( 1 ) AS "攻击次数"
WHERE action = 'block'
GROUP BY
province
```

- **攻击类型分布**图表所关联的查询分析语句如下所示：

```
* | SELECT time_format(from_unixtime(__time - __time%60000), 'yyyy-MM-dd HH:mm:ss', '+08:00') AS
dt, count( 1 ) AS cnt, CASE WHEN action = 'block' THEN '拦截' WHEN action = 'log' THEN '仅记录'
WHEN action = 'captcha' THEN '人机验证' END AS attack FROM log WHERE action != '' GROUP BY dt,
attack ORDER BY dt
```

- **CC攻击(世界)**图表所关联的查询分析语句如下所示：

```
* | SELECT
ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS country,
count( 1 ) AS "攻击次数"
WHERE attack != 'default'
GROUP BY
country
```

- **Web攻击(世界)**图表所关联的查询分析语句如下所示：

```
* | SELECT
ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS country,
count( 1 ) AS "攻击次数"
WHERE action = 'block'
GROUP BY
country
```

----结束

查看 WAF 访问日志中心

步骤1 在仪表盘模板下方，选择“WAF仪表盘模板 > WAF访问日志中心”，查看图表详情。

- **PV**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN diff [ 1 ] < 1000 THEN concat( cast( diff [ 1 ] AS
DECIMAL(20,2)) AS VARCHAR ), ' 次' ) WHEN diff [ 1 ] < 1000 * 1000 THEN
concat( cast( cast( diff [ 1 ] / 1000 AS DECIMAL(20,2)) AS VARCHAR ), ' 千次' ) WHEN diff [ 1 ] <
1000000000 THEN concat( cast( cast( diff [ 1 ] / 1000000.0 AS DECIMAL(20,2)) AS VARCHAR ), '
百万次' ) WHEN diff [ 1 ] / 1000.0 < 1000000000 THEN concat( cast( cast( diff [ 1 ] / 1000 /
1000000.0 AS DECIMAL(20,2)) AS VARCHAR ), ' 十亿次' ) ELSE concat( cast( diff [ 1 ] / 1000.0 /
1000 / 1000 / 1000 AS DECIMAL(20,2)) AS VARCHAR ), ' 万亿次' ) END AS
"VALUE" , CASE WHEN diff [ 2 ] = 0 THEN 0 ELSE round( diff [ 3 ] - 1, 2 ) END AS ratio
FROM ( SELECT compare ( DATA, 86400 ) AS diff FROM ( SELECT count( 1 ) AS
DATA FROM log ) )
```

- **UV**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN diff [ 1 ] < 1000 THEN concat( cast( diff [ 1 ] AS
DECIMAL(20,2)) AS VARCHAR ), ' 次' ) WHEN diff [ 1 ] < 1000 * 1000 THEN
concat( cast( cast( diff [ 1 ] / 1000 AS DECIMAL(20,2)) AS VARCHAR ), ' 千次' ) WHEN diff [ 1 ] <
1000000000 THEN concat( cast( cast( diff [ 1 ] / 1000000.0 AS DECIMAL(20,2)) AS VARCHAR ), '
百万次' ) WHEN diff [ 1 ] / 1000.0 < 1000000000 THEN concat( cast( cast( diff [ 1 ] / 1000 /
1000000.0 AS DECIMAL(20,2)) AS VARCHAR ), ' 十亿次' ) ELSE concat( cast( diff [ 1 ] / 1000.0 /
1000 / 1000 / 1000 AS DECIMAL(20,2)) AS VARCHAR ), ' 万亿次' ) END AS
"VALUE" , CASE WHEN diff [ 2 ] = 0 THEN 0 ELSE round( diff [ 3 ] - 1, 2 )
END AS ratio FROM ( SELECT compare ( DATA, 86400 ) AS diff FROM
( SELECT count( DISTINCT CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) AS "DATA" FROM
log ) )
```

- **流入流量**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN diff [ 1 ] < 102 THEN concat( cast( diff [ 1 ] AS
DECIMAL(20,2)) AS VARCHAR ), ' B' ) WHEN diff [ 1 ] < 1024 * 1024 THEN
concat( cast( cast( diff [ 1 ] / 1024 AS DECIMAL(20,2)) AS VARCHAR ), ' KB' ) WHEN diff [ 1 ] <
1024 * 1024 * 1024 THEN concat( cast( cast( diff [ 1 ] / 1024.0 / 1024 AS DECIMAL(20,2)) AS
VARCHAR ), ' MB' ) WHEN diff [ 1 ] / 1024.0 < 1024 * 1024 * 1024 THEN
concat( cast( cast( diff [ 1 ] / 1024.0 / 1024 / 1024 AS DECIMAL(20,2)) AS VARCHAR ), ' GB' ) ELSE
concat( cast( cast( diff [ 1 ] / 1024.0 / 1024 / 1024 / 1024 AS DECIMAL(20,2)) AS VARCHAR ), '
TB' ) END AS "VALUE" , CASE WHEN diff [ 2 ] = 0 THEN 0 ELSE
round( diff [ 3 ] - 1, 2 ) END AS ratio FROM ( SELECT compare ( "DATA", 86400 )
AS diff FROM ( SELECT COALESCE ( sum( try_cast(request_length as bigint) ), 0 ) AS "DATA"
FROM log ) )
```

- **网络in带宽峰值**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN diff [ 1 ] < 102 THEN concat( cast( diff [ 1 ] AS DECIMAL(20,2))
AS VARCHAR ), ' B/s' ) WHEN diff [ 1 ] < 1024 * 1024 THEN concat( cast( cast( diff [ 1 ] / 1024
AS DECIMAL(20,2)) AS VARCHAR ), ' KB/s' ) WHEN diff [ 1 ] < 1024 * 1024 * 1024 THEN
concat( cast( cast( diff [ 1 ] / 1024.0 / 1024 AS DECIMAL(20,2)) AS VARCHAR ), ' MB/s' ) WHEN
diff [ 1 ] / 1024.0 < 1024 * 1024 * 1024 THEN concat( cast( cast( diff [ 1 ] / 1024.0 / 1024 / 1024
AS DECIMAL(20,2)) AS VARCHAR ), ' GB/s' ) ELSE concat( cast( diff [ 1 ] / 1024.0 / 1024 / 1024 /
1024 AS DECIMAL(20,2)) AS VARCHAR ), ' TB/s' ) END AS "VALUE", CASE WHEN diff [ 2 ] = 0
THEN 0 ELSE round( diff [ 3 ] - 1, 2 ) END AS ratio FROM ( SELECT compare ( "DATA",
86400 ) AS diff FROM ( SELECT COALESCE ( max( "DATA" ), 0 ) AS "DATA" FROM
( SELECT time_format(from_unixtime( __time - __time%60000 ), 'yyyy-MM-dd HH:mm:ss', '+08:00')
AS dt, sum( try_cast( request_length AS BIGINT ) ) / 60.0 AS "DATA" FROM log GROUP
BY dt LIMIT 1000 ) ) )
```

- **网络out带宽峰值**图表所关联的查询分析语句如下所示：

```
* | SELECT CASE WHEN diff [ 1 ] < 102 THEN concat( cast( diff [ 1 ] AS DECIMAL(20,2)) AS
VARCHAR ), ' B/s' ) WHEN diff [ 1 ] < 1024 * 1024 THEN concat( cast( diff [ 1 ] / 1024 AS
DECIMAL(20,2)) AS VARCHAR ), ' KB/s' ) WHEN diff [ 1 ] < 1024 * 1024 * 1024 THEN
```

```
concat( cast( cast( diff [ 1 ] / 1024.0 / 1024 AS DECIMAL(20,2)) AS VARCHAR ), ' MB/s' ) WHEN diff  
[ 1 ] / 1024.0 < 1024 * 1024 * 1024 THEN concat( cast( cast( diff [ 1 ] / 1024.0 / 1024 / 1024 AS  
DECIMAL(20,2)) AS VARCHAR ), ' GB/s' ) ELSE concat( cast( cast( diff [ 1 ] / 1024.0 / 1024 / 1024 /  
1024 AS DECIMAL(20,2)) AS VARCHAR ), ' TB/s' ) END AS "value", case when diff [ 2 ] = 0 then 0 else  
round( diff [ 3 ] - 1, 2 ) END AS "ratio" FROM ( SELECT compare ( "DATA", 86400 ) AS diff FROM  
( SELECT COALESCE ( max( bytes_out ), 0 ) AS "DATA" FROM ( SELECT  
time_format(from_unixtime(__time - __time%600000),'yyyy-MM-dd HH:mm:ss','+08:00') AS dt,  
sum( try_cast(body_bytes_sent as double precision) ) / 60.0 AS bytes_out FROM log GROUP BY dt  
LIMIT 10000 )))
```

- **流量带宽趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT time_format(from_unixtime(__time - __time%600000),'yyyy-MM-dd HH:mm:ss','+08:00') AS  
dt, round( sum( try_cast( request_length AS double precision ) ) / 1024.0 / 600, 2 ) AS "流入流量(KB/  
s)", round( sum( try_cast( body_bytes_sent AS double precision ) ) / 1024.0 / 600, 2 ) AS "流出流量  
(KB/s)" GROUP BY dt ORDER BY dt LIMIT 1000
```

- **PV/UV趋势**图表所关联的查询分析语句如下所示：

```
* | SELECT time_format(from_unixtime(__time - __time%3600000),'yyyy-MM-dd HH:mm:ss','+08:00')  
AS dt, count( 1 ) AS PV, COUNT( DISTINCT ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) )  
AS UV FROM log GROUP BY dt ORDER BY dt LIMIT 1000
```

- **访问状态分布**图表所关联的查询分析语句如下所示：

```
* | SELECT time_format(from_unixtime(__time - __time%3600000),'yyyy-MM-dd HH:mm:ss','+08:00')  
AS dt, count( 1 ) AS cnt, concat( cast( cast("response_code" as BIGINT) / 100 AS VARCHAR ), 'XX' ) AS  
"status" GROUP BY dt, cast("response_code" as BIGINT) / 100 ORDER BY dt LIMIT 10000
```

- **访问来源**图表所关联的查询分析语句如下所示：

```
* | SELECT ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS country,  
count( 1 ) AS "访问次数" GROUP BY country having country != "
```

- **流入流量来源（中国）**图表所关联的查询分析语句如下所示：

```
* | SELECT ip_to_province (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS region,  
round( sum( try_cast(request_length as double precision) ) / 1024.0 / 1024, 4 ) AS "流入流量  
(MB)" GROUP BY region having region != "
```

- **流入流量来源（世界）**图表所关联的查询分析语句如下所示：

```
* | SELECT ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS region,  
round( sum( try_cast(request_length as double precision) ) / 1024.0 / 1024, 4 ) AS "流入流量  
(MB)" GROUP BY region having region != "
```

- **来源网络提供商**图表所关联的查询分析语句如下所示：

```
* | SELECT ip_to_provider (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS provider,  
cast(round( sum( try_cast(request_length as bigint) ) / 1024.0 / 1024.0, 3 ) AS double precision) AS  
mb_in GROUP BY provider HAVING ip_to_provider (CASE WHEN sip = '-' THEN  
remote_ip ELSE sip END) != '*' ORDER BY mb_in DESC LIMIT 10
```

- **访问域名**图表所关联的查询分析语句如下所示：

```
* | SELECT http_host, count( 1 ) AS "被访问次数" GROUP BY http_host ORDER  
BY "被访问次数" DESC LIMIT 30
```

- **响应最慢的URL**图表所关联的查询分析语句如下所示：

```
* | SELECT http_host AS "网站",url_extract_path (COALESCE ( url, '/' ) ) AS URL,sum( request_time )/  
count( 1 ) AS "响应时间(毫秒)",count( 1 ) AS "访问次数" GROUP BY http_host, url ORDER BY "响应时  
间(毫秒)" DESC LIMIT 100
```

- **访问最多的客户端**图表所关联的查询分析语句如下所示：

```
* | SELECT ip AS "客户端", client AS "地理网络", concat( cast( ( CASE WHEN pv IS NULL THEN 0 ELSE  
pv END ) AS VARCHAR ), '(', cast( CASE WHEN head_pv is null THEN 0 ELSE ( CASE WHEN head_pv  
> 0 THEN head_pv ELSE 0 END ) END AS VARCHAR ), '/', cast( CASE WHEN get_pv is null THEN 0  
ELSE ( CASE WHEN get_pv > 0 THEN get_pv ELSE 0 END ) END AS VARCHAR ), '/', cast( CASE  
WHEN put_pv is null THEN 0 ELSE ( CASE WHEN put_pv > 0 THEN put_pv ELSE 0 END ) END AS  
VARCHAR ), '/', cast( CASE WHEN post_pv is null THEN 0 ELSE ( CASE WHEN post_pv > 0 THEN  
post_pv ELSE 0 END ) END AS VARCHAR ), '/', cast( CASE WHEN delete_pv is null THEN 0 ELSE  
( CASE WHEN delete_pv > 0 THEN delete_pv ELSE 0 END ) END AS VARCHAR ), '/', ')' ) AS "PV  
(Head, Get, Put, Post, Delete方法)", error_count AS "错误访问次数" FROM ( SELECT ip, client,  
sum( CASE WHEN "method" = 'PUT' AND "status" < 400 THEN pv ELSE 0 END ) AS put_pv,  
sum( CASE WHEN "method" = 'GET' AND "status" < 400 THEN pv ELSE 0 END ) AS get_pv,  
sum( CASE WHEN "method" = 'POST' AND "status" < 400 THEN pv ELSE 0 END ) AS post_pv,  
sum( CASE WHEN "method" = 'DELETE' AND "status" < 400 THEN pv ELSE 0 END ) AS delete_pv,  
sum( CASE WHEN "method" = 'HEAD' AND "status" < 400 THEN pv ELSE 0 END ) AS head_pv,  
sum( throughput ) AS throughput, sum( pv ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE  
0 END ) AS error_count FROM ( SELECT CASE WHEN sip = '-' THEN remote_ip ELSE
```



```
sip      END AS ip, "method", CASE WHEN ip_to_country ( CASE WHEN sip = '-' THEN
remote_ip ELSE sip END )= '香港' THEN '中国香港' WHEN ip_to_province ( CASE WHEN sip =
 '-' THEN remote_ip ELSE sip END )= '*' THEN '未知IP' WHEN ip_to_provider ( CASE WHEN
sip = '-' THEN remote_ip ELSE sip END )= '内网IP' THEN '内网IP' ELSE concat( ip_to_country
( CASE WHEN sip = '-' THEN remote_ip ELSE sip END ), '/', ip_to_province ( CASE WHEN sip = '-' THEN
remote_ip ELSE sip END ), '/', CASE WHEN ip_to_city
```

----结束

7.3.3.19 采集诊断仪表盘模板

采集诊断仪表盘模板支持[查看ICAgent采集监控](#)、[查看ICAgent整体状态](#)和[查看ICAgent异常监控](#)。

前提条件

在LTS控制台配置中心页面的“ICAgent采集开关”页签，开启ICAgent诊断开关，请参考[设置ICAgent日志采集开关](#)。

查看 ICAgent 采集监控

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“采集诊断仪表盘模板 > ICAgent采集监控”，查看图表详情。

- 过滤日志组ID，所关联的查询分析语句如下所示：

```
* | select loggroup from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm'
group by loggroup
```

- 过滤日志流ID，所关联的查询分析语句如下所示：

```
* | select logstream from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm'
group by logstream
```

- 原始数据流量图表所关联的查询分析语句如下所示：

```
* | SELECT
case
  when diff [ 1 ] is null then '0'
  when diff [ 1 ] > 1024 and diff [ 1 ] <= 1024*1024 then concat(cast(round(diff [ 1 ]*1.0/1024,4) as
decimal(12,4)), ' KB')
  when diff [ 1 ] > 1024*1024 and diff [ 1 ] < 1024*1024*1024 then concat(cast(round(diff
[ 1 ]*1.0/1024/1024,4) as decimal(12,4)), ' MB')
  when diff [ 1 ] > 1024*1024*1024 and diff [ 1 ] < cast(1024 as bigint)*1024*1024*1024 then
concat(cast(round(diff [ 1 ]*1.0/1024/1024/1024,4) as decimal(12,4)), ' GB')
  when diff [ 1 ] > cast(1024 as bigint)*1024*1024*1024 then concat(cast(round(diff [ 1 ]*1.0/1024,4)
as decimal(12,4)), ' TB')
  else concat(cast(round(diff [ 1 ]*1.0,4) as decimal(12,4)), ' B')
END AS "原始流量",
case
  when diff [ 3 ] is null then '昨日无数据'
  else cast(cast(round(diff [ 3 ] - 1,2) as decimal(12,2)) as varchar)
END AS "同比昨日"
FROM
(
  SELECT
    report_topic,
    compare ( traffic, 86400 ) AS diff
  FROM
    ( SELECT report_topic, sum( read_bytes ) AS traffic FROM log WHERE report_topic = 'icagent_profile'
    GROUP BY report_topic )
  GROUP BY
    report_topic)
```

- **采集文件数**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "采集文件数", case when diff[3] is not null then cast(cast(round(diff [ 3 ] - 1,2) as decimal(12,2)) as varchar) else '昨日无数据' end as "同比昨天" from (select compare(uv,86400) as diff from (select report_topic,count(distinct concat(file_name,host_ip)) as uv from log where report_topic = 'icagent_profile' group by report_topic) group by report_topic)
```

- **采集机器数/同比昨天**图表所关联的查询分析语句如下所示

```
* | select diff[1] as "采集机器数", case when diff[3] is not null then cast(cast(round(diff [ 3 ] - 1,2) as decimal(12,2)) as varchar) else '昨日无数据' end as "同比昨天" from (select compare(uv,86400) as diff from (select report_topic,count(distinct host_ip) as uv from log where report_topic = 'icagent_profile' group by report_topic) group by report_topic)
```

- **数据发送流量**图表所关联的查询分析语句如下所示：

```
report_topic = "icagent_profile" | SELECT  
"time",  
case  
when traffic is null then 0  
else round(traffic*1.0/1024/1024,2)  
END AS "发送流量 MB"  
from  
(SELECT  
time_floor ( __time, 'PT5M' ) AS "time",  
sum( read_bytes ) AS "traffic"  
FROM  
log  
GROUP BY  
"time") order by "time"
```

- **ICAgent写入次数**图表所关联的查询分析语句如下所示：

```
* | select time_floor(__time,'PT5M') as \"time\",sum(read_count) as \"写入次数\" where report_topic = 'icagent_profile' group by \"time\" order by \"time\"
```

- **采集机器数**图表所关联的查询分析语句如下所示：

```
* | select time_floor(__time,'PT5M') as "time", count(distinct host_ip) as "采集机器数" where report_topic = 'icagent_profile' group by "time" order by "time"
```

- **采集文件分布**图表所关联的查询分析语句如下所示：

```
* | SELECT  
file_name AS "采集路径",  
host_ip AS "IP",  
case  
when traffic is null then '0'  
when traffic > 1024 and traffic <= 1024*1024 then concat(cast(round(traffic*1.0/1024,4) as decimal(12,4)), ' KB')  
when traffic > 1024*1024 and traffic < 1024*1024*1024 then concat(cast(round(traffic*1.0/1024,4) as decimal(12,4)), ' MB')  
when traffic > 1024*1024*1024 and traffic < cast(1024 as bigint)*1024*1024*1024 then concat(cast(round(traffic*1.0/1024,4) as decimal(12,4)), ' GB')  
when traffic > cast(1024 as bigint)*1024*1024*1024 then concat(cast(round(traffic*1.0/1024,2) as decimal(12,4)), ' TB')  
else concat(cast(round(traffic*1.0,4) as decimal(12,4)), ' B')  
END AS "采集流量"  
FROM  
(SELECT  
file_name,  
host_ip,  
sum( read_bytes ) AS "traffic"  
WHERE  
"report_topic" = 'icagent_profile'  
GROUP BY  
file_name,  
host_ip)
```

----结束

查看 ICAgent 整体状态

步骤1 在仪表盘模板下方，选择“采集诊断仪表盘模板 > ICAgent整体状态”，查看图表详情。

- **活跃ICAgent数**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "活跃ICAgent数", case when diff[2] is not null then cast(diff[2] as varchar) else '昨日无数据' end as "昨日活跃ICAgent数" from (select report_topic, compare(uv, 86400) as diff from (SELECT report_topic, COUNT(DISTINCT ip) as uv FROM log where report_topic = 'icagent_status' group by report_topic) group by report_topic)
```
- **发送延迟/次数趋势**图表所关联的查询分析语句如下所示：

```
* | select time_floor(__time, 'PT5M') as "time", sum("metric.lts_cost.below_100_ms") as "below_100_ms", sum("metric.lts_cost.100to500ms") as "100to500ms", sum("metric.lts_cost.500msto1s") as "500msto1s", sum("metric.lts_cost.1sto10s") as "1sto10s", sum("metric.lts_cost.10ston") as "10ston" from log where "report_topic" = 'icagent_status' group by "time" order by "time"
```
- **运行状态分布**图表所关联的查询分析语句如下所示：

```
* | select status, count(DISTINCT ip) as pv from log where report_topic = 'icagent_status' group by status
```
- **CPU趋势**图表所关联的查询分析语句如下所示：

```
* | select ip, time_floor(__time, 'PT5M') as "time", avg("metric.cpu_usage") as "CPU占用率" from log where report_topic = 'icagent_status' and "metric.cpu_usage" is not null group by "time", ip order by "time"
```
- **ICAgent整体状态**图表所关联的查询分析语句如下所示：

```
* | select host_name as "主机名", ip as "IP", version as "版本号", os as "操作系统", MILLIS_TO_TIMESTAMP(ANY_VALUE("metric.start_time")) as "启动时间", avg("metric.cpu_usage") as "CPU", avg("metric.mem_used") * 1.0 / 1024 as "内存(KB)", status as "运行状态" where report_topic = 'icagent_status' group by host_name, ip, version, os, status order by "启动时间" desc
```

----结束

查看 ICAgent 异常监控

步骤1 在仪表盘模板下方，选择“采集诊断仪表盘模板 > ICAgent异常监控”，查看图表详情。

- **过滤日志组ID**，所关联的查询分析语句如下所示：

```
* | select loggroup from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm' group by loggroup limit 10000
```
- **过滤日志流ID**，所关联的查询分析语句如下所示：

```
* | select logstream from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm' group by logstream limit 10000
```
- **关键错误数**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "错误数", case when diff[3] is not null then cast(cast(round(diff[3] - 1, 4) as decimal(12, 4)) as varchar) else '昨日无数据' end as "错误数对比昨天" from (select report_topic, compare(pv, 86400) as diff from (select report_topic, count(1) as pv from log where report_topic = 'icagent_alarm' group by report_topic) group by report_topic)
```
- **丢弃超大行**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "丢弃行数", case when diff[3] is not null then cast(cast(round(diff[3] - 1, 4) as decimal(12, 4)) as varchar) else '昨日无数据' end as "丢弃行数对比昨天" from (select report_topic, compare(pv, 400) as diff from (select report_topic, count(1) as pv from log where report_topic = 'icagent_alarm' and alarm_type = 'DISCARD_BIG_LINE' group by report_topic) group by report_topic)
```
- **请求LTS失败**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "请求失败数", case when diff[3] is not null then cast(cast(round(diff[3] - 1, 4) as decimal(12, 4)) as varchar) else '昨日无数据' end as "请求失败数对比昨天" from (select report_topic, compare(pv, 86400) as diff from (select report_topic, count(1) as pv from log where report_topic = 'icagent_alarm' and alarm_type = 'HTTP_REQUEST_ALARM' group by report_topic) group by report_topic)
```
- **文件超过上限问题数**图表所关联的查询分析语句如下所示：

```
* | select diff[1] as "文件超过上限问题数", case when diff[3] is not null then cast(cast(round(diff[3] - 1, 4) as decimal(12, 4)) as varchar) else '昨日无数据' end as "文件超过上限问题数对比昨天" from (select report_topic, compare(pv, 86400) as diff from (select report_topic, count(1) as pv from log where report_topic = 'icagent_alarm' and alarm_type = 'COLLECT_FILE_EXCEED' group by report_topic) group by report_topic)
```

- **关键错误数(必需处理)**图表所关联的查询分析语句如下所示:

```
* | select MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)) as "最近发生时间", loggroup as "日志组ID", logstream as "日志流ID", host_ip as "IP", alarm_type as "告警类型", os as "系统类型", alarm_message as "告警详情" where report_topic = 'icagent_alarm' group by loggroup, logstream, host_ip, alarm_type, os, alarm_message order by "最近发生时间" desc limit 10000
```

- **丢弃超大行详情**图表所关联的查询分析语句如下所示:

```
* | select MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)) as "最近发生时间", loggroup as "日志组ID", logstream as "日志流ID", host_ip as "IP", alarm_type as "告警类型", os as "系统类型", alarm_message as "告警详情" where report_topic = 'icagent_alarm' and alarm_type = 'DISCARD_BIG_LINE' group by loggroup, logstream, host_ip, alarm_type, os, alarm_message order by "最近发生时间" desc limit 10000
```

- **请求LTS失败详情**图表所关联的查询分析语句如下所示:

```
* | select MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)) as "最近发生时间", loggroup as "日志组ID", logstream as "日志流ID", host_ip as "IP", alarm_type as "告警类型", os as "系统类型", alarm_message as "告警详情" where report_topic = 'icagent_alarm' and alarm_type = 'HTTP_REQUEST_ALARM' group by loggroup, logstream, host_ip, alarm_type, os, alarm_message order by "最近发生时间" desc limit 10000
```

- **文件超过上限问题数详情**图表所关联的查询分析语句如下所示:

```
* | select MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)) as "最近发生时间", loggroup as "日志组ID", logstream as "日志流ID", host_ip as "IP", alarm_type as "告警类型", os as "系统类型", alarm_message as "告警详情" where report_topic = 'icagent_alarm' and alarm_type = 'COLLECT_FILE_EXCEED' group by loggroup, logstream, host_ip, alarm_type, os, alarm_message order by "最近发生时间" desc limit 10000
```

----结束

7.3.3.20 VPC 仪表盘模板

虚拟私有云（Virtual Private Cloud，VPC）是隔离的、私密的虚拟网络环境。用户可以自由配置VPC内的IP地址段、子网、安全组等子服务，也可以申请弹性带宽和弹性公网IP搭建业务系统。VPC日志流中记录了虚拟私有云中的流量信息，可以帮助您检查和优化安全组和网络ACL控制规则、监控网络流量、进行网络攻击分析等。

云日志服务支持日志采集向导一站式采集VPC日志，并为VPC日志配置结构化和仪表盘。该仪表盘主要展示VPC日志的Action总次数、ACCEPT总字节数、ACCEPT总包数、REJECT总字节数、REJECT总包数、源地址的Action次数分布、总分钟Action次数、Action分布、流日志记录状态分布、Action次数的源地址运行商分布、Top5字节数的源地址、Top5字节数的目标地址、Top5包数的目标端口、各协议的每分钟包数、弹性网卡。

前提条件

- 已采集VPC日志，详情请参见[虚拟私有云VPC接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 VPC 流日志

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“日志管理”。

步骤3 在“日志应用”模块中，单击“VPC日志流中心”，选择“进入仪表盘”。

步骤4 在仪表盘模板下方，选择“VPC仪表盘模板 > VPC流日志”，查看图表详情。

- **Action总次数**图表所关联的查询分析语句如下所示:

```
* | select CASE WHEN total_actions < 1000 THEN concat(cast( total_actions AS VARCHAR), '次') WHEN total_actions < 1000 * 1000 THEN concat(cast(round(total_actions / 1000.0, 2) AS VARCHAR), '千次') WHEN total_actions < 1000000000 THEN concat(cast(round(total_actions / 1000000.0, 2) AS VARCHAR), '百万次') WHEN total_actions / 1000.0 < 1000000000 THEN
```

```
concat(cast(round(total_actions / 1000 / 1000000.0, 1) AS VARCHAR), '十亿次') ELSE  
concat(cast(round(total_actions / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR), '万亿次') END AS  
"total_actions" from (select count(1) as total_actions where log_status='OK' and version=1)
```

- **ACCEPT总字节数**图表所关联的查询分析语句如下所示:

```
* | select CASE WHEN accept_bytes < 1024 THEN concat(cast( accept_bytes AS VARCHAR), 'B') WHEN  
accept_bytes < 1024 * 1024 THEN concat(cast(round(accept_bytes / 1024, 2) AS VARCHAR), 'KB')  
WHEN accept_bytes < 1000000000 THEN concat(cast(round(accept_bytes / 1024.0 / 1024, 2) AS  
VARCHAR), 'MB') WHEN accept_bytes / 1000.0 < 1000000000 THEN concat(cast(round(accept_bytes /  
1024 / 1000000.0, 2) AS VARCHAR), 'GB') ELSE concat(cast(round(accept_bytes / 1000.0 / 1000 /  
1000 / 1000, 1) AS VARCHAR), 'TB') END AS "accept_bytes" from (select sum(bytes) as accept_bytes  
where log_status='OK' and version=1 and action='ACCEPT')
```

- **ACCEPT总包数**图表所关联的查询分析语句如下所示:

```
* | select CASE WHEN accept_packets < 1024 THEN concat(cast( accept_packets AS VARCHAR), 'B')  
WHEN accept_packets < 1024 * 1024 THEN concat(cast(round(accept_packets / 1024, 2) AS  
VARCHAR), 'KB') WHEN accept_packets < 1000000000 THEN concat(cast(round(accept_packets /  
1024.0 / 1024, 2) AS VARCHAR), 'MB') WHEN accept_packets / 1000.0 < 1000000000 THEN  
concat(cast(round(accept_packets / 1024 / 1000000.0, 2) AS VARCHAR), 'GB') ELSE  
concat(cast(round(accept_packets / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR), 'TB') END AS  
"accept_packets" from (select sum(packets) as accept_packets where log_status='OK' and version=1  
and action='ACCEPT')
```

- **REJECT总字节数**图表所关联的查询分析语句如下所示:

```
* | select CASE WHEN reject_bytes < 1024 THEN concat(cast( reject_bytes AS VARCHAR), 'B') WHEN  
reject_bytes < 1024 * 1024 THEN concat(cast(round(reject_bytes / 1024, 2) AS VARCHAR), 'KB') WHEN  
reject_bytes < 1000000000 THEN concat(cast(round(reject_bytes / 1024.0 / 1024, 2) AS  
VARCHAR), 'MB') WHEN reject_bytes / 1000.0 < 1000000000 THEN concat(cast(round(reject_bytes /  
1024 / 1000000.0, 2) AS VARCHAR), 'GB') ELSE concat(cast(round(reject_bytes / 1000.0 / 1000 / 1000 /  
1000, 1) AS VARCHAR), 'TB') END AS "reject_bytes" from (select sum(bytes) as reject_bytes where  
log_status='OK' and version=1 and action='REJECT')
```

- **REJECT总包数**图表所关联的查询分析语句如下所示:

```
* | select CASE WHEN reject_packets < 1024 THEN concat(cast( reject_packets AS VARCHAR), 'B')  
WHEN reject_packets < 1024 * 1024 THEN concat(cast(round(reject_packets / 1024, 2) AS  
VARCHAR), 'KB') WHEN reject_packets < 1000000000 THEN concat(cast(round(reject_packets / 1024.0 /  
1024, 2) AS VARCHAR), 'MB') WHEN reject_packets / 1000.0 < 1000000000 THEN  
concat(cast(round(reject_packets / 1024 / 1000000.0, 2) AS VARCHAR), 'GB') ELSE  
concat(cast(round(reject_packets / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR), 'TB') END AS  
"reject_packets" from (select sum(packets) as reject_packets where log_status='OK' and version=1 and  
action='REJECT')
```

- **源地址的Action次数分布**图表所关联的查询分析语句如下所示:

```
* | select IP_TO_PROVINCE(srcaddr) as province, count(1) as total_actions where IP_TO_COUNTRY  
(srcaddr) = '中国' group by province HAVING province not in ('', '保留地址', '*')
```

- **每分钟Action次数**图表所关联的查询分析语句如下所示:

```
* | select TIME_FORMAT(date_trunc('minute', MILLIS_TO_TIMESTAMP("start" * 1000)), 'MM-dd  
HH:mm') as "t", "action", count(1) as "total_actions" where log_status='OK' and version=1 group by  
"t", "action" order by t asc limit 1000
```

- **Action分布**图表所关联的查询分析语句如下所示:

```
* | select action, count(1) as total_actions where log_status='OK' and version=1 group by action
```

- **流日志记录状态分布**图表所关联的查询分析语句如下所示:

```
* | select log_status, count(1) as total_actions where version=1 group by log_status
```

- **Action次数的源地址运营商分布**图表所关联的查询分析语句如下所示:

```
* | select ip_to_provider(srcaddr) as src_addr_provider, count(1) as total_actions where log_status='OK'  
and version=1 group by src_addr_provider order by total_actions desc limit 5
```

- **Top5字节数的源地址**图表所关联的查询分析语句如下所示:

```
* | select srcaddr, sum(bytes) as total_bytes where log_status='OK' and version=1 group by srcaddr  
order by total_bytes desc limit 5
```

- **Top5字节数的目标地址**图表所关联的查询分析语句如下所示:

```
* | select dstaddr, sum(bytes) as total_bytes where log_status='OK' and version=1 group by dstaddr  
order by total_bytes desc limit 5
```

- **Top5包数的目标端口**图表所关联的查询分析语句如下所示:

```
* | select dstport, sum(packets) as total_packets where log_status='OK' and version=1 group by  
dstport order by total_packets desc limit 5
```

- 各协议的每分钟包数图表所关联的查询分析语句如下所示：

```
* | select TIME_FORMAT(date_trunc('minute', MILLIS_TO_TIMESTAMP("start" * 1000)), 'MM-dd HH:mm') as t, protocol, sum(packets) as total_packets where log_status='OK' and version=1 group by t, protocol order by t asc limit 1000
```

- 弹性网卡图表所关联的查询分析语句如下所示：

```
* | select interface_id as "ID", sum(packets) as '数据包总数量', sum(bytes) as '数据包总大小' where log_status='OK' and version=1 group by "ID"
```

----结束

7.3.4 日志仪表盘模板（SQL 分析方式）

7.3.4.1 APIG 仪表盘模板

APIG（API Gateway）提供高性能、高可用、高安全的API托管服务，能快速将企业服务能力包装成标准API服务，帮助您轻松构建、管理和部署任意规模的API，并上架API云商店进行售卖。借助API网关，可以简单、快速、低成本、低风险地实现内部系统集成、业务能力开放及业务能力变现。API网关帮助您变现服务能力的同时，降低企业研发投入，让您专注于企业核心业务，提升运营效率。

APIG仪表盘模板支持[查看APIG访问中心](#)、[查看APIG监控中心](#)、[分析APIG秒级监控](#)。

前提条件

- 已采集APIG日志，详情请参见[API网关APIG接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 APIG 访问中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“APIG仪表盘模板 > APIG访问中心”，查看图表详情。

- 过滤请求域名，所关联的查询分析语句如下所示：

```
select distinct(host)
```

- 过滤app_id，所关联的查询分析语句如下所示：

```
select distinct(app_id)
```

- 访问量PV分布(世界)图表所关联的查询分析语句如下所示：

```
SELECT ip_to_country(my_remote_addr) as country,sum(ori_pv) as PV from (select my_remote_addr, count(1) as ori_pv group by my_remote_addr ORDER BY ori_pv desc LIMIT 10000) GROUP BY country HAVING country not in ('','保留地址','*')
```

- 平均时延分布（中国）所关联的查询分析语句如下所示：

```
SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(my_remote_addr) as province,sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select my_remote_addr, sum(request_time) as rt,count(1) as ori_pv group by my_remote_addr ORDER BY ori_pv desc LIMIT 10000) WHERE IP_TO_COUNTRY (my_remote_addr) = '中国' GROUP BY province ) where province not in ('','保留地址','*')
```

- 平均时延分布（世界）所关联的查询分析语句如下所示：

```
SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(my_remote_addr) as country,sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select my_remote_addr, sum(request_time) as rt,count(1) as ori_pv
```

```
group by my_remote_addr
ORDER BY ori_pv desc
LIMIT 10000) GROUP BY country )
where country not in ('','保留地址','*')
```

- 今日PV/UV所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( __time_, 'yyyy-MM-dd HH:mm:ss' ) as _time_,PV,UV FROM (select
TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') AS _time_ , count(1) as
PV, APPROX_COUNT_DISTINCT(my_remote_addr) as UV from log WHERE __time_ <=
CURRENT_TIMESTAMP and __time_ >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8'
HOUR)) - INTERVAL '8' HOUR group by _time_ order by _time_)
```

- 区域访问TOP10(省份)所关联的查询分析语句如下所示：

```
select ip_to_province(my_remote_addr) as "province", sum(ori_pv) as "访问次数" from(select
my_remote_addr, count(1) as ori_pv
group by my_remote_addr
ORDER BY ori_pv desc
LIMIT 10000)group by "province" HAVING "province" <> '-1' order by "访问次数" desc limit 10
```

- 区域访问TOP10(城市)图表所关联的查询分析语句如下所示：

```
select ip_to_city(my_remote_addr) as "city", sum(ori_pv) as "访问次数" from(select my_remote_addr,
count(1) as ori_pv
group by my_remote_addr
ORDER BY ori_pv desc
LIMIT 10000) group by "city" HAVING "city" <> '-1' order by "访问次数" desc limit 10
```

- Host访问TOP10图表所关联的查询分析语句如下所示：

```
select host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```

- UserAgent访问TOP10图表所关联的查询分析语句如下所示：

```
select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc
limit 10
```

- 设备占比(终端)图表所关联的查询分析语句如下所示：

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC
端' end as type , count(1) as total group by type
```

- 设备占比(系统)图表所关联的查询分析语句如下所示：

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when
regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as
total group by type HAVING type != 'other'
```

- TOP URL图表所关联的查询分析语句如下所示：

```
select router_uri , count(1) as pv, APPROX_COUNT_DISTINCT(my_remote_addr) as UV,
round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group
by router_uri ORDER by pv desc
```

- TOP 访问IP图表所关联的查询分析语句如下所示：

```
select my_remote_addr as "来源IP",ip_to_country(my_remote_addr) as "国家
",ip_to_province(my_remote_addr) as "省份",ip_to_city(my_remote_addr) as "城市
",ip_to_provider(my_remote_addr) as "运营商",count(1) as "PV" group by my_remote_addr ORDER by
"PV" desc limit 100
```

----结束

查看 APIG 监控中心

步骤1 在仪表盘模板下方，选择“APIG仪表盘模板 > APIG监控中心”，查看图表详情。

- 过滤请求域名，所关联的查询分析语句如下所示：

```
select distinct(host)
```

- 过滤app_id，所关联的查询分析语句如下所示：

```
select distinct(app_id)
```

- 访问量PV图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( __time_, 'yyyy-MM-dd HH:mm:ss' ) as _time_,PV FROM ( SELECT TIME_CEIL
( TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT300S' ) AS _time_ , count( 1 ) AS PV
FROM log GROUP BY _time_ )
```

- **请求成功率**图表所关联的查询分析语句如下所示：

```
select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```
- **平均延迟**图表所关联的查询分析语句如下所示：

```
select round(avg(request_time) * 1000, 3) as cnt
```
- **4XX请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
```
- **404请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" = 404
```
- **429请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" = 429
```
- **504请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" = 504
```
- **5XX请求数**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_cnt FROM ( SELECT TIME_CEIL( TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'), 'PT300S' ) AS _time_, count( 1 ) AS cnt FROM log where "status" >= 500 GROUP BY _time_ )
```
- **状态码分布**图表所关联的查询分析语句如下所示：

```
SELECT status, COUNT(1) AS rm GROUP BY status
```
- **访问量UV**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_UV FROM (select TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') AS _time_, APPROX_COUNT_DISTINCT(my_remote_addr) as UV from log group by _time_)
```
- **流量**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss') AS _time_, round( CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量", round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') AS _time_, sum(request_length) / 1024.0 AS "入流量", sum(bytes_sent) / 1024.0 AS "出流量" group by _time_)
```
- **访问失败率**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_, round( CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率", round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') AS _time_, sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率', sum(case when status >= 500 then 1 else 0 end) * 100.0 / COUNT(1) as '5XX比例' group by _time_)
```
- **延迟**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_, round( CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均", round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 ) AS "P50", round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90", round( CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99", round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') as _time_, avg(request_time) * 1000 as "平均", APPROX_QUANTILE_DS("request_time", 0.50)*1000 as "P50", APPROX_QUANTILE_DS("request_time", 0.90)*1000 as "P90", APPROX_QUANTILE_DS("request_time", 0.99)*1000 as "P99", APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as "P9999" group by _time_)
```
- **Host请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT "host", pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( my_remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY "host" ) ORDER BY pv DESC
```
- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)"
```



```
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY "host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示:

```
SELECT "host", pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY "host" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示:

```
SELECT upstream_uri, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT upstream_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( my_remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY upstream_uri ) ORDER BY pv desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示:

```
SELECT upstream_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT upstream_uri, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY upstream_uri ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示:

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT my_remote_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( my_remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY addr having length(my_remote_addr) > 2 ) ORDER BY "pv" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示:

```
SELECT addr, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT my_remote_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " and "my_remote_addr" != " " GROUP BY addr ) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示:

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT my_remote_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
```

```
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " and  
"my_remote_addr" != '-' GROUP BY addr) ORDER BY "访问失败率(%)" desc
```

- **URL延迟TOP图表所关联的查询分析语句如下所示：**

```
SELECT upstream_uri, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0  
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0  
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END,  
3 ) AS "P99延迟(ms)" FROM (SELECT upstream_uri, count( 1 ) AS pv, sum( CASE WHEN "status" <  
400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平  
均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
upstream_uri ) ORDER BY "平均延迟(ms)" desc
```

----结束

查看 APIG 秒级监控

步骤1 在仪表盘模板下方，选择“APIG仪表盘模板 > APIG秒级监控”，查看图表详情。

- 过滤请求域名，所关联的查询分析语句如下所示：

```
select distinct(host)
```

- 过滤app_id，所关联的查询分析语句如下所示：

```
select distinct(app_id)
```

- **QPS图表所关联的查询分析语句如下所示：**

```
SELECT TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss  
ZZ'), 'PT1S'), 'yyyy-MM-dd HH:mm:ss') AS _time_, COUNT(*) as QPS from log group by _time_
```

- **成功率图表所关联的查询分析语句如下所示：**

```
select __time, round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end, 2) as "成功率" from (select  
TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'), 'PT5S'), 'yyyy-MM-  
dd HH:mm:ss') as __time, sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功  
率' from log group by __time)
```

- **延迟图表所关联的查询分析语句如下所示：**

```
select __time, round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end, 2) as "访问延迟"  
, round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end, 2) as "Upstream延迟"  
from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss  
ZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss') as __time, avg(request_time) * 1000 as '访问延迟'  
, avg(upstream_response_time) * 1000 as 'Upstream延迟' from log group by __time)
```

- **流量图表所关联的查询分析语句如下所示：**

```
select __time, round( CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 3 ) AS "请求流量"  
, round( CASE WHEN "返回body流量" > 0 THEN "返回body流量" ELSE 0 END, 3 ) AS "返回body流量"  
from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss  
ZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss') as __time , sum("request_length") / 1024.0 as "请求流量",  
sum("body_bytes_sent") / 1024.0 as "返回body流量" group by __time)
```

- **状态码图表所关联的查询分析语句如下所示：**

```
SELECT TIME_CEIL ( TIME_PARSE ( time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ' ), 'PT5S' ) AS "time",  
SUM( CASE WHEN "status" >= 200 AND "status" < 300 THEN 1 ELSE 0 END ) AS "2XX", SUM( CASE  
WHEN "status" >= 300 AND "status" < 400 THEN 1 ELSE 0 END ) AS "3XX", SUM( CASE WHEN  
"status" >= 400 AND "status" < 500 THEN 1 ELSE 0 END ) AS "4XX", SUM( CASE WHEN "status" >=  
500 AND "status" < 600 THEN 1 ELSE 0 END ) AS "5XX", SUM( CASE WHEN "status" < 200 OR  
"status" >= 600 THEN 1 ELSE 0 END ) AS "其他" FROM log WHERE TIME_PARSE ( time_local,  
'dd/MMM/yyyy:HH:mm:ss ZZ' ) IS NOT NULL GROUP BY "time" ORDER BY "time" ASC LIMIT 100000
```

- **后端响应码图表所关联的查询分析语句如下所示：**

```
SELECT TIME_CEIL ( TIME_PARSE ( time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ' ), 'PT5S' ) AS "time",  
SUM( CASE WHEN "upstream_status" >= 200 AND "upstream_status" < 300 THEN 1 ELSE 0 END ) AS  
"2XX", SUM( CASE WHEN "upstream_status" >= 300 AND "upstream_status" < 400 THEN 1 ELSE 0  
END ) AS "3XX", SUM( CASE WHEN "upstream_status" >= 400 AND "upstream_status" < 500 THEN 1  
ELSE 0 END ) AS "4XX", SUM( CASE WHEN "upstream_status" >= 500 AND "upstream_status" < 600  
THEN 1 ELSE 0 END ) AS "5XX", SUM( CASE WHEN "upstream_status" < 200 OR "upstream_status"  
>= 600 THEN 1 ELSE 0 END ) AS "其他" FROM log WHERE TIME_PARSE ( time_local, 'dd/MMM/  
yyyy:HH:mm:ss ZZ' ) IS NOT NULL GROUP BY "time" ORDER BY "time" ASC LIMIT 100000
```

----结束

7.3.4.2 CCE 仪表盘模板

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群。

CCE仪表盘模板支持[查看CCE日志节点操作](#)、[查看CCE日志K8s对象操作](#)、[查看CCE日志K8s事件查询](#)、[查看CCE日志K8s事件中心](#)、[查看CCE日志聚合检索](#)、[查看CCE日志账号操作审计](#)和[查看CCE日志审计中心](#)。

前提条件

- 已采集CCE日志，详情请参见[云容器引擎CCE应用日志接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CCE 日志节点操作

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志节点操作”，查看图表详情。

- 过滤节点名称，所关联的查询分析语句如下所示：

```
select distinct("objectRef.name")
```
- 过滤操作用户，所关联的查询分析语句如下所示：

```
select distinct("user.username")
```
- 过滤状态码，所关联的查询分析语句如下所示：

```
select distinct("responseStatus.code")
```
- 过滤操作类型，所关联的查询分析语句如下所示：

```
select distinct("verb")
```
- 节点数趋势图表所关联的查询分析语句如下所示：

```
SELECT time_series( TIME_PARSE(LEFT(requestReceivedTimestamp, 23),'yyyy-MM-dd"T"HH:mm:ss.SSS'), 'PT1H', 'yyyy-MM-dd HH', '0' ) as "dt", count(DISTINCT("objectRef.name")) as "节点数" where "objectRef.resource" = 'nodes' and "objectRef.subresource" = 'status' and "verb" in ('update', 'patch') and "user.username" = 'system:node' group by "dt" order by "dt" desc limit 10000
```
- 非系统用户操作趋势图表所关联的查询分析语句如下所示：

```
SELECT time_series( TIME_PARSE(LEFT(requestReceivedTimestamp, 23),'yyyy-MM-dd"T"HH:mm:ss.SSS'), 'PT1H', 'yyyy-MM-dd HH', '0' ) as "dt", count(*) as "请求", "user.username" where "objectRef.resource" = 'nodes' and "user.username" not in ( 'kube-controller-manager','kube-apiserver-kubelet-client','apiserver' ) and "user.username" not like 'system:%' and "verb" in ('create','delete','update','patch') group by "dt", "user.username" order by "dt","请求" desc limit 10000
```
- create操作状态码分布图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'create' group by "状态码"
```
- delete操作状态码分布图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'delete' group by "状态码"
```
- patch操作状态码分布图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'patch' group by "状态码"
```
- update操作状态码分布图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'update' group by "状态码"
```
- 节点封锁/解除封锁操作状态码分布图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "requestObject" in ( '{"spec":{"unschedulable":false}}', '{"spec":{"unschedulable":true}}' ) group by "状态码"
```

- **Label操作状态码分布图表**所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where  
"objectRef.resource" = 'nodes' and "verb" in ('patch','update') and "requestObject" = 'labels' and  
"requestObject" = 'metadata' group by "状态码"
```
- **Taint操作状态码分布图表**所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where  
"objectRef.resource" = 'nodes' and "verb" in ('patch','update') and "requestObject" = 'taints' group by  
"状态码"
```
- **驱逐操作状态码分布图表**所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where  
"objectRef.subresource" = 'eviction' and "objectRef.resource" = 'pods' and "verb" = 'create' group by "状  
态码"
```
- **节点增删操作列表图表**所关联的查询分析语句如下所示：

```
select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "verb" AS "操作动作",  
"stageTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码"  
where "objectRef.resource" = 'nodes' and "verb" in ('create','delete')
```
- **Taint操作列表图表**所关联的查询分析语句如下所示：

```
select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "requestObject" AS "Taints",  
"requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code"  
AS "状态码" where "objectRef.resource" = 'nodes' and "verb" = 'patch' and "requestObject" = 'taints'
```
- **驱逐操作列表图表**所关联的查询分析语句如下所示：

```
select "auditID" AS "Audit ID", "objectRef.name" AS "pod", "sourceIPs" AS "源地址",  
"requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code"  
AS "状态码" where "objectRef.resource" = 'pods' and "verb" = 'create' and "objectRef.subresource" =  
'eviction'
```
- **Label操作列表图表**所关联的查询分析语句如下所示：

```
select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "requestObject" AS "Label",  
"requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code"  
AS "状态码" where "objectRef.resource" = 'nodes' and "verb" = 'patch' and "requestObject" = 'labels'
```
- **封锁操作列表图表**所关联的查询分析语句如下所示：

```
select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "requestReceivedTimestamp" AS "操作  
时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "verb" = 'patch' and  
"objectRef.resource" = 'nodes' and "requestObject" = 'true' and "requestObject" = 'unschedulable'
```
- **取消封锁操作列表图表**所关联的查询分析语句如下所示：

```
select "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "requestReceivedTimestamp" AS "操作  
时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "verb" = 'patch' and  
"objectRef.resource" = 'nodes' and "requestObject" not in ('true','taints','unschedulable')
```

----结束

查看 CCE 日志 K8s 对象操作

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志K8s对象操作”，查看图表详情。

- 过滤命名空间，所关联的查询分析语句如下所示：

```
select distinct("objectRef.namespace")
```
- 过滤操作类型，所关联的查询分析语句如下所示：

```
select distinct("verb")
```
- 过滤状态码，所关联的查询分析语句如下所示：

```
select distinct("responseStatus.code")
```
- 过滤资源对象，所关联的查询分析语句如下所示：

```
select distinct("objectRef.name")
```
- 过滤资源类型，所关联的查询分析语句如下所示：

```
select distinct("objectRef.resource")
```
- 过滤操作用户，所关联的查询分析语句如下所示：

```
select distinct("user.username")
```

- **重要操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(requestReceivedTimestamp, 16), 'T', ' ') AS "dt", "verb" as "操作类型", count(*) as "count" where "verb" in ('create','delete','update','patch') and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "dt", "操作类型" order by "dt" limit 10000
```
- **非系统用户操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(requestReceivedTimestamp, 16), 'T', ' ') AS "dt", count(*) as "请求次数", "user.username" WHERE "user.username" not in ('kube-controller-manager','kube-apiserver-kubelet-client','apiserver') and "user.username" not like 'system:%' and "verb" in ('create','delete','update','patch') and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','secrets','pvc's') group by "dt", "user.username" limit 10000
```
- **create操作资源类型分布**图表所关联的查询分析语句如下所示：

```
select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```
- **delete操作资源类型分布**图表所关联的查询分析语句如下所示：

```
select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```
- **update操作资源类型分布**图表所关联的查询分析语句如下所示：

```
select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```
- **patch操作资源类型分布**图表所关联的查询分析语句如下所示：

```
select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```
- **create操作用户分布**图表所关联的查询分析语句如下所示：

```
select "user.username" as "操作用户", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```
- **delete操作用户分布**图表所关联的查询分析语句如下所示：

```
select "user.username" as "操作用户", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```
- **update操作用户分布**图表所关联的查询分析语句如下所示：

```
select "user.username" as "操作用户", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"select "user.username" as "操作用户", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```
- **patch操作用户分布**图表所关联的查询分析语句如下所示：

```
select "user.username" as "操作用户", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```
- **create操作状态码分布**图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```

- **delete操作状态码分布**图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```
- **update操作状态码分布**图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```
- **patch操作状态码分布**图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```
- **create操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(stageTimestamp, 16), 'T', ' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```
- **delete操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(stageTimestamp, 16), 'T', ' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```
- **update操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(stageTimestamp, 16), 'T', ' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```
- **patch操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(stageTimestamp, 16), 'T', ' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```

----结束

查看 CCE 日志 K8s 事件查询

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志K8s对象操作”，查看图表详情。

- 过滤命名空间，所关联的查询分析语句如下所示：

```
select distinct("objectRef.namespace")
```
- 过滤操作类型，所关联的查询分析语句如下所示：

```
select distinct("verb")
```
- 过滤状态码，所关联的查询分析语句如下所示：

```
select distinct("responseStatus.code")
```
- 过滤资源对象，所关联的查询分析语句如下所示：

```
select distinct("objectRef.name")
```
- 过滤资源类型，所关联的查询分析语句如下所示：

```
select distinct("objectRef.resource")
```
- 过滤操作用户，所关联的查询分析语句如下所示：

```
select distinct("user.username")
```
- **重要操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(requestReceivedTimestamp, 16), 'T', ' ') AS "dt", "verb" as "操作类型", count(*) as "count" where "verb" in ('create','delete','update','patch') and "objectRef.resource" in
```

```
('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "dt","操作类型" order by "dt" limit 10000
```

- **非系统用户操作趋势图表**所关联的查询分析语句如下所示:

```
SELECT REPLACE(LEFT(requestReceivedTimestamp, 16),'T',' ') AS "dt", count(*) as "请求次数", "user.username" WHERE "user.username" not in ('kube-controller-manager','kube-apiserver-kubelet-client','apiserver') and "user.username" not like 'system:%' and "verb" in ('create','delete','update','patch') and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','secrets','pvc') group by "dt", "user.username" limit 10000
```

- **create操作资源类型分布图表**所关联的查询分析语句如下所示:

```
select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```

- **delete操作资源类型分布图表**所关联的查询分析语句如下所示:

```
select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```

- **update操作资源类型分布图表**所关联的查询分析语句如下所示:

```
select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```

- **patch操作资源类型分布图表**所关联的查询分析语句如下所示:

```
select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
```

- **create操作用户分布图表**所关联的查询分析语句如下所示:

```
select "user.username" as "操作用户", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```

- **delete操作用户分布图表**所关联的查询分析语句如下所示:

```
select "user.username" as "操作用户", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```

- **update操作用户分布图表**所关联的查询分析语句如下所示:

```
select "user.username" as "操作用户", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"select "user.username" as "操作用户", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```

- **patch操作用户分布图表**所关联的查询分析语句如下所示:

```
select "user.username" as "操作用户", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "user.username"
```

- **create操作状态码分布图表**所关联的查询分析语句如下所示:

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```

- **delete操作状态码分布图表**所关联的查询分析语句如下所示:

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```

- **update操作状态码分布**图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```
- **patch操作状态码分布**图表所关联的查询分析语句如下所示：

```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "responseStatus.code"
```
- **create操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```
- **delete操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```
- **update操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```
- **patch操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by dt, "objectRef.resource" order by dt limit 10000
```

----结束

查看 CCE 日志 K8s 事件中心

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志K8s事件中心”，查看图表详情。

- 事件等级分为Warning和Normal。
- 过滤事件类型，所关联的查询分析语句如下所示：

```
select distinct("name")
```
- 过滤集群ID，所关联的查询分析语句如下所示：

```
select distinct("cluster_id")
```
- 过滤命名空间，所关联的查询分析语句如下所示：

```
select distinct("namespace")
```
- 过滤名称，所关联的查询分析语句如下所示：

```
select distinct("resource_name")
```
- **Conntrack Full**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='ConntrackFull' ) )
```
- **事件同步异常**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"='NTPIsDown') )
```
- **节点Pid不足**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where "name" in ('PIDPressure','NodeHasPIDPressure') ) )
```


- **节点FD不足**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"=  
'NodeHasFDPressure' ) )
```
- **节点磁盘空间不足**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"=  
'NodeHasDiskPressure' ) )
```
- **Pod OOM**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( "total", 3600) as diff from( select count(1) as "total" from log where "reason" in  
( 'OOMKilling','PodOOMKilling' ) )
```
- **DockerHung**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( "total", 3600) as diff from( select count(1) as "total" from log where "name" = 'Failed' and  
"reason" = 'DockerHung' ) )
```
- **节点重启**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( "total", 3600) as diff from( select count(1) as "total" from log where "name"=  
'NodeRebooted' ) )
```
- **镜像拉取失败**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( "total", 3600) as diff from( select count(1) as "total" from log where "name" = 'Failed' and  
"reason" = 'ImagePullBackOff' ) )
```
- **节点OOM**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( "total", 3600) as diff from( select count(1) as "total" from log where "name" =  
'SystemOOM' ) )
```
- **Pod启动失败**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( "total", 3600) as diff from( select count(1) as "total" from log where "name" = 'Failed' and  
"resource_kind" = 'Pod' and "reason" = 'ImagePullBackOff' ) )
```
- **事件分布**图表所关联的查询分析语句如下所示：

```
select "type", count(*) as "事件数" group by "type"
```
- **Warning事件趋势**图表所关联的查询分析语句如下所示：

```
select time_series( __time, 'PT1H', 'yyyy-MM-dd HH', '0') as "dt", count(1) as "count" from log where  
"type" = 'Warning' group by "dt" order by "dt"
```
- **Error事件趋势**图表所关联的查询分析语句如下所示：

```
select time_series( __time, 'PT1H', 'yyyy-MM-dd HH', '0') as "dt", count(1) as "count" from log where  
"type" = 'Error' group by "dt" order by "dt"
```
- **Pod OOM事件列表**所关联的查询分析语句如下所示：

```
select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "Time", "resource_kind" as "事件  
目标", "name" as "类型", "resource_name" as "目标名", "reason" as "详细内容" from log where  
"name" in ( 'OOMKilling','PodOOMKilling' ) order by __time desc limit 100
```
- **Pod驱动事件列表**所关联的查询分析语句如下所示：

```
select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "Time", "resource_kind" as "事件  
目标", "name" as "类型", "resource_name" as "目标名", "reason" as "详细内容" from log where  
"name" = 'NodeControllerEviction' order by __time desc limit 100
```
- **重要事件列表**所关联的查询分析语句如下所示：

```
select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "Time", "type" as "等级",  
"resource_kind" as "事件目标", "name" as "类型", "resource_name" as "目标名", "reason" as "详细内  
容" from log where "type" in ( 'Warning','Error' ) order by __time desc limit 100
```

----结束

查看 CCE 日志聚合检索

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志聚合检索”，查看图表详情。

- 过滤命名空间，所关联的查询分析语句如下所示：
`select distinct("objectRef.namespace")`
- 过滤操作用户，所关联的查询分析语句如下所示：
`select distinct("user.username")`
- 过滤状态码，所关联的查询分析语句如下所示：
`select distinct("responseStatus.code")`
- 过滤操作类型，所关联的查询分析语句如下所示：
`select distinct("verb")`
- 过滤资源对象，所关联的查询分析语句如下所示：
`select distinct("objectRef.name")`
- 过滤资源类型，所关联的查询分析语句如下所示：
`select distinct("objectRef.resource")`
- 过滤请求URL，所关联的查询分析语句如下所示：
`select distinct("requestURI")`
- 过滤userAgent，所关联的查询分析语句如下所示：
`select distinct("userAgent")`
- 操作用户分布趋势图表所关联的查询分析语句如下所示：
`SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "user.username" as "操作用户", count(*) as "count" group by dt, "user.username" order by dt limit 10000`
- 命名空间分布趋势图表所关联的查询分析语句如下所示：
`SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.namespace" as "命名空间", count(*) as "count" group by dt, "objectRef.namespace" order by dt limit 10000`
- 操作类型分布趋势图表所关联的查询分析语句如下所示：
`SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.namespace" as "命名空间", count(*) as "count" group by dt, "objectRef.namespace" order by dt limit 10000`
- 状态码分布趋势图表所关联的查询分析语句如下所示：
`SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, cast("responseStatus.code" as varchar) as "返回码", count(*) as "count" group by dt, "返回码" order by dt limit 10000`
- 资源类型分布趋势图表所关联的查询分析语句如下所示：
`SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" group by dt, "objectRef.resource" order by dt limit 10000 SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" group by dt, "objectRef.resource" order by dt limit 10000`
- 重要操作列表所关联的查询分析语句如下所示：
`select "auditID" AS "Audit ID", "verb" AS "操作类型", "requestReceivedTimestamp" AS "开始时间", "stageTimestamp" AS "结束时间", "user.username" AS "操作账号", "sourceIPs" AS "操作源", "userAgent", "objectRef.namespace" AS "命名空间", CONCAT("objectRef.resource", '/'), "objectRef.subresource") AS "操作对象", "objectRef.name" AS "资源名", "responseStatus.code" AS "返回码"`

----结束

查看 CCE 日志账号操作审计

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志账号操作审计”，查看图表详情。

- 过滤用户名，所关联的查询分析语句如下所示：
`select distinct("user.username")`
- 过滤命名空间，所关联的查询分析语句如下所示：
`select distinct("objectRef.namespace")`

- 过滤状态码，所关联的查询分析语句如下所示：

```
select distinct("responseStatus.code")
```
- 资源创建数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total, 86400) as diff from( select count(1) as total from log where "verb" = 'create' ) )
```
- 资源修改数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total, 86400) as diff from( select count(*) as "total" from log where "verb" in  
( 'update','patch' ) )
```
- 资源删除数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total, 86400) as diff from( select count(*) as "total" from log where "verb" = 'delete' ) )
```
- 操作命名空间分布图表所关联的查询分析语句如下所示：

```
select case when "objectRef.namespace" is null then '_all_' else "objectRef.namespace" end as ns,  
count(1) as total group by ns limit 10000
```
- 删除资源分布图表所关联的查询分析语句如下所示：

```
SELECT "objectRef.resource" as "resource", count(1) as "count" where "verb" = 'delete' group by  
"resource"
```
- 操作轨迹图表所关联的查询分析语句如下所示：

```
select case when "操作" is null then '无' else "操作" end as "操作", "时间", v from (select  
concat(CASE WHEN "objectRef.subresource" is null then "objectRef.resource" else  
"objectRef.subresource" end, '[', verb, ']') as "操作", time_series(__time, 'PT1H', 'yyyy-MM-dd HH',  
'0') as "时间", count(1) as v from log where "verb" in ('create', 'patch', 'update', 'delete') group by  
"操作", "时间" order by "时间" desc limit 10000 )
```
- 资源操作分布图表所关联的查询分析语句如下所示：

```
select CASE WHEN "objectRef.subresource" is null then "objectRef.resource" else  
"objectRef.subresource" end as "资源", verb as "操作", count(1) as total where "verb" in  
( 'create','update','patch','delete' ) group by "资源", "操作" limit 10000
```
- 创建资源列表所关联的查询分析语句如下所示：

```
SELECT "auditID" as "事件ID", time_format("__time", 'yyyy-MM-dd HH:mm:ss') as "操作时间",  
"requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码", "sourceIPs"  
as "源地址", "requestObject" as "详细内容" where "verb" = 'create' order by __time desc limit 1000
```
- 修改资源列表所关联的查询分析语句如下所示：

```
SELECT auditID as "事件ID", time_format("__time", 'yyyy-MM-dd HH:mm:ss') as "操作时间",  
"requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码", "sourceIPs"  
as "源地址", requestObject as "详细内容" where "verb" in ('update','patch') order by __time desc limit  
1000
```
- 资源访问列表所关联的查询分析语句如下所示：

```
SELECT auditID as "事件ID", time_format("__time", 'yyyy-MM-dd HH:mm:ss') as "操作时间",  
"requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码", "sourceIPs"  
as "源地址", requestObject as "详细内容" where "verb" in ('get','list') order by __time desc limit 1000
```
- 资源删除列表所关联的查询分析语句如下所示：

```
SELECT auditID as "事件ID", time_format("__time", 'yyyy-MM-dd HH:mm:ss') as "操作时间",  
"requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码", "sourceIPs"  
as "源地址", requestObject as "详细内容" where "verb" = 'delete' order by __time desc limit 1000
```

----结束

查看 CCE 日志审计中心

步骤1 在仪表盘模板下方，选择“CCE仪表盘模板 > CCE日志审计中心”，查看图表详情。

- 过滤命名空间，所关联的查询分析语句如下所示：

```
select distinct("objectRef.namespace")
```
- 过滤操作用户，所关联的查询分析语句如下所示：

```
select distinct("user.username")
```
- 过滤操作类型，所关联的查询分析语句如下所示：

```
select distinct("verb")
```

- 过滤状态码，所关联的查询分析语句如下所示：

```
select distinct("responseStatus.code")
```
- 过滤资源对象，所关联的查询分析语句如下所示：

```
select distinct("objectRef.name")
```
- 过滤资源类型，所关联的查询分析语句如下所示：

```
select distinct("objectRef.resource")
```
- 过滤请求URL，所关联的查询分析语句如下所示：

```
select distinct("requestURI")
```
- 过滤UserAgent，所关联的查询分析语句如下所示：

```
select distinct("userAgent")
```
- 总审计记录数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total , 86400) as diff from( select count(1) as total from log ) )
```
- 操作用户数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total , 86400) as diff from( select count(distinct("user.username")) as total from log ) )
```
- 活跃节点数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total , 86400) as diff from( select count(DISTINCT "user.username") as total from log  
where "objectRef.resource" = 'nodes' and "objectRef.subresource" = 'status' and "verb" in  
( 'update','put','patch' ) and "user.username" in ( 'node','system' ) ) )
```
- 异常访问次数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total , 86400) as diff from( select count(1) as total from log where "responseStatus.code"  
>= 400) )
```
- 敏感操作次数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total , 86400) as diff from( select count(1) as "total" from log where ("verb" = 'create'  
AND "objectRef.subresource" = 'exec') OR ("verb" = 'create' AND "objectRef.subresource" = 'attach'  
AND "objectRef.resource" = 'pods') OR ("objectRef.resource" = 'secrets' AND "verb" = 'get' AND  
( "user.username" != 'apiserver') AND ("user.username" not like 'system:node:%') OR ("verb" =  
'delete' AND ( "user.username" not like 'system:node:%') AND ( "user.username" not like  
'system:serviceaccount:kube-system:%') AND ( "user.username" != 'system:apiserve') AND  
( "user.username" != 'system:apiserve') AND ( "user.username" != 'system:kube-scheduler') AND  
( "user.username" != 'system:kube-controller-manager')) ) )
```
- 创建操作次数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total , 86400) as diff from( select count(1) as total from log where verb = 'create' ) )
```
- 更新操作次数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total , 86400) as diff from( select count(1) as total from log where verb in  
( 'update','patch' ) ) )
```
- 删除操作次数图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as "inc" from (select  
compare( total , 86400) as diff from( select count(1) as total from log where verb = 'delete' ) )
```
- 操作用户分布图表所关联的查询分析语句如下所示：

```
select "user.username" as "用户名", count(*) as "count" group by "用户名" order by "count" desc
```
- 命名空间分布图表所关联的查询分析语句如下所示：

```
select "objectRef.namespace" as "命名空间", count(*) as "count" group by "命名空间"
```
- 资源类型分布图表所关联的查询分析语句如下所示：

```
select "objectRef.resource" as "资源类型", count(*) as "count" group by "资源类型" order by "count"  
desc limit 20
```
- 操作类型分布图表所关联的查询分析语句如下所示：

```
select verb as "操作类型", count(*) as "count" group by "操作类型" order by "count" desc
```
- 节点操作分布图表所关联的查询分析语句如下所示：

```
select "verb" as "操作类型", count(*) as "count" where "objectRef.resource" = 'nodes' AND ("verb" in ('create','delete')) group by "操作类型" order by "count" desc
```

- **工作负载操作分布**图表所关联的查询分析语句如下所示：

```
select "verb" as "操作类型", count(*) as "count" where "verb" in ('create', 'delete') and "objectRef.resource" in ('deployments','statefulsets','daemonsets','jobs','cronjobs') group by "操作类型" order by "count" desc
```
- **Service/Ingress操作分布**图表所关联的查询分析语句如下所示：

```
select "verb" as "操作类型", count(*) as "count" where "verb" in ('create', 'delete') and "objectRef.resource" in ('ingresses','services') group by "verb" order by "count" desc
```
- **重要操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT("stageTimestamp", 16),'T',' ') AS "dt", "verb", count(*) as "count" where "verb" in ('create','delete','update','patch') group by "dt", "verb" order by "dt" limit 10000
```
- **非系统用户操作趋势**图表所关联的查询分析语句如下所示：

```
SELECT REPLACE(LEFT("stageTimestamp", 16),'T',' ') AS "dt", count(*) as "count", "user.username" as "用户名称" where "user.username" not in ('kube-controller-manager','kube-apiserver-kubelet-client','system','apiserver') group by "dt", "用户名称" order by "dt" limit 10000
```

----结束

7.3.4.3 CDN 仪表盘模板

CDN（Content Delivery Network，内容分发网络）记录了所有域名（包括已删除域名，如果您开通了企业项目，则已删除域名不支持此功能）被网络用户访问的详细信息，您可以将日志接入LTS，对您的业务资源被访问情况进行详细分析。

CDN仪表盘模板支持[查看CDN错误分析](#)、[查看CDN基础数据](#)、[查看CDN用户分析](#)和[查看CDN热门资源](#)。

前提条件

- 已采集CDN日志，详情请参见[内容分发网络CDN接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CDN 错误分析

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“CDN仪表盘模板 > CDN错误分析”，查看图表详情。

- **错误域名访问Top5**图表所关联的查询分析语句如下所示：

```
select domain , count(*) as c where http_code > 400 group by domain order by c desc limit 5
```
- **错误URI访问Top5**图表所关联的查询分析语句如下所示：

```
select uri , count(*) as c where http_code > 400 group by uri order by c desc limit 5
```
- **错误请求状态分布**图表所关联的查询分析语句如下所示：

```
select http_code , count(*) as c where http_code > 400 group by http_code order by c desc
```
- **错误按运营商统计**图表所关联的查询分析语句如下所示：

```
select ip_to_provider(client_ip) as isp , count(*) as c where http_code > 400 group by isp having ip_to_provider(client_ip) != '' order by c desc limit 10
```
- **错误按客户端统计**图表所关联的查询分析语句如下所示：

```
select user_agent as "客户端版本", count(*) as "错误次数" where http_code > 400 group by user_agent order by "错误次数" desc limit 10
```
- **错误按省份统计**图表所关联的查询分析语句如下所示：

```
select ip_to_province(client_ip) as province , count(*) as c where http_code > 400 and IP_TO_COUNTRY (client_ip) = '中国' group by province order by c desc limit 50
```

- **4XX错误详情**图表所关联的查询分析语句如下所示：

```
SELECT
  province AS "省份",
  isp AS "运营商",
  c AS "错误次数",
  round( c * 100.0 / sum( c ), 2 ) AS "错误比率(%)"
FROM
  (
    SELECT
      ip_to_province ( client_ip ) AS province,
      ip_to_provider ( client_ip ) AS isp,
      count(*) AS c
    FROM
      log
    WHERE
      http_code >= 400
      AND http_code < 500
    GROUP BY
      province,
      isp
    HAVING
      (
        ip_to_provider ( client_ip )) != "
    ORDER BY
      c DESC
  )
GROUP BY
  province,
  isp,
  c
```

- **5XX错误详情**图表所关联的查询分析语句如下所示：

```
SELECT
  province AS "省份",
  isp AS "运营商",
  c AS "错误次数",
  round( c * 100.0 / sum( c ), 2 ) AS "错误比率(%)"
FROM
  (
    SELECT
      ip_to_province ( client_ip ) AS province,
      ip_to_provider ( client_ip ) AS isp,
      count(*) AS c
    FROM
      log
    WHERE
      http_code >= 500
    GROUP BY
      province,
      isp
    HAVING
      (
        ip_to_provider ( client_ip )) != "
    ORDER BY
      c DESC
  )
GROUP BY
  province,
  isp,
  c
```

- **错误按国家统计**图表所关联的查询分析语句如下所示：

```
select ip_to_country(client_ip) as country , count(*) as c where http_code > 400 group by country
order by c desc limit 50
```

----结束

查看 CDN 基础数据

步骤1 在仪表盘模板下方，选择“CDN仪表盘模板 > CDN基础数据”，查看图表详情。

- **缓存命中率**图表所关联的查询分析语句如下所示：

```
select round(diff[1],2) as Hit_ratio, round(diff[2],2) as diff, round((diff[3]-1)*100, 2) from (select
compare(Hit_ratio, 86400) as diff from (select sum(s) * 100.0/count(*) as Hit_ratio from (select case
when hit_info = 'HIT' then 1 else 0 end as s from log)))
```

- **下载速度**图表所关联的查询分析语句如下所示：

```
select round(diff[1],2) as speed, round(diff[2],2) as diff, round((diff[3]-1)*100, 2) from (select
compare(speed, 86400) as diff from (select sum(response_size) * 1.0 /sum(response_time) as speed
from log ))
```

- **访问状态**图表所关联的查询分析语句如下所示：

```
select http_code , count(*) as c group by http_code order by c desc
```

- **访问延时分布**图表所关联的查询分析语句如下所示：

```
select
case when response_time < 100 then '~100ms'
when response_time < 500 then '100~500ms'
when response_time < 1000 then '500ms~1s'
when response_time < 5000 then '1~5s'
when response_time < 6000 then '5~6s'
when response_time < 7000 then '6~7s'
when response_time < 8000 then '7~8s'
when response_time < 10000 then '8~10s'
when response_time < 15000 then '10~15s'
else '15s~' end as latency ,
count(*) as cnt
group by latency
order by cnt
```

- **请求带宽**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT (TIME_FLOOR(__time,'PT1M'), 'HH:mm', '+08:00') as thisdate,
sum(response_size) * 8/1000000000.0 as "带宽Gbit/min"
group by TIME_FLOOR(__time,'PT1M')
order by TIME_FLOOR(__time,'PT1M')
```

- **访问次数/人数**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT (TIME_FLOOR(__time,'PT1M'), 'HH:mm', '+08:00') as thisdate,
count(*) as pv, APPROX_COUNT_DISTINCT(client_ip) as uv group by TIME_FLOOR(__time,'PT1M')
order by TIME_FLOOR(__time,'PT1M')
```

- **访问平均延时**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT (TIME_FLOOR(__time,'PT1M'), 'HH:mm', '+08:00') as thisdate,
avg(response_time) as "平均延时(ms)" group by TIME_FLOOR(__time,'PT1M') order by
TIME_FLOOR(__time,'PT1M')
```

- **请求命中率**图表所关联的查询分析语句如下所示：

```
select
TIME_FORMAT (TIME_FLOOR(m_time,'PT1M'), 'HH:mm', '+08:00') as thisdate ,
sum(is_hit)*100.0/count(*) as hit_ratio
from (select TIME_FLOOR(__time,'PT1M') as m_time , case when hit_info = 'HIT'
then 1 else 0 end as is_hit from log ) group by m_time order by m_time
```

----结束

查看 CDN 用户分析

步骤1 在仪表盘模板下方，选择“CDN仪表盘模板 > CDN用户分析”，查看图表详情。

- **访问次数**图表所关联的查询分析语句如下所示：

```
select diff[1] as pv,diff[2] as diff, round(100*(diff[3]-1), 2) from (select compare(pv, 86400) as diff
from (select count(*) as pv from log))
```

- **访问人数**图表所关联的查询分析语句如下所示：

```
select diff[1] as uv, diff[2] as diff, round((diff[3]-1)*100, 2) from (select compare(uv , 86400) as diff
from (select APPROX_COUNT_DISTINCT(client_ip) as uv from log))
```

- **访问客户端统计图表所关联的查询分析语句如下所示：**

```
select ua as "客户端", sum(c) as "访问次数" from (select case when strpos(ua, 'iphone') > 1 then 'iphone' when strpos(ua, 'ipad') > 1 then 'ipad' when strpos(ua, 'android') > 1 then 'android' when strpos(ua, 'windows') > 1 then 'windows' when strpos(ua, 'mac') > 1 then 'mac' when strpos(ua, 'linux') > 1 then 'linux' else ua end as ua, c from (select count(*) as c, lower(user_agent) as ua from log group by ua order by c desc limit 2000) ) group by "客户端" order by "访问次数" desc limit 100
```
- **运营商次数统计图表所关联的查询分析语句如下所示：**

```
select ip_to_provider(client_ip) as isp ,count(*) as "访问次数" group by isp order by "访问次数" desc limit 100
```
- **访问地区分布图表所关联的查询分析语句如下所示：**

```
select ip_to_province(client_ip) as province , count(*) as cnt where IP_TO_COUNTRY (client_ip) = '中国' group by province HAVING province not in ('','保留地址','*') order by cnt desc limit 100
```
- **有效访问用户TOP图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN ip_to_country(client_ip) = '上海' THEN concat(client_ip, ' ( shanghai )') WHEN ip_to_province(client_ip) = ' ' THEN concat(client_ip, ' ( Unknown IP )') WHEN ip_to_provider(client_ip) = '内网IP' THEN concat(client_ip, ' ( Private IP )') ELSE concat( client_ip, ' ( ', ip_to_country(client_ip), '/', ip_to_province(client_ip), '/', CASE WHEN ip_to_city(client_ip) = '-1' THEN 'Unknown city' ELSE ip_to_city(client_ip) END, ' ', ip_to_provider(client_ip), ' )' ) END AS client, pv as "总访问数", (pv - success_count) as "错误访问数", round( CASE WHEN "throughput" > 0 THEN "throughput" ELSE 0 END, 1 ) AS "下载总量(GB)" from ( select client_ip, count(*) as pv, sum(response_size) / 1024.0 / 1024 / 1024.0 AS throughput, sum( CASE WHEN http_code < 400 THEN 1 ELSE 0 END ) AS success_count from log group by client_ip order by success_count desc limit 100 )
```
- **下载量TOP用户图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN ip_to_country(client_ip)='上海' THEN concat(client_ip, ' ( shanghai )') WHEN ip_to_province(client_ip)=' ' THEN concat(client_ip, ' ( Unknown IP )') WHEN ip_to_provider(client_ip)='内网IP' THEN concat(client_ip, ' ( Private IP )') ELSE concat(client_ip, ' ( ', ip_to_country(client_ip), '/', ip_to_province(client_ip), '/', CASE WHEN ip_to_city(client_ip)='-1' THEN 'Unknown city' ELSE ip_to_city(client_ip) END, ' ', ip_to_provider(client_ip), ' )') END AS client, pv as "总访问数", error_count as "错误访问数", round( CASE WHEN "throughput" > 0 THEN "throughput" ELSE 0 END, 1 ) AS "下载总量(GB)" from ( select client_ip, count(*) as pv, sum(response_size)/1024.0/1024/1024.0 AS throughput, sum(CASE WHEN http_code > 400 THEN 1 ELSE 0 END) AS error_count from log group by client_ip order by throughput desc limit 100)
```

----结束

查看 CDN 热门资源

步骤1 在仪表盘模板下方，选择“CDN仪表盘模板 > CDN热门资源”，查看图表详情。

- **域名访问次数Top5图表所关联的查询分析语句如下所示：**

```
select domain ,count(*) as cnt group by domain order by cnt desc limit 5
```
- **域名下载流量Top5图表所关联的查询分析语句如下所示：**

```
select domain , sum(response_size) as "下载总量" group by domain order by "下载总量" desc limit 5
```
- **热门访问（URI）图表所关联的查询分析语句如下所示：**

```
select uri as URI, "访问次数", "访问人数", round( CASE WHEN "下载总量(GB)" > 0 THEN "下载总量(GB)" ELSE 0 END, 2 ) AS "下载总量(GB)" from ( select uri ,count(*) as "访问次数", APPROX_COUNT_DISTINCT(client_ip) as "访问人数", sum(response_size)/1024.0/1024.0/1024.0 as "下载总量(GB)" where http_code < 400 group by uri order by "访问次数" desc limit 100)
```
- **热门访问（来源）图表所关联的查询分析语句如下所示：**

```
select refer_domain as "来源",c as "次数",uv as "人数", round(c * 100.0 / sum(c), 2) as "百分比%" from (select refer_domain as refer_domain,count(*) as c,APPROX_COUNT_DISTINCT(client_ip) as uv from log where refer_domain != " " group by refer_domain order by c desc limit 100 ) GROUP BY refer_domain, c, uv
```
- **全国访问次数分布统计图表所关联的查询分析语句如下所示：**

```
select ip_to_province(client_ip) as province , count(*) as cnt where IP_TO_COUNTRY (client_ip) = '中国' group by province HAVING province not in ('','保留地址','*') order by cnt desc limit 1000
```
- **全国下载网速统计图表所关联的查询分析语句如下所示：**


```
select province, round( CASE WHEN "speed" > 0 THEN "speed" ELSE 0 END, 3 ) AS "speed" from
(select ip_to_province(client_ip) as province , sum(response_size)* 1.0 /(sum(response_time)+1) as
"speed" , count(*) as c where IP_TO_COUNTRY (client_ip) = '中国' group by province HAVING province
not in ('','保留地址','*') order by c desc limit 40)
```

- **省份统计图表**所关联的查询分析语句如下所示：

```
select ip_to_province(client_ip) as "省份" ,count(*) as "访问次数", sum(response_size)/
1024.0/1024.0/1024.0 as "下载流量(GB)" , sum(response_size) * 1.0 /sum(response_time) as "下载速度
(KB/s)" group by "省份" having ip_to_province(client_ip) != "" order by "下载流量(GB)" desc limit 200
```

- **运营商流量和速度图表**所关联的查询分析语句如下所示：

```
select ip_to_provider(client_ip) as isp , sum(response_size)* 1.0 /(sum(response_time)+1) as "下载速度
(KB/s)" , sum(response_size)/1024.0/1024.0/1024.0 as "下载总量(GB)" , count(*) as c group by isp
having ip_to_provider(client_ip) != "" order by c desc limit 10
```

- **运营商统计图表**所关联的查询分析语句如下所示：

```
select "运营商" , "访问次数" , round( CASE WHEN "下载流量(GB)" > 0 THEN "下载流量(GB)" ELSE 0
END, 2 ) AS "下载流量(GB)" , round( CASE WHEN "下载速度(KB/s)" > 0 THEN "下载速度(KB/s)"
ELSE 0 END, 2 ) AS "下载速度(KB/s)" from ( select ip_to_provider(client_ip) as "运营商" ,count(*)
as "访问次数" , sum(response_size)/1024.0/1024.0/1024.0 as "下载流量(GB)" , sum(response_size) *
1.0 /sum(response_time) as "下载速度(KB/s)" group by "运营商" having ip_to_provider(client_ip) != ""
and "运营商" not in ('*') order by "下载流量(GB)" desc limit 200)
```

----结束

7.3.4.4 CFW 仪表盘模板

云防火墙（Cloud Firewall，CFW）是新一代的云原生防火墙，提供云上互联网边界和VPC边界的防护，包括实时入侵检测与防御、全局统一访问控制、全流量分析可视化、日志审计与溯源分析等，同时支持按需弹性扩容、AI提升智能防御能力、灵活扩展满足云上业务的变化和扩张需求，极简应用让用户快速灵活应对威胁。云防火墙可以通过攻击事件日志查看检测到的危险流量的危险等级、受影响的端口、命中的规则、攻击事件类型等信息；通过访问控制日志查看根据访问控制策略放行或阻断的所有流量，以便更好的调整访问控制策略。

CFW仪表盘模板支持[查看CFW访问日志中心](#)、[查看CFW流量日志中心](#)和[查看CFW攻击日志中心](#)。

前提条件

- 已采集CFW日志，详情请参见[云防火墙CFW接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CFW 访问日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“日志管理”。

步骤3 在“日志应用”模块中，单击“CFW日志中心”，选择“进入仪表盘”。

步骤4 在仪表盘模板下方，选择“CFW仪表盘模板 > CFW访问日志中心”，查看图表详情。

- **互联网访问-拦截趋势图**展示互联网访问-拦截趋势的变化情况，所关联的查询分析语句如下所示：

```
select time_series(MILLIS_TO_TIMESTAMP(hit_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as
t_time,COUNT(*) as frequency WHERE action='deny' AND direction='out2in' group by t_time order by
t_time
```

- **主动外联-拦截趋势图**展示主动外联-拦截趋势的变化情况，所关联的查询分析语句如下所示：

```
select time_series(MILLIS_TO_TIMESTAMP(hit_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as  
t_time, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' group by t_time order by  
t_time
```

- **互联网阻断应用TOP5图**展示互联网阻断应用TOP5的变化情况，所关联的查询分析语句如下所示：
SELECT app, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' GROUP BY app
ORDER BY frequency DESC LIMIT 5
- **互联网阻断目的TOP5图**展示互联网阻断目的TOP5的变化情况，所关联的查询分析语句如下所示：
SELECT dst_ip, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' GROUP BY dst_ip
ORDER BY frequency DESC LIMIT 5
- **互联网阻断来源TOP5图**展示互联网阻断来源TOP5的变化情况，所关联的查询分析语句如下所示：
SELECT src_ip, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' GROUP BY src_ip
ORDER BY frequency DESC LIMIT 5
- **主动外联阻断应用TOP5图**展示主动外联阻断应用TOP5的变化情况，所关联的查询分析语句如下所示：
SELECT app, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' GROUP BY app
ORDER BY frequency DESC LIMIT 5
- **主动外联阻断目的TOP5图**展示主动外联阻断目的TOP5的变化情况，所关联的查询分析语句如下所示：
SELECT dst_ip, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' GROUP BY dst_ip
ORDER BY frequency DESC LIMIT 5
- **主动外联阻断来源TOP5图**展示主动外联阻断来源TOP5的变化情况，所关联的查询分析语句如下所示：
SELECT src_ip, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' GROUP BY src_ip
ORDER BY frequency DESC LIMIT 5

----结束

查看 CFW 流量日志中心

步骤1 在“日志应用”模块中，单击“CFW日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“CFW仪表盘模板 > CFW流量日志中心”，查看图表详情。

- **互联网访问流量趋势图**展示互联网访问流量趋势的变化情况，所关联的查询分析语句如下所示：
select time_series(MILLIS_TO_TIMESTAMP(start_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time,
SUM(to_s_bytes) AS '入流量', SUM(to_c_bytes) AS '出流量' WHERE direction='out2in' group by t_time
order by t_time
- **互联网访问流入地域分布(中国)图**展示互联网访问流入地域分布(中国)的变化情况，所关联的查询分析语句如下所示：
SELECT count(*) AS PV, ip_to_province(src_ip) AS province WHERE direction='out2in' and
IP_TO_COUNTRY (src_ip) = '中国' GROUP BY province HAVING province not in ('','保留地址','*') ORDER
BY PV DESC
- **互联网访问流入地域分布(世界)图**展示互联网访问流入地域分布(世界)的变化情况，所关联的查询分析语句如下所示：
SELECT count(*) AS PV, ip_to_country(src_ip) AS country WHERE direction='out2in' GROUP BY country
HAVING country not in ('','保留地址','*') ORDER BY PV DESC
- **互联网访问应用分布图**展示互联网访问应用分布的变化情况，所关联的查询分析语句如下所示：
SELECT app, COUNT(*) AS num WHERE direction='out2in' GROUP BY app ORDER BY num DESC
- **互联网访问源IP TOP5图**展示互联网访问源IP TOP5的变化情况，所关联的查询分析语句如下所示：
select src_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='out2in' GROUP BY src_ip ORDER BY
sum_bytes DESC LIMIT 5

- **互联网访问目的IP TOP5图**展示互联网访问目的IP TOP5的变化情况，所关联的查询分析语句如下所示：

```
select dst_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='out2in' GROUP BY dst_ip ORDER BY sum_bytes DESC LIMIT 5
```
- **主动外联流量趋势图**展示主动外联流量趋势的变化情况，所关联的查询分析语句如下所示：

```
select time_series(MILLIS_TO_TIMESTAMP(start_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, SUM(to_c_bytes) AS '入流量', SUM(to_s_bytes) AS '出流量' WHERE direction='in2out' group by t_time order by t_time
```
- **主动外联目的地域分布(中国)图**展示主动外联目的地域分布(中国)的变化情况，所关联的查询分析语句如下所示：

```
SELECT count(*) AS PV, ip_to_province(dst_ip) AS province WHERE direction='in2out' and IP_TO_COUNTRY (dst_ip) = '中国' GROUP BY province HAVING province not in ('','保留地址','*') ORDER BY PV DESC
```
- **目的地域分布(世界)图**展示目的地域分布(世界)的变化情况，所关联的查询分析语句如下所示：

```
SELECT count(*) AS PV, ip_to_country(dst_ip) AS country WHERE direction='in2out' GROUP BY country HAVING country not in ('','保留地址','*') ORDER BY PV DESC
```
- **主动外联-应用分布图**展示主动外联-应用分布的变化情况，所关联的查询分析语句如下所示：

```
SELECT app, COUNT(*) AS num WHERE direction='in2out' GROUP BY app ORDER BY num DESC
```
- **主动外联源IP TOP5图**展示主动外联源IP TOP5的变化情况，所关联的查询分析语句如下所示：

```
select src_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='in2out' GROUP BY src_ip ORDER BY sum_bytes DESC LIMIT 5
```
- **主动外联目的IP TOP5图**展示主动外联目的IP TOP5的变化情况，所关联的查询分析语句如下所示：

```
select dst_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='in2out' GROUP BY dst_ip ORDER BY sum_bytes DESC LIMIT 5
```

----结束

查看 CFW 攻击日志中心

步骤1 在“日志应用”模块中，单击“CFW日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“CFW仪表盘模板 > CFW攻击日志中心”，查看图表详情。

- **攻击趋势图**表所关联的查询分析语句如下所示：

```
select time_series(MILLIS_TO_TIMESTAMP(event_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, count(*) as frequency group by t_time order by t_time
```
- **攻击来源分布(中国)图**表所关联的查询分析语句如下所示：

```
SELECT count(*) as PV, ip_to_province(src_ip) as province WHERE IP_TO_COUNTRY (src_ip) = '中国' GROUP BY province HAVING province not in ('','保留地址','*')
```
- **攻击来源分布(世界)图**表所关联的查询分析语句如下所示：

```
SELECT count(*) AS PV, ip_to_country(src_ip) AS country GROUP BY country HAVING country not in ('','保留地址','*')
```
- **攻击类型分布图**表所关联的查询分析语句如下所示：

```
SELECT attack_type, COUNT(*) as num GROUP BY attack_type ORDER BY num
```
- **攻击目的TOP5图**表所关联的查询分析语句如下所示：

```
SELECT dst_ip, COUNT(*) as frequency GROUP BY dst_ip ORDER BY frequency DESC LIMIT 5
```
- **攻击来源TOP5图**表所关联的查询分析语句如下所示：

```
SELECT src_ip, COUNT(*) as frequency GROUP BY src_ip ORDER BY frequency DESC LIMIT 5
```

----结束

7.3.4.5 CSE 仪表盘模板

微服务引擎（Cloud Service Engine，CSE）是用于微服务应用的云中间件。用户可结合其他云服务，快速构建云原生微服务体系，实现微服务应用的快速开发和高可用运维。

CSE仪表盘模板支持[查看CSE层访问中心](#)、[查看CSE层监控中心](#)和[查看CSE层秒级监控](#)。

前提条件

- 已采集CSE日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CSE 层访问中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“CSE仪表盘模板 > CSE层访问中心”，查看图表详情。

- 过滤上游IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_host)
```
- 过滤调用链trace_id，所关联的查询分析语句如下所示：

```
select distinct(trace_id)
```
- PV对比昨日**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "pv" , 86400) as diff from (select count(1) as "pv" from log))
```
- PV对比上周**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "pv" , 604800) as diff from (select count(1) as "pv" from log))
```
- UV对比昨日**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "uv" , 86400) as diff from (select APPROX_COUNT_DISTINCT(authority) as "uv" from log))
```
- UV对比上周**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "uv" , 604800) as diff from (select APPROX_COUNT_DISTINCT(authority) as "uv" from log))
```
- 访问量PV分布(中国)**图表所关联的查询分析语句如下所示：

```
select ip_to_province(authority) as province, sum(ori_pv) as pv from (select authority, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) where IP_TO_COUNTRY (authority) = '中国' group by province HAVING province not in ('','保留地址','')
```
- 访问量PV分布(世界)**图表所关联的查询分析语句如下所示：

```
SELECT ip_to_country(authority) as country, sum(ori_pv) as PV from (select authority, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) GROUP BY country HAVING country not in ('','保留地址','')
```
- 平均时延分布（中国）**图表所关联的查询分析语句如下所示：

```
SELECT province, round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" FROM (SELECT ip_to_province(authority) as province, sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select authority, sum(duration) as rt, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) WHERE IP_TO_COUNTRY (authority) = '中国' GROUP BY province ) where province not in ('','保留地址','')
```
- 平均时延分布（世界）**图表所关联的查询分析语句如下所示：

```
SELECT country, round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)" FROM (SELECT ip_to_country(authority) as country, sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select authority, sum(duration) as rt, count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) GROUP BY country ) where country not in ('','保留地址','')
```

- **今日PV/UV**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss') as _time_PV,UV FROM (select  
TIME_CEIL(TIME_PARSE(start_time),'PT600S') AS _time_ , count(1) as PV,  
APPROX_COUNT_DISTINCT(authority) as UV from log WHERE _time_ <= CURRENT_TIMESTAMP and  
_time_ >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR  
group by _time_ order by _time_)
```
- **区域访问TOP10(省份)**图表所关联的查询分析语句如下所示：

```
select ip_to_province(authority) as "province", sum(ori_pv) as "访问次数" from (select authority,  
count(1) as ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) group by "province"  
HAVING "province" <> '-1' order by "访问次数" desc limit 10
```
- **区域访问TOP10(城市)**图表所关联的查询分析语句如下所示：

```
select ip_to_city(authority) as "city", sum(ori_pv) as "访问次数" from (select authority, count(1) as  
ori_pv group by authority ORDER BY ori_pv desc LIMIT 10000) group by "city" HAVING "city" <> '-1'  
order by "访问次数" desc limit 10
```
- **Host访问TOP10**图表所关联的查询分析语句如下所示：

```
select upstream_host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```
- **UserAgent访问TOP10**图表所关联的查询分析语句如下所示：

```
select user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc limit 10
```
- **设备占比(终端)**图表所关联的查询分析语句如下所示：

```
select case when regexp_like(lower(user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端'  
end as type , count(1) as total group by type
```
- **设备占比(系统)**图表所关联的查询分析语句如下所示：

```
select case when regexp_like(lower(user_agent), 'iphone|ipod|ios') then 'IOS' when  
regexp_like(lower(user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total  
group by type HAVING type != 'other'
```
- **TOP URL**图表所关联的查询分析语句如下所示：

```
select path , count(1) as pv, APPROX_COUNT_DISTINCT(authority) as UV, round(sum( case when  
response_code < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group by path  
ORDER by pv desc
```
- **TOP 访问IP**图表所关联的查询分析语句如下所示：

```
select authority as "来源IP",ip_to_country(authority) as "国家",ip_to_province(authority) as "省份",  
ip_to_city(authority) as "城市",ip_to_provider(authority) as "运营商",count(1) as "PV" group by  
authority ORDER by "PV" desc limit 100
```

----结束

查看 CSE 层监控中心

步骤1 在仪表盘模板下方，选择“CSE仪表盘模板 > CSE层监控中心”，查看图表详情。

- **过滤上游IP**，所关联的查询分析语句如下所示：

```
select distinct(upstream_host)
```
- **过滤调用链trace_id**，所关联的查询分析语句如下所示：

```
select distinct(trace_id)
```
- **访问量PV**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss') as _time_PV FROM ( SELECT TIME_CEIL  
( TIME_PARSE(start_time), 'PT300S' ) AS _time_ , count( 1 ) AS PV FROM log GROUP BY _time_ )
```
- **请求成功率**图表所关联的查询分析语句如下所示：

```
select ROUND(sum(case when response_code < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```
- **平均延迟**图表所关联的查询分析语句如下所示：

```
select round(avg(duration) * 1000, 3) as cnt
```
- **4XX请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "response_code" >= 400 and "response_code" < 500
```
- **404请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "response_code" = 404
```
- **429请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "response_code" = 429
```

- **504请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "response_code" = 504
```

- **5XX请求数**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_cnt FROM ( SELECT TIME_CEIL  
( TIME_PARSE(start_time), 'PT300S' ) AS _time_, count( 1 ) AS cnt FROM log where "response_code"  
>= 500 GROUP BY _time_ )
```

- **状态码分布**图表所关联的查询分析语句如下所示：

```
SELECT response_code, COUNT(1) AS rm GROUP BY response_code
```

- **访问量UV**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_UV FROM (select  
TIME_CEIL(TIME_PARSE(start_time),'PT600S') AS _time_ , APPROX_COUNT_DISTINCT(authority) as  
UV from log group by _time_)
```

- **流量**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT(_time_,'yyyy-MM-dd HH:mm:ss') AS _time_,round( CASE WHEN "入流量" > 0  
THEN "入流量" ELSE 0 END, 2 ) AS "入流量",round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0  
END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(TIME_PARSE(start_time),'PT600S') AS  
_time_,sum(bytes_received) / 1024.0 AS "入流量",sum(bytes_sent) / 1024.0 AS "出流量" group by  
_time_)
```

- **访问失败率**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,round( CASE WHEN "失败率" > 0  
THEN "失败率" ELSE 0 END, 2 ) AS "失败率",round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE  
0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(TIME_PARSE(start_time),'PT600S') AS  
_time_,sum(case when response_code >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率',  
sum(case when response_code >=500 THEN 1 ELSE 0 END)*100.0/COUNT(1) as '5XX比例' group by  
_time_)
```

- **延迟**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,round( CASE WHEN "平均" > 0  
THEN "平均" ELSE 0 END, 2 ) AS "平均",round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 )  
AS "P50",round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90",round( CASE WHEN  
"P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99",round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE  
0 END, 2 ) AS "P9999" from (select TIME_CEIL(TIME_PARSE(start_time),'PT600S') as  
_time_,avg(duration) * 1000 as "平均", APPROX_QUANTILE_DS("duration", 0.50)*1000 as "P50",  
APPROX_QUANTILE_DS("duration", 0.90)*1000 as "P90",APPROX_QUANTILE_DS("duration",  
0.99)*1000 as "P99",APPROX_QUANTILE_DS("duration", 0.9999)*1000 as "P9999" group by _time_)
```

- **Host请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT "upstream_host", pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)"  
ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)"  
ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0  
END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 )  
AS "出流量(KB)" FROM ( SELECT "upstream_host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( authority ) AS uv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 /  
count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", sum( bytes_received ) /  
1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "upstream_host" != "  
GROUP BY "upstream_host" ) ORDER BY pv DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT "upstream_host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0  
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0  
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0  
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "upstream_host", count( 1 ) AS pv, sum( CASE WHEN  
"response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration )  
* 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != "  
GROUP BY "upstream_host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT "upstream_host", pv,round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0  
END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0  
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0  
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "upstream_host", count( 1 ) AS pv, sum( CASE WHEN  
"response_code" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)",
```

```
avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " " GROUP BY "upstream_host") ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示:

```
SELECT path, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT path, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( authority ) AS uv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", sum( bytes_received ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "upstream_host" != " " GROUP BY path ) ORDER BY pv desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示:

```
SELECT path, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT path, count( 1 ) AS pv, sum( CASE WHEN "response_code" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " " GROUP BY path ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示:

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT authority as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( authority ) AS uv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", sum( bytes_received ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "upstream_host" != " " GROUP BY addr having length(authority) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示:

```
SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT authority as addr,count( 1 ) AS pv,sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( duration ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)",APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " " and "authority" != '-' GROUP BY addr ) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示:

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT authority as addr, count( 1 ) AS pv, sum( CASE WHEN "response_code" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " " and "authority" != '-' GROUP BY addr ) ORDER BY "访问失败率(%)" desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示:

```
SELECT path, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT path, count( 1 ) AS pv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != " " GROUP BY path ) ORDER BY "平均延迟(ms)" desc
```

----结束

查看 CSE 层秒级监控

步骤1 在仪表盘模板下方，选择“CSE仪表盘模板 > CSE层秒级监控”，查看图表详情。

- 过滤上游IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_host)
```
- 过滤调用链trace_id，所关联的查询分析语句如下所示：

```
select distinct(trace_id)
```
- **QPS**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time),'PT5S'),'yyyy-MM-dd HH:mm:ss') AS  
_time_, COUNT(*) as QPS from log group by _time_
```
- **成功率**图表所关联的查询分析语句如下所示：

```
select __time,round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select  
TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time),'PT5S'),'yyyy-MM-dd HH:mm:ss') as __time,  
sum(case when response_code < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group  
by __time)
```
- **延迟**图表所关联的查询分析语句如下所示：

```
select __time,round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟"  
",round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟"  
from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time),'PT5S'),'yyyy-MM-dd HH:mm:ss') as  
__time, avg(duration)* 1000 as '访问延迟',avg(upstream_service_time)* 1000 as 'Upstream延迟' from  
log group by __time)
```
- **流量**图表所关联的查询分析语句如下所示：

```
select __time,round( CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 3 ) AS "请求流量"  
",round( CASE WHEN "返回body流量" > 0 THEN "返回body流量" ELSE 0 END, 3 ) AS "返回body流量"  
from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time),'PT5S'),'yyyy-MM-dd HH:mm:ss') as  
__time , sum("bytes_received") / 1024.0 as "请求流量", sum("bytes_sent") / 1024.0 as "返回body流量"  
group by __time)
```
- **状态码**图表所关联的查询分析语句如下所示：

```
SELECT TIME_CEIL ( TIME_PARSE ( start_time ) , 'PT5S' ) AS "time", SUM( CASE WHEN  
"response_code" >= 200 AND "response_code" < 300 THEN 1 ELSE 0 END ) AS "2XX", SUM( CASE  
WHEN "response_code" >= 300 AND "response_code" < 400 THEN 1 ELSE 0 END ) AS "3XX",  
SUM( CASE WHEN "response_code" >= 400 AND "response_code" < 500 THEN 1 ELSE 0 END ) AS  
"4XX", SUM( CASE WHEN "response_code" >= 500 AND "response_code" < 600 THEN 1 ELSE 0 END )  
AS "5XX", SUM( CASE WHEN "response_code" < 200 OR "response_code" >= 600 THEN 1 ELSE 0  
END ) AS "其他" FROM log WHERE TIME_PARSE ( start_time ) IS NOT NULL GROUP BY "time"  
ORDER BY "time" ASC LIMIT 100000
```

----结束

7.3.4.6 CoreDNS 仪表盘模板

CoreDNS作为集群内的DNS服务器，您可以通过CoreDNS仪表盘模板查看CoreDNS日志来分析CoreDNS解析慢、访问高危请求域名等问题。

说明

请先开通管道符搜索方式才能使用CoreDNS仪表盘模板功能。如有需要请[提交工单](#)申请开通。

前提条件

- 已采集CoreDNS日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 CoreDNS 日志分析

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“CoreDNS仪表盘模板 > CoreDNS日志分析”，查看图表详情。

- 过滤域名，所关联的查询分析语句如下所示：

```
* | select name group by name limit 10000
```

- 过滤客户端IP，所关联的查询分析语句如下所示：

```
* | select remote group by remote limit 10000
```

- 过滤状态码，所关联的查询分析语句如下所示：

```
* | select rcode group by rcode limit 10000
```

- 总请求次数图表所关联的查询分析语句如下所示：

```
* | select diff[1] as nowCount, diff[1] - diff[2] as delta from (select compare(total, 86400) as diff from( select count(1) as total from log ) )
```

- NXDOMAIN次图表所关联的查询分析语句如下所示

```
(* and rcode: NXDOMAIN) | select diff[1] as nowCount, diff[1] - diff[2] as delta from (select compare(total, 86400) as diff from( select count(1) as total from log ) )
```

- 请求成功率图表所关联的查询分析语句如下所示：

```
* | select diff[1] as nowCount, round(diff[1] - diff[2], 2) as delta from (select compare(total, 86400) as diff from( select round( count_if(rcode = 'NOERROR' or rcode = 'NXDOMAIN') * 100.0 / (case when count(1) = 0 then 1 else count(1) end), 2) as total from log ) )
```

- 域名数图表所关联的查询分析语句如下所示：

```
* | select diff[1] as nowCount, diff[1] - diff[2] as delta from (select compare(total, 86400) as diff from( select approx_distinct(name) as total from log ) )
```

- 平均延迟图表所关联的查询分析语句如下所示：

```
* | select diff[1] as nowCount, round(diff[1] - diff[2], 3) as delta from (select compare(total, 86400) as diff from( select round(avg(duration * 1000), 3) as total from log ) )
```

- Timeout次数图表所关联的查询分析语句如下所示：

```
(level : ERROR) | select diff[1] as nowCount, round(diff[1] - diff[2], 3) as delta from (select compare(total, 86400) as diff from( select count(1) as "total" from log where errmsg like '%timeout%' ) )
```

- P95延迟图表所关联的查询分析语句如下所示：

```
* | select diff[1] as nowCount, round(diff[1] - diff[2], 3) as delta from (select compare(total, 86400) as diff from( select round(approx_percentile(duration * 1000, 0.95), 3) as total from log ) )
```

- P99延迟图表所关联的查询分析语句如下所示：

```
* | select diff[1] as nowCount, round(diff[1] - diff[2], 3) as delta from (select compare(total, 86400) as diff from( select round(approx_percentile(duration * 1000, 0.99), 3) as total from log ) )
```

- 请求QPS图表所关联的查询分析语句如下所示：

```
* | select date_format(t, '%Y-%m-%d %H:%i') as time , d[1] as "今天", d[2] as "昨天", d[3] as "上周" from( select t, ts_compare(pv, 86400, 604800 ) as d from(select from_unixtime(__time - __time % 60) as t, round(count(1)/ 60.0, 2) as pv from log group by t order by t limit 1440 ) group by t ) where date_format(t, '%Y-%m-%d %H:%i') is not null order by time
```

- 请求平均延迟(ms)图表所关联的查询分析语句如下所示：

```
* | select date_format(t, '%Y-%m-%d %H:%i') as time , d[1] as "今天", d[2] as "昨天", d[3] as "上周" from( select t, ts_compare(pv, 86400, 604800 ) as d from(select from_unixtime(__time - __time % 60) as t, round(avg(duration * 1000.0), 3) as pv from log group by t order by t limit 1440 ) group by t ) where date_format(t, '%Y-%m-%d %H:%i') is not null order by time
```

- 成功率(%)图表所关联的查询分析语句如下所示：

```
* | select date_format(t, '%Y-%m-%d %H:%i') as time , d[1] as "今天", d[2] as "昨天", d[3] as "上周" from( select t, ts_compare(pv, 86400, 604800 ) as d from(select from_unixtime(__time - __time % 60) as t, round( count_if(rcode = 'NOERROR') * 100.0 / (case when count(1) = 0 then 1 else count(1) end), 2) as pv from log group by t order by t limit 1440 ) group by t ) where date_format(t, '%Y-%m-%d %H:%i') is not null order by time
```

- P99延迟(ms)图表所关联的查询分析语句如下所示：

```
* | select date_format(t, '%Y-%m-%d %H:%i') as time , d[1] as "今天", d[2] as "昨天", d[3] as "上周" from( select t, ts_compare(pv, 86400, 604800 ) as d from(select from_unixtime(__time - __time % 60) as t, round(approx_percentile(duration * 1000.0, 0.99), 3) as pv from log group by t order by t limit 1440 ) group by t ) where date_format(t, '%Y-%m-%d %H:%i') is not null order by time
```

- Top请求域名图表所关联的查询分析语句如下所示：
`*| SELECT name, COUNT(*) as total group by name order by total desc`
- 状态码分布图表所关联的查询分析语句如下所示：
`*| SELECT rcode, COUNT(*) as total group by rcode order by total desc`
- TOP域名图表所关联的查询分析语句如下所示：
`*| SELECT name as "域名", rcode as "状态码", round(sum(size/ 1024.0), 3) as "请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB", COUNT(*) as "请求次数", round(avg(duration * 1000), 3) as "平均延迟(ms)" group by name, rcode order by "请求次数" desc`
- TOP NXDOMAIN域名图表所关联的查询分析语句如下所示：
`(* and rcode: NXDOMAIN)| SELECT name as "域名", rcode as "状态码", round(sum(size/ 1024.0), 3) as "请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB", COUNT(*) as "请求次数", round(avg(duration * 1000), 3) as "平均延迟(ms)" group by name, rcode order by "请求次数" desc`
- TOP 错误域名（非NXDOMAIN）图表所关联的查询分析语句如下所示：
`(* not rcode: NXDOMAIN not rcode : NOERROR)| SELECT name as "域名", rcode as "状态码", round(sum(size/ 1024.0), 3) as "请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB", COUNT(*) as "请求次数", round(avg(duration * 1000), 3) as "平均延迟(ms)" group by name, rcode order by "请求次数" desc`
- TOP请求Pod图表所关联的查询分析语句如下所示：
`*| SELECT remote as "IP", arbitrary(name) as "请求域名采样", round(sum(size/ 1024.0), 3) as "请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB", COUNT(*) as "请求次数", round(avg(duration * 1000), 3) as "平均延迟(ms)" group by remote order by "请求次数" desc`
- TOP NXDOMAIN Pod图表所关联的查询分析语句如下所示：
`(* and rcode : NXDOMAIN)| SELECT remote as "IP", arbitrary(name) as "请求域名采样", round(sum(size/ 1024.0), 3) as "请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB", COUNT(*) as "请求次数", round(avg(duration * 1000), 3) as "平均延迟(ms)" group by remote order by "请求次数" desc`
- TOP错误Pod（非NXDOMAIN）图表所关联的查询分析语句如下所示：
`(* not rcode : NXDOMAIN not rcode : NOERROR)| SELECT remote as "IP", arbitrary(name) as "请求域名采样", round(sum(size/ 1024.0), 3) as "请求流量KB", round(sum(rsize / 1024.0),3 ) as "返回流量KB", COUNT(*) as "请求次数", round(avg(duration * 1000), 3) as "平均延迟(ms)" group by remote order by "请求次数" desc`
- Timeout域名图表所关联的查询分析语句如下所示：
`(level : ERROR)| select name as "域名", count(1) as "次数" from log where errmsg like '%timeout%' group by name order by "次数" desc limit 100`
- 近100条请求APIServier的错误图表所关联的查询分析语句如下所示：
`(content : reflector.go)| select date_format(from_unixtime(__time), '%Y-%m-%d %H:%i:%S') as "时间", errmsg as "错误" from log order by __time desc limit 100`
- 解析日志图表所关联的查询分析语句如下所示：
`(* and level: INFO and duration > 0.010)| SELECT name as "域名", round( duration * 1000.0, 3) as "延迟(ms)", remote as "请求者", port as "端口", type as "类型", rcode as "结果", podName as "CoreDNS" from log order by duration desc limit 100`

----结束

7.3.4.7 DCS 仪表盘模板

分布式缓存服务（Distributed Cache Service，简称DCS）是一款内存数据库服务，兼容了Redis内存数据库引擎，满足用户高并发及数据快速访问的业务诉求。云日志服务提供对各种云资源操作记录的收集、存储和查询功能，可用于支撑安全分析、合规审计、资源跟踪和问题定位等常见应用场景。

云日志服务支持日志采集向导一站式采集DCS日志，DCS审计日志中心仪表盘支持展示访问用户数、访问客户端数、审计日志条数等图表。

前提条件

- 已采集DCS日志。

- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

DCS 审计日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“DCS仪表盘模板 > DCS审计日志中心”，查看图表详情。

- **访问用户数**图表所关联的查询分析语句如下所示：

```
select count(distinct(user)) as user_num
```

- **访问客户端数**图表所关联的查询分析语句如下所示：

```
select count(distinct(client_addr)) as client_num
```

- **审计日志条数**图表所关联的查询分析语句如下所示：

```
select count(1) as log_num
```

- **平均响应时间**图表所关联的查询分析语句如下所示：

```
select avg(use_time) as avg_time
```

- **平均QPS**图表所关联的查询分析语句如下所示：

```
select count(*) / CAST((TIMESTAMPDIFF (minute, MIN(__time),MAX(__time))+1) as FLOAT)
```

- **TOP5 用户**图表所关联的查询分析语句如下所示：

```
select user, count(1) as 'user_count' group by user order by count(1) desc LIMIT 5
```

- **TOP5 客户端**图表所关联的查询分析语句如下所示：

```
select client_addr, count(1) as 'remote_count' group by client_addr order by count(1) desc limit 5
```

- **TOP5 执行命令**图表所关联的查询分析语句如下所示：

```
select command_name, count(1) as 'command_count' group by command_name order by count(1) desc
```

- **热Key**图表所关联的查询分析语句如下所示：

```
SELECT count(*) as 'count', command_keys GROUP by command_keys order by count(*) desc limit 5
```

- **审计日志详情**图表所关联的查询分析语句如下所示：

```
select "client_addr" as '客户端IP',"client_type" as '客户端类型',"server_addr" as '服务端IP',"command_name" as '执行命令',"command_keys" as '命令KEYS',"command_param" as '命令内容',"command_type" as '命令类型',"use_time" as '执行耗时',"time" as '执行时间',"db" as 'DB',"user" as '账号名',"instance_id" as '实例ID',"role" as '节点角色',"extend" as '扩展信息'
```

----结束

7.3.4.8 DDS 仪表盘模板

文档数据库服务（Document Database Service）完全兼容MongoDB协议，提供安全、高可用、高可靠、弹性伸缩和易用的数据库服务，同时提供一键部署、弹性扩容、容灾、备份、恢复、监控和告警等功能。云日志服务进行分析日志、搜索日志、日志可视化、下载日志和查看实时日志等操作。

云日志服务支持日志采集向导一站式采集DDS日志，DDS审计日志中心展示审计日志条数、访问用户数、访问客户端数等图表。

前提条件

- 已采集DDS日志，详情请参见[文档数据库服务DDS接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

DDS 审计日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“DDS仪表盘模板 > DDS审计日志中心”，查看图表详情。

- 过滤操作类型，所关联的查询分析语句如下所示：

```
select distinct(optype)
```
- 过滤客户端IP，所关联的查询分析语句如下所示：

```
select distinct(user_ip)
```
- 过滤数据库名，所关联的查询分析语句如下所示：

```
select distinct(db)
```
- 过滤用户名，所关联的查询分析语句如下所示：

```
select distinct(user)
```
- 过滤集合名，所关联的查询分析语句如下所示：

```
select distinct(coll)
```
- **审计日志条数**图表所关联的查询分析语句如下所示：

```
select count(1) as log_num
```
- **访问用户数**图表所关联的查询分析语句如下所示：

```
select count(distinct(user)) as user_num
```
- **访问客户端数**图表所关联的查询分析语句如下所示：

```
select count(distinct(user_ip)) as client_num
```
- **TOP5 执行命令**图表所关联的查询分析语句如下所示：

```
select optype, count(1) as 'command_count' group by optype order by count(1) desc LIMIT 5
```
- **TOP5 用户**图表所关联的查询分析语句如下所示：

```
select user, count(1) as 'user_count' group by user order by count(1) desc LIMIT 5
```
- **TOP5 客户端**图表所关联的查询分析语句如下所示：

```
select user_ip, count(1) as 'remote_count' group by user_ip order by count(1) desc LIMIT 5
```
- **审计日志详情**图表所关联的查询分析语句如下所示：

```
select "time" as "执行时间","user" as "用户名","param" as "查询语句","instanceid" as "实例ID","db" as "DB","coll" as "集合名","user_ip" as "客户端IP"
```

----结束

7.3.4.9 DMS 仪表盘模板

分布式消息服务Kafka版（Distributed Message Service for Kafka）是一款基于开源社区版Kafka提供的消息队列服务，向用户提供计算、存储和带宽资源独占式的Kafka专享实例。重平衡日志记录Rebalance的详情，包括Rebalance时间、原因和触发Rebalance的客户端等。

云日志服务支持日志采集向导一站式采集DMS重平衡日志，支持多维度分析，并为DMS-Rebalance日志配置结构化和仪表盘。该仪表盘主要展示DMS重平衡日志的重平衡消费组个数、重平衡次数、消费组重平衡次数、重平衡原因及组详情。

前提条件

- 已采集DMS日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

DMS 重平衡日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“DMS仪表盘模板 > DMS重平衡日志中心”，查看图表详情。

- 消费组ID，所关联的查询分析语句如下所示：

```
select distinct("message.groupId")
```
- 过滤重平衡原因。
- **重平衡消费组个数**图表所关联的查询分析语句如下所示：

```
select count(distinct("message.groupId")) as 'total' from log where ("message.type"!='RESPONSE' and "message.type" !='REQUEST')
```
- **重平衡次数**图表所关联的查询分析语句如下所示：

```
select count(*) as 'total' where ("message.type" !='RESPONSE' and "message.type" !='REQUEST')
```
- **消费组重平衡次数**图表所关联的查询分析语句如下所示：

```
select "message.groupId" as 'GroupId', count(*) as 'Count' where ("message.type" !='RESPONSE' and 'message.type' !='REQUEST') group by "message.groupId"
```
- **重平衡原因及组详情**图表所关联的查询分析语句如下所示：

```
select __time as 'Time', "message.type" as 'Type', "message.groupId" as 'GroupId', "message.reason" as 'Reason', "message.group" as 'Group' where ("message.type" !='RESPONSE' and "message.type" !='REQUEST')
```

----结束

7.3.4.10 DSL 仪表盘模板

DSL加工是LTS为您提供的一站式日志加工平台，基于领域自定义的脚本语言和200多个内置函数，您可以在LTS控制台实现端到端的日志规整、富化、流转、脱敏、过滤等加工任务。

云日志服务支持DSL（Domain Specific Language）仪表盘模板，DSL加工任务监控中心主要展示加工任务ID、加工任务名称、输入行数、输出行数等信息。

前提条件

- 已创建DSL加工任务。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

DSL 加工任务监控中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“DSL仪表盘模板 > DSL加工任务监控中心”，查看图表详情。

- 过滤加工任务ID，所关联的查询分析语句如下所示：

```
select distinct(task_id)
```
- 过滤加工任务名称，所关联的查询分析语句如下所示：

```
select distinct(task_name)
```
- **输入行数**图表所关联的查询分析语句如下所示：

```
SELECT CASE WHEN "input" < 1000 THEN concat( cast( "input" AS VARCHAR ), '行' ) WHEN "input" < 1000 * 1000 THEN concat( cast( round( "input"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN
```

```
"input" < 1000000000 THEN concat( cast( round( "input"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )  
WHEN "input"/ 1000.0 < 1000000000 THEN concat( cast( round( "input"/ 1000 / 1000000.0, 1 ) AS  
VARCHAR ), '十亿行' ) ELSE concat( cast( round( "input"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS  
VARCHAR ), '万亿行' ) END AS "total" from (select sum("process.accept") as "input")
```

- **输出行数图表所关联的查询分析语句如下所示**

```
SELECT CASE WHEN "delivered" < 1000 THEN concat( cast( "delivered" AS VARCHAR ), '行' )  
WHEN "delivered" < 1000 * 1000 THEN concat( cast( round( "delivered"/ 1000, 1 ) AS VARCHAR ), '千  
行' ) WHEN "delivered" < 1000000000 THEN concat( cast( round( "delivered"/ 1000000.0, 1 ) AS  
VARCHAR ), '百万行' ) WHEN "delivered"/ 1000.0 < 1000000000 THEN  
concat( cast( round( "delivered"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE  
concat( cast( round( "delivered"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS  
"total" from (select sum("process.delivered") as "delivered")
```

- **过滤行数图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN "drop" < 1000 THEN concat( cast( "drop" AS VARCHAR ), '行' ) WHEN "drop"  
< 1000 * 1000 THEN concat( cast( round( "drop"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "drop" <  
1000000000 THEN concat( cast( round( "drop"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN  
"drop"/ 1000.0 < 1000000000 THEN concat( cast( round( "drop"/ 1000 / 1000000.0, 1 ) AS  
VARCHAR ), '十亿行' ) ELSE concat( cast( round( "drop"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS  
VARCHAR ), '万亿行' ) END AS "total" from (select sum("process.drop") as "drop")
```

- **失败行数图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN "failed" < 1000 THEN concat( cast( "failed" AS VARCHAR ), '行' ) WHEN  
"failed" < 1000 * 1000 THEN concat( cast( round( "failed"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN  
"failed" < 1000000000 THEN concat( cast( round( "failed"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )  
WHEN "failed"/ 1000.0 < 1000000000 THEN concat( cast( round( "failed"/ 1000 / 1000000.0, 1 ) AS  
VARCHAR ), '十亿行' ) ELSE concat( cast( round( "failed"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS  
VARCHAR ), '万亿行' ) END AS "total" from (select sum("process.failed") as "failed")
```

- **执行记录图表所关联的查询分析语句如下所示：**

```
select TIME_FORMAT( MILLIS_TO_TIMESTAMP("start"), 'yyyy-MM-dd HH:mm:ss:SSS', '+08:00') as "统  
计开始时间", TIME_FORMAT( MILLIS_TO_TIMESTAMP("end"), 'yyyy-MM-dd HH:mm:ss:SSS', '+08:00')  
as "统计结束时间", "process.accept" as "输入行数", "process.delivered" as "输出行数", "process.drop" as  
"过滤行数", "process.failed" as "失败行数" limit 1000
```

----结束

7.3.4.11 ELB 仪表盘模板（共享型）

弹性负载均衡（Elastic Load Balance，简称ELB）是将访问流量根据分配策略分发到后端多台服务器的流量分发控制服务。弹性负载均衡可以通过流量分发扩展应用系统对外的服务能力，同时通过消除单点故障提升应用系统的可用性，支持查看和分析对七层负载均衡HTTP和HTTPS进行请求的详细访问日志记录，包括请求时间、客户端IP地址、请求路径和服务器响应等。

ELB仪表盘模板支持[查看ELB7层访问中心](#)、[查看ELB7层监控中心](#)和[查看ELB7层秒级监控](#)。

- upstream_addr_priv（该参数适用于共享型负载均衡）：后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。
- upstream_addr（该参数适用于独享型负载均衡）：后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。

更多信息请参考[独享型负载均衡与共享型弹性负载均衡的区别](#)。

前提条件

- 已采集ELB日志，详情请参见[弹性负载均衡ELB接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 ELB7 层访问中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“日志管理”。

步骤3 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤4 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层访问中心”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **PV对比昨日**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "pv", 86400) as diff from (select count(1) as "pv" from log))
```

- **PV对比上周**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "pv", 604800) as diff from (select count(1) as "pv" from log))
```

- **UV对比昨日**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "uv", 86400) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log))
```

- **UV对比上周**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "uv", 604800) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log))
```

- **访问量PV分布(中国)**图表所关联的查询分析语句如下所示：

```
select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','*')
```

- **访问量PV分布(世界)**图表所关联的查询分析语句如下所示：

```
SELECT ip_to_country(remote_addr) as country,COUNT(1) as PV GROUP BY country HAVING country not in ('','保留地址','*')
```

- **访问量UV分布(中国)**图表所关联的查询分析语句如下所示：

```
select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as uv where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','*')
```

- **访问量UV分布(世界)**图表所关联的查询分析语句如下所示：

```
select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv group by country HAVING country not in ('','保留地址','*')
```

- **平均时延分布（中国）** 图表所关联的查询分析语句如下所示：

```
SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(remote_addr) as province,avg(request_time) * 1000 AS "平均延迟(ms)"WHERE IP_TO_COUNTRY (remote_addr) = '中国'GROUP BY province HAVING province not in ('','保留地址','*'))
```
- **平均时延分布（世界）** 图表所关联的查询分析语句如下所示：

```
SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(remote_addr) as country,avg(request_time) * 1000 AS "平均延迟(ms)" GROUP BY country HAVING country not in ('','保留地址','*'))
```
- **今日PV/UV** 图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'),PT600S) AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ') <= CURRENT_TIMESTAMP and TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ') >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR group by _time_ order by _time_)
```
- **7日PV/UV** 图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'),PT600S) AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ') <= CURRENT_TIMESTAMP and TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ') >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR - INTERVAL '7' DAY group by _time_ order by _time_ )
```
- **区域访问TOP10(省份)** 图表所关联的查询分析语句如下所示：

```
select ip_to_province(remote_addr) as "province", count(1) as "访问次数" group by "province" HAVING "province" <> '-1' order by "访问次数" desc limit 10
```
- **区域访问TOP10(城市)** 图表所关联的查询分析语句如下所示：

```
select ip_to_city(remote_addr) as "city", count(1) as "访问次数" group by "city" HAVING "city" <> '-1' order by "访问次数" desc limit 10
```
- **Host访问TOP10** 图表所关联的查询分析语句如下所示：

```
select host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```
- **UserAgent访问TOP10** 图表所关联的查询分析语句如下所示：

```
select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc limit 10
```
- **设备占比(终端)** 图表所关联的查询分析语句如下所示：

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type , count(1) as total group by type
```
- **设备占比(系统)** 图表所关联的查询分析语句如下所示：

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total group by type HAVING type != 'other'
```
- **TOP URL** 图表所关联的查询分析语句如下所示：

```
select router_request_uri , count(1) as pv, APPROX_COUNT_DISTINCT(remote_addr) as uv, round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group by router_request_uri ORDER by pv desc
```
- **TOP 访问IP** 图表所关联的查询分析语句如下所示：

```
select remote_addr as "来源IP",ip_to_country(remote_addr) as "国家",ip_to_province(remote_addr) as "省份",ip_to_city(remote_addr) as "城市",ip_to_provider(remote_addr) as "运营商",count(1) as "PV",http_user_agent as "UserAgent采样",router_request_uri as "URL采样" group by remote_addr,http_user_agent,router_request_uri ORDER by "PV" desc
```

----结束

查看 ELB7 层监控中心

步骤1 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层监控中心”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **访问量PV**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , PV FROM ( select TIME_CEIL( TIME_PARSE( SUBSTRING( time_iso8601 , 2 , 25 ) , 'yyyy-MM-dd"T"HH:mm:ssZZ' ) , 'PT600S' ) AS _time_ , count(1) as PV from log group by _time_ order by _time_ )
```

- **请求成功率**图表所关联的查询分析语句如下所示：

```
select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```

- **4XX请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
```

- **499请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" = 499
```

- **平均延迟**图表所关联的查询分析语句如下所示：

```
select round(avg(request_time) * 1000, 3) as cnt
```

- **404请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" = 404
```

- **504请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" = 504
```

- **5XX请求数**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , cnt FROM ( SELECT TIME_CEIL ( TIME_PARSE( SUBSTRING( time_iso8601 , 2 , 25 ) , 'yyyy-MM-dd"T"HH:mm:ssZZ' ) , 'PT300S' ) AS _time_ , count( 1 ) AS cnt FROM log where "status" >= 500 GROUP BY _time_ )
```

- **状态码分布**图表所关联的查询分析语句如下所示：

```
SELECT status, COUNT(1) AS rm GROUP BY status
```

- **访问量UV**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , UV FROM ( select TIME_CEIL( TIME_PARSE( SUBSTRING( time_iso8601 , 2 , 25 ) , 'yyyy-MM-dd"T"HH:mm:ssZZ' ) , 'PT600S' ) AS _time_ , APPROX_COUNT_DISTINCT( remote_addr ) as UV from log group by _time_ )
```

- **流量**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) AS _time_ , round( CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量" , round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2 ) AS "出流量" FROM ( SELECT TIME_CEIL( TIME_PARSE( SUBSTRING( time_iso8601 , 2 , 25 ) , 'yyyy-MM-dd"T"HH:mm:ssZZ' ) , 'PT600S' ) AS _time_ , sum( request_length ) / 1024.0 AS "入流量" , sum( bytes_sent ) / 1024.0 AS "出流量" group by _time_ )
```

- **访问失败率**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_, round( CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率", round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS _time_, sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率', sum(case when status >= 500 THEN 1 ELSE 0 END)*100.0/COUNT(1) as '5XX比例' group by _time_)
```

- **延迟**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_, round( CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均", round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 ) AS "P50", round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90", round( CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99", round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') as _time_, avg(request_time) * 1000 as "平均", APPROX_QUANTILE_DS("request_time", 0.50)*1000 as "P50", APPROX_QUANTILE_DS("request_time", 0.90)*1000 as "P90", APPROX_QUANTILE_DS("request_time", 0.99)*1000 as "P99", APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as "P9999" group by _time_)
```

- **Host请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT host, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != "" GROUP BY "host" ) ORDER BY pv DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != "" GROUP BY "host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT "host", pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != "" GROUP BY "host" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT router_request_uri, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT router_request_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != "" GROUP BY router_request_uri ) ORDER BY pv desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT router_request_uri, pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT router_request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
```

```
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
router_request_uri ) ORDER BY "平均延迟(ms)" desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT router_request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0  
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0  
END, 3 ) AS "P99延迟(ms)" FROM( SELECT router_request_uri, count( 1 ) AS pv, sum( CASE WHEN  
"status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) *  
1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
router_request_uri) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )  
AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入  
流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量  
(KB)" FROM ( SELECT upstream_addr_priv as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 /  
count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)"  
WHERE "host" != " " GROUP BY addr having length(upstream_addr_priv) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS  
"访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS  
"平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90  
延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟  
(ms)" FROM (SELECT upstream_addr_priv as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400  
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延  
迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟  
(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != "  
GROUP BY addr having length(upstream_addr_priv) > 2) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS  
"访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS  
"平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90  
延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟  
(ms)" FROM ( SELECT upstream_addr_priv as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >=  
400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平  
均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
addr having length(upstream_addr_priv) > 2) ORDER BY "访问失败率(%)" desc
```

----结束

查看 ELB7 层秒级监控

步骤1 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层秒级监控”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间

的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。
- **QPS图**展示QPS的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT1S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') AS __time_, COUNT(*) as QPS from log group by __time_
```
- **成功率图**展示成功率的变化情况，所关联的查询分析语句如下所示：

```
select __time, round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end, 2) as "成功率" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') as __time, sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group by __time)
```
- **延迟图**展示延迟的变化情况，所关联的查询分析语句如下所示：

```
select __time, round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end, 2) as "访问延迟", round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end, 2) as "Upstream延迟" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') as __time, avg(request_time) * 1000 as '访问延迟', avg(upstream_response_time) * 1000 as 'Upstream延迟' from log group by __time)
```
- **流量图**展示流量的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(__time, 'yyyy-MM-dd HH:mm:ss', '+08:00') AS __time, round(CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 2) AS "请求流量", round(CASE WHEN "返回body流量" > 0 THEN "返回body流量" ELSE 0 END, 2) AS "返回body流量" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as __time, sum("request_length") / 1024.0 as "请求流量", sum("body_bytes_sent") / 1024.0 as "返回body流量" group by __time )
```
- **状态码图**展示状态码的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(t.t, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "time", CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX", CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX", CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX", CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX", CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他" from ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) AS VARCHAR) as "2XX" from log group by t order by t asc ) t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) AS VARCHAR) as "3XX" from log group by t order by t asc ) a on t.t = a.t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) AS VARCHAR) as "4XX" from log group by t order by t asc ) b on t.t = b.t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) AS VARCHAR) as "5XX" from log group by t order by t asc ) c on t.t = c.t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) AS VARCHAR) as "其他" from log group by t order by t asc ) d on t.t = d.t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) AS VARCHAR) as "其他" from log group by t order by t asc ) e on t.t = e.t
```

- 后端响应码图展示后端响应码的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(t.t, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "time", CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX", CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX", CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX", CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX", CASE WHEN e."其他" IS NOT NULL THEN e."其他" ELSE 0 END as "其他" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"HH:mm:ssZZ'), 'PT5S') as t from log group by t order by t asc) t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "upstream_status" >= 200 and "upstream_status" < 300 group by t order by t asc) a on t.t = a.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "upstream_status" >= 300 and "upstream_status" < 400 group by t order by t asc) b on t.t = b.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "upstream_status" >= 400 and "upstream_status" < 500 group by t order by t asc) c on t.t = c.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "upstream_status" >= 500 and "upstream_status" < 600 group by t order by t asc) d on t.t = d.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "upstream_status" < 200 or "upstream_status" >= 600 group by t order by t asc) e on t.t = e.t
```

----结束

7.3.4.12 ELB 仪表盘模板（独享型）

弹性负载均衡（Elastic Load Balance，简称ELB）是将访问流量根据分配策略分发到后端多台服务器的流量分发控制服务。弹性负载均衡可以通过流量分发扩展应用系统对外的服务能力，同时通过消除单点故障提升应用系统的可用性，支持查看和分析对七层负载均衡HTTP和HTTPS进行请求的详细访问日志记录，包括请求时间、客户端IP地址、请求路径和服务器响应等。

ELB仪表盘模板支持[查看ELB7层访问中心](#)、[查看ELB7层监控中心](#)和[查看ELB7层秒级监控](#)。

- upstream_addr_priv（该参数适用于共享型负载均衡）：后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。
- upstream_addr（该参数适用于独享型负载均衡）：后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。

更多信息请参考[独享型负载均衡与共享型弹性负载均衡的区别](#)。

前提条件

- 已采集ELB日志，详情请参见[弹性负载均衡ELB接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 ELB7 层访问中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“日志管理”。

步骤3 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤4 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层访问中心”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：
select distinct(lb_name)

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **PV对比昨日**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "pv" , 86400) as diff from (select count(1) as "pv" from log))
```

- **PV对比上周**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "pv" , 604800) as diff from (select count(1) as "pv" from log))
```

- **UV对比昨日**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "uv" , 86400) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log))
```

- **UV对比上周**图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "uv" , 604800) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log))
```

- **访问量PV分布(中国)**图表所关联的查询分析语句如下所示：

```
select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','*')
```

- **访问量PV分布(世界)**图表所关联的查询分析语句如下所示：

```
SELECT ip_to_country(remote_addr) as country,COUNT(1) as PV GROUP BY country HAVING country not in ('','保留地址','*')
```

- **访问量UV分布(中国)**图表所关联的查询分析语句如下所示：

```
select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as uv where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','*')
```

- **访问量UV分布(世界)**图表所关联的查询分析语句如下所示：

```
select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv group by country HAVING country not in ('','保留地址','*')
```

- **平均时延分布（中国）**图表所关联的查询分析语句如下所示：

```
SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(remote_addr) as province,avg(request_time) * 1000 AS "平均延迟(ms)"WHERE IP_TO_COUNTRY (remote_addr) = '中国'GROUP BY province HAVING province not in ('','保留地址','*'))
```

- **平均时延分布（世界）**图表所关联的查询分析语句如下所示：

```
SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(remote_addr) as country,avg(request_time) * 1000 AS "平均延迟(ms)" GROUP BY country HAVING country not in ('','保留地址','*'))
```

- **今日PV/UV**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"THH:mm:ssZZ'), 'PT600S') AS
```

```
_time_ , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE  
TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ') <=  
CURRENT_TIMESTAMP and TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-  
dd"T"HH:mm:ssZZ') >= DATE_TRUNC( 'DAY', (CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) -  
INTERVAL '8' HOUR group by _time_ order by _time_)
```

- **7日PV/UV图表所关联的查询分析语句如下所示：**

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss' , '+08:00' ) as _time_ , PV, UV FROM (select  
TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ') , 'PT600S') AS  
_time_ , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE  
TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ') <=  
CURRENT_TIMESTAMP and TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-  
dd"T"HH:mm:ssZZ') >= DATE_TRUNC( 'DAY', (CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) -  
INTERVAL '8' HOUR - INTERVAL '7' DAY group by _time_ order by _time_ )
```

- **区域访问TOP10(省份)图表所关联的查询分析语句如下所示：**

```
select ip_to_province(remote_addr) as "province", count(1) as "访问次数" group by "province"  
HAVING "province" <> '-1' order by "访问次数" desc limit 10
```

- **区域访问TOP10(城市)图表所关联的查询分析语句如下所示：**

```
select ip_to_city(remote_addr) as "city", count(1) as "访问次数" group by "city" HAVING "city" <> '-1'  
order by "访问次数" desc limit 10
```

- **Host访问TOP10图表所关联的查询分析语句如下所示：**

```
select host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```

- **UserAgent访问TOP10图表所关联的查询分析语句如下所示：**

```
select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc  
limit 10
```

- **设备占比(终端)图表所关联的查询分析语句如下所示：**

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC  
端' end as type , count(1) as total group by type
```

- **设备占比(系统)图表所关联的查询分析语句如下所示：**

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when  
regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as  
total group by type HAVING type != 'other'
```

- **TOP URL图表所关联的查询分析语句如下所示：**

```
select router_request_uri , count(1) as pv, APPROX_COUNT_DISTINCT(remote_addr) as uv,  
round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group  
by router_request_uri ORDER by pv desc
```

- **TOP 访问IP图表所关联的查询分析语句如下所示：**

```
select remote_addr as "来源IP", ip_to_country(remote_addr) as "国家", ip_to_province(remote_addr) as  
"省份", ip_to_city(remote_addr) as "城市", ip_to_provider(remote_addr) as "运营商", count(1) as  
"PV", http_user_agent as "UserAgent采样", router_request_uri as "URL采样" group by  
remote_addr, http_user_agent, router_request_uri ORDER by "PV" desc
```

----结束

查看 ELB7 层监控中心

步骤1 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层监控中心”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间

的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **访问量PV**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , PV FROM (select  
TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS  
_time_ , count(1) as PV from log group by _time_ order by _time_ )
```

- **请求成功率**图表所关联的查询分析语句如下所示：

```
select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```

- **4XX请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
```

- **499请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" = 499
```

- **平均延迟**图表所关联的查询分析语句如下所示：

```
select round(avg(request_time) * 1000, 3) as cnt
```

- **404请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" = 404
```

- **504请求数**图表所关联的查询分析语句如下所示：

```
SELECT COUNT(1) as cnt WHERE "status" = 504
```

- **5XX请求数**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , cnt FROM ( SELECT  
TIME_CEIL ( TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT300S' )  
AS _time_ , count( 1 ) AS cnt FROM log where "status" >= 500 GROUP BY _time_ )
```

- **状态码分布**图表所关联的查询分析语句如下所示：

```
SELECT status, COUNT(1) AS rm GROUP BY status
```

- **访问量UV**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , UV FROM (select  
TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS  
_time_ , APPROX_COUNT_DISTINCT(remote_addr) as UV from log group by _time_ )
```

- **流量**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT(_time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00') AS _time_ , round( CASE WHEN "入流  
量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量" , round( CASE WHEN "出流量" > 0 THEN "出流量"  
ELSE 0 END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2,  
25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS _time_ , sum(request_length) / 1024.0 AS "入流量"  
 , sum(bytes_sent) / 1024.0 AS "出流量" group by _time_ )
```

- **访问失败率**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , round( CASE WHEN "失  
败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率" , round( CASE WHEN "5XX比例" > 0 THEN "5XX  
比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601,  
2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') AS _time_ , sum(case when status >= 400 then 1 else 0  
end) * 100.0 / count(1) as '失败率' , sum(case when status >= 500 THEN 1 ELSE 0 END)*100.0/  
COUNT(1) as '5XX比例' group by _time_ )
```

- **延迟**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_ , round( CASE WHEN "平  
均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均" , round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0  
END, 2 ) AS "P50" , round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90" , round( CASE  
WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99" , round( CASE WHEN "P9999" > 0 THEN  
"P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601,  
2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT600S') as _time_ , avg(request_time) * 1000 as "平均",
```



```
APPROX_QUANTILE_DS("request_time", 0.50)*1000 as "P50", APPROX_QUANTILE_DS("request_time",  
0.90)*1000 as "P90",APPROX_QUANTILE_DS("request_time", 0.99)*1000 as  
'P99',APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as 'P9999' group by _time_)
```

- **Host请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT host, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )  
AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入  
流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量  
(KB)" FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( http_x_forwarded_for )  
AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率  
(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",  
sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY "host" ) ORDER BY  
pv DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )  
AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0  
END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟  
(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT "host", pv,round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 )  
AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE  
0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"host" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT router_request_uri, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)"  
ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)"  
ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0  
END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 )  
AS "出流量(KB)" FROM ( SELECT router_request_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 /  
count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)"  
WHERE "host" != " " GROUP BY router_request_uri ) ORDER BY pv desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT router_request_uri, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE  
0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0  
END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END,  
3 ) AS "P99延迟(ms)" FROM ( SELECT router_request_uri, count( 1 ) AS pv, sum( CASE WHEN "status"  
< 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS  
"平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
router_request_uri ) ORDER BY "平均延迟(ms)" desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT router_request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE  
0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0  
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0  
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT router_request_uri, count( 1 ) AS pv, sum( CASE WHEN  
"status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) *  
1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
router_request_uri) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY addr having length(upstream_addr) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP图表所关联的查询分析语句如下所示：**

```
SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT upstream_addr as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY addr having length(upstream_addr) > 2) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP图表所关联的查询分析语句如下所示：**

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY addr having length(upstream_addr) > 2) ORDER BY "访问失败率(%)" desc
```

----结束

查看 ELB7 层秒级监控

步骤1 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤2 在仪表盘模板下方，选择“ELB仪表盘模板 > ELB7层秒级监控”，查看图表详情。

- 过滤所有负载均衡器，所关联的查询分析语句如下所示：

```
select distinct(lb_name)
```

- 过滤所有客户端IP，所关联的查询分析语句如下所示：

```
select distinct(remote_addr)
```

- 过滤所有后端服务器IP，所关联的查询分析语句如下所示：

```
select distinct(upstream_addr)
```

- 过滤所有弹性IP地址，所关联的查询分析语句如下所示：

```
select distinct(eip_address)
```

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **QPS图展示QPS的变化情况，所关联的查询分析语句如下所示：**

```
select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT1S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') AS _time_, COUNT(*) as QPS from log group by _time_
```

- **成功率图**展示成功率的变化情况，所关联的查询分析语句如下所示：

```
select _time, round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end, 2) as "成功率" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time, sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group by _time)
```

- **延迟图**展示延迟的变化情况，所关联的查询分析语句如下所示：

```
select _time, round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end, 2) as "访问延迟", round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end, 2) as "Upstream延迟" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time, avg(request_time) * 1000 as '访问延迟', avg(upstream_response_time) * 1000 as 'Upstream延迟' from log group by _time)
```

- **流量图**展示流量的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(_time, 'yyyy-MM-dd HH:mm:ss', '+08:00') AS _time, round(CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 2) AS "请求流量", round(CASE WHEN "返回body流量" > 0 THEN "返回body流量" ELSE 0 END, 2) AS "返回body流量" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as _time, sum("request_length") / 1024.0 as "请求流量", sum("body_bytes_sent") / 1024.0 as "返回body流量" group by _time)
```

- **状态码图**展示状态码的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(t.t, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "time", CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX", CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX", CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX", CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX", CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t from log group by t order by t asc) t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "status" >= 200 and "status" < 300 group by t order by t asc) a on t.t = a.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "status" >= 300 and "status" < 400 group by t order by t asc) b on t.t = b.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "status" >= 400 and "status" < 500 group by t order by t asc) c on t.t = c.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "status" >= 500 and "status" < 600 group by t order by t asc) d on t.t = d.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "status" < 200 or "status" >= 600 group by t order by t asc) e on t.t = e.t
```

- **后端响应码图**展示后端响应码的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(t.t, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "time", CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX", CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX", CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX", CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX", CASE WHEN e."其他" IS NOT NULL THEN e."其他" ELSE 0 END as "其他" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t from log group by t order by t asc) t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "upstream_status" >= 200 and "upstream_status" < 300 group by t order by t asc) a on t.t = a.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t, CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "upstream_status" >= 300 and "upstream_status" < 400 group by t order by t asc) b on t.t = b.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25), 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t,
```

```
CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "upstream_status" >= 400 and  
"upstream_status" < 500 group by t order by t asc) c on t.t =c.t left join (select  
TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t ,  
CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "upstream_status" >= 500 and  
"upstream_status" < 600 group by t order by t asc) d on t.t =d.t left join (select  
TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) , 'yyyy-MM-dd"T"HH:mm:ssZZ'), 'PT5S') as t ,  
CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "upstream_status" < 200 or  
"upstream_status" >= 600 group by t order by t asc) e on t.t =e.t
```

----结束

7.3.4.13 ER 仪表盘模板

企业路由器（Enterprise Router, ER）可以连接虚拟私有云（Virtual Private Cloud, VPC）或本地网络来构建中心辐射型组网，是云上大规格、高带宽、高性能的集中路由器。企业路由器使用边界网关协议（Border Gateway Protocol, BGP），支持路由学习、动态选路以及链路切换，极大的提升网络的可扩展性及运维效率，从而保证业务的连续性。

云日志服务支持日志采集向导一站式采集ER日志，支持多维度分析，并为ER日志配置结构化和仪表盘。该仪表盘主要展示ER日志的TOP20包数统计、TOP20流量统计、流日志条数等信息。

前提条件

- 已采集ER流量日志，详情请参见[企业路由器ER接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

ER 流量日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“ER仪表盘模板 > ER流量日志中心”，查看图表详情。

- 过滤实例ID图，所关联的查询分析语句如下所示：

```
SELECT DISTINCT(instance_id)
```
- 过滤连接ID图，所关联的查询分析语句如下所示：

```
SELECT DISTINCT(resource_id)
```
- 流量方向图展示为静态过滤器，可按照入方向和出方向的流量进行过滤，所关联的查询分析语句如下所示：
- 过滤源IP图，所关联的查询分析语句如下所示：

```
SELECT DISTINCT(srcaddr)
```
- 过滤目的IP图，所关联的查询分析语句如下所示：

```
SELECT DISTINCT(dstaddr)
```
- 过滤协议类型图，所关联的查询分析语句如下所示：

```
SELECT DISTINCT(protocol)
```
- TOP20包数统计**图表所关联的查询分析语句如下所示：

```
SELECT "srcaddr" as "源地址", "dstaddr" as "目的地址", sum("packets") as "包数", "resource_id" as "连接ID", "instance_id" as "实例ID" group by "instance_id", "resource_id", "srcaddr", "dstaddr" order by "包数" desc limit 20
```
- TOP20流量统计**图表所关联的查询分析语句如下所示：

```
SELECT "srcaddr" as "源地址", "dstaddr" as "目的地址", sum("bytes") as "字节数", "resource_id" as "连接ID", "instance_id" as "实例ID" group by "instance_id", "resource_id", "srcaddr", "dstaddr" order by "字节数" desc limit 20
```

- **流日志条数**图表所关联的查询分析语句如下所示：

```
select time_series(__time, 'PT1H', 'yyyy-MM-dd HH:mm:ss', '0', '+08:00') as "时间", count(*) as "流日志条数" group by "时间" order by "时间"
```
- **流日志详情**图表所关联的查询分析语句如下所示：

```
SELECT "instance_id" as "实例ID", "resource_id" as "连接ID", "project_id" as "项目ID", "srcaddr" as "源IP", "dstaddr" as "目的IP", "srcport" as "源端口", "dstport" as "目的端口", "protocol" as "协议类型", "direct" as "流量方向", "packets" as "包数", "bytes" as "字节数",  
TIME_FORMAT( MILLIS_TO_TIMESTAMP("start"*1000), 'yyyy-MM-dd HH:mm:ss', '+08:00') as "开始时间",  
TIME_FORMAT( MILLIS_TO_TIMESTAMP("end"*1000), 'yyyy-MM-dd HH:mm:ss', '+08:00') as "结束时间"
```

----结束

7.3.4.14 LTS 仪表盘模板

LTS仪表盘模板主要介绍OBS导入任务监控仪表盘、资源用量明细仪表盘、用量Top排序仪表盘。

- **OBS导入任务监控仪表盘**：支持查看OBS导入LTS的任务详情，例如任务ID、任务名称、成功条数、失败条数、处理速率等关键指标，帮助用户实时监控数据导入状态，快速定位异常问题，确保数据高效、完整导入LTS。
- **资源用量明细仪表盘**：需要开通“可观测For LTS”功能，详细操作请参考[设置LTS日志采集配额和使用量预警](#)。支持按时间范围、日志组ID或日志流ID筛选，查看日志流的资源用量明细（如原始日志流量、索引流量、存储量等），帮助用户快速识别资源消耗情况，及时发现异常流量，优化资源配置，提升资源利用率。
- **用量Top排序仪表盘**：需要开通“可观测For LTS”功能，详细操作请参考[设置LTS日志采集配额和使用量预警](#)。支持按时间范围、日志组ID或日志流ID筛选，查看TOP N日志组或日志流的资源用量（如原始日志流量、索引流量、存储量等），帮助用户快速识别高资源消耗情况，及时发现异常流量，优化日志采集策略与资源配置，降低无效成本。

说明

请先开通管道符搜索方式才能使用OBS导入任务监控仪表盘、资源用量明细仪表盘、用量Top排序仪表盘。如有需要请[提交工单](#)申请开通。

OBS 导入任务监控仪表盘

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“LTS仪表盘模板”。

步骤4 在仪表盘列表页面，单击“OBS导入任务监控仪表盘”仪表盘名称，查看图表详情。

- 过滤任务ID，所关联的查询分析语句如下所示：

```
* | select task_set_id group by task_set_id
```
- 过滤任务名称，所关联的查询分析语句如下所示：

```
* | select task_set_name group by task_set_name
```
- 成功条数图表所关联的查询分析语句如下所示：

```
* | select sum(max_read_line) as "成功条数" from (select max_by(read_lines,__time) as max_read_line  
group by task_set_id,period_times limit 10000)
```
- 失败条数图表所关联的查询分析语句如下所示：

```
* | select sum(max_failure_lines) as "失败条数" from (select max_by(failure_lines,__time) as  
max_failure_lines group by task_set_id,period_times limit 10000)
```

- 丢弃条数图表所关联的查询分析语句如下所示：

```
* | select sum(max_ignore_lines) as "丢弃条数" from (select max_by(ignore_lines,__time) as max_ignore_lines group by task_set_id,period_times limit 10000)
```
- 读流量图表所关联的查询分析语句如下所示：

```
* | select sum(max_in_flow)/1024.0/1024 as "读流量 MB" from (select max_by(in_flow,__time) as max_in_flow group by task_set_id,period_times limit 10000)
```
- 写流量图表所关联的查询分析语句如下所示：

```
* | select sum(max_out_flow)/1024.0/1024 as "写流量 MB" from (select max_by(out_flow,__time) as max_out_flow group by task_set_id,period_times limit 10000)
```
- 导入进度图表所关联的查询分析语句如下所示：

```
* | select max_by(case when total_files = 0 or total_files is null then 'NA' else cast(complete_files*1.0/total_files*100 as VARCHAR) end ,__time ) as progress
```
- 处理速率图表所关联的查询分析语句如下所示：

```
* | SELECT t AS "统计时间", ROUND(( CASE WHEN diff [ 1 ] IS NOT NULL AND diff [ 2 ] IS NOT NULL AND diff [ 1 ] - diff [ 2 ] > 0 THEN diff [ 1 ] - diff [ 2 ] WHEN diff [ 1 ] IS NOT NULL AND diff [ 2 ] IS NULL THEN diff [ 1 ] ELSE 0 END ) / 300,0) AS "成功条数/s" FROM ( SELECT t, ts_compare ( max_read_lines, 300 ) AS diff FROM ( SELECT t, sum( max_read_lines ) AS max_read_lines FROM ( SELECT from_unixtime( __time - __time % 300000 ) AS t, max_by ( read_lines, __time ) AS max_read_lines, task_set_id, period_times GROUP BY t, task_set_id, period_times LIMIT 10000 ) GROUP BY t ORDER BY t ) GROUP BY t ORDER BY t LIMIT 10000 ) LIMIT 10000
```
- 运行异常图表所关联的查询分析语句如下所示：

```
* | select task_set_name as "任务名称",task_set_id as "任务ID" , max_by(task_set_state,__time) as "任务状态" , from_unixtime(max(__time)) as "采集时间" , max_by(failure_lines,__time) as "失败读取条数" group by task_set_id,task_set_name HAVING "任务状态" = 'UNHEALTHY' OR "任务状态" = 'FAILURE' limit 10000
```
- 运行状态图表所关联的查询分析语句如下所示：

```
* | select task_set_name as "任务名称",task_set_id as "任务ID" , max_by(task_set_state,__time) as "任务状态" , max_by(read_lines,__time) as "成功条数",max_by(failure_lines,__time) as "失败条数" , max_by(in_flow,__time)/1024/1024 as "读流量/MB" , max_by(out_flow,__time)/1024/1024 as "写流量/MB",period_times as "周期" group by task_set_id,task_set_name,period_times order by max(__time),task_set_id,period_times limit 10000
```

----结束

资源用量明细仪表盘

步骤1 在仪表盘模板下方，选择“LTS仪表盘模板”。

步骤2 在仪表盘列表页面，单击“资源用量明细”仪表盘名称，查看图表详情。

若仪表盘中的部分图表展示异常，您可在索引配置页面单击“自动配置”更新索引字段，详细操作请参考[创建LTS日志索引](#)。索引更新完成后，返回仪表盘页面，查看图表显示正常。

- 过滤日志组ID，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select log_group_id from log group by log_group_id limit 100000
```
- 过滤日志组名称，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select log_group_name_alias from log group by log_group_name_alias limit 100000
```
- 过滤日志流ID，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select log_stream_id from log group by log_stream_id limit 100000
```
- 过滤日志流名称，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select log_stream_name_alias from log group by log_stream_name_alias limit 100000
```
- 原始日志流量/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
metric_name:"storage_traffic" | select sum(counter)/1024/1024 as "原始日志流量/MB"
```

- 索引流量(标准型)/MB图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"index_traffic" | select sum(counter)/1024/1024 as "索引流量/MB"
- 索引流量(搜索型)/MB图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"search_index_traffic" | select sum(counter)/1024/1024 as "索引流量/MB"
- DSL加工流量/MB图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"dsl_traffic" && counter:* | select sum(counter)/1024/1024 as "日志DSL加工流量/MB"
- 写流量(压缩后)/MB图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"write_traffic" | select sum(counter)/1024/1024 as "写流量/MB"
- 标准存储量(最新值)/MB图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"storage" && counter:* | select select sum(max_storage) as "标准存储量 (最新值) /MB"
from (select max_by(counter,__time)/1024/1024 as max_storage group by log_stream_id limit 10000)
- 冷存储量(最新值)/MB图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"cold_storage" && counter:* | select sum(max_cold_storage) as "冷存储量 (最新
值) /MB" from (select max_by(counter,__time)/1024/1024 as max_cold_storage group by
log_stream_id limit 10000)
- 基础转储流量/MB图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
(metric_name:"obs_transfer" or metric_name:"dms_transfer" or metric_name:"dis_transfer") | select
sum(counter)/1024.0/1024 as "基础转储流量/MB"
- 高级转储流量/MB图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
(metric_name:"dws_transfer" or metric_name:"dli_transfer" or metric_name:"obs_orc_transfer") |
select sum(counter)/1024/1024 as "高级转储流量/MB"
- 原始日志流量图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"storage_traffic" | select from_unixtime(__time - __time%60000) as "时间",sum(counter/
1024/1024) as "原始日志流量/MB" group by "时间" order by "时间" limit 10000
- 索引流量 (标准型) 图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"index_traffic" | select from_unixtime(__time - __time%60000) as "时间",sum(counter)/
1024/1024 as "索引流量/MB" group by "时间" order by "时间" limit 10000
- 索引流量 (搜索型) 图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"search_index_traffic" | select from_unixtime(__time - __time%60000) as "时间"
",sum(counter)/1024/1024 as "索引流量/MB" group by "时间" order by "时间" limit 10000
- DSL加工流量图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"dsl_traffic" | select from_unixtime(__time - __time%60000) as "时间",sum(counter)/
1024/1024 as "日志DSL加工流量/MB" group by "时间" order by "时间" limit 10000
- 写流量 (压缩后) 图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"write_traffic" | select from_unixtime(__time - __time%60000) as "时间",sum(counter)/
1024/1024 as "写流量/MB" group by "时间" order by "时间" limit 10000
- 标准存储量 (最新值) 图表所关联的查询分析语句如下所示:
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and
metric_name:"storage"| select from_unixtime(__time - __time%60000) as "时间"
",max_by(counter,__time)/1024/1024 as "标准存储量 (最新值) /MB" group by "时间" order by "时间"
limit 10000
- 冷存储量 (最新值) 图表所关联的查询分析语句如下所示:


```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
metric_name:"cold_storage" | select from_unixtime(__time - __time%60000) as "时间"  
",max_by(counter,__time)/1024/1024 as "冷存储量 (最新值)/MB" group by "时间" order by "时间"  
limit 10000
```

- 基础转储流量图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
(metric_name:"obs_transfer" or metric_name:"dms_transfer" or metric_name:"dis_transfer") | select  
from_unixtime(__time - __time%60000) as "时间",sum(counter)/1024/1024 as "基础转储流量/MB"  
group by "时间" order by "时间" limit 10000
```

- 高级转储流量图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
(metric_name:"dws_transfer" or metric_name:"dli_transfer" or metric_name:"obs_orc_transfer") |  
select from_unixtime(__time - __time%60000) as "时间",sum(counter)/1024/1024 as "高级转储流  
量/MB" group by "时间" order by "时间" limit 10000
```

----结束

用量 Top 排序仪表盘

步骤1 在仪表盘模板下方，选择“LTS仪表盘模板”。

步骤2 在仪表盘列表页面，单击“用量Top排序”仪表盘名称，查看图表详情。

若仪表盘中的部分图表展示异常，您可在索引配置页面单击“自动配置”更新索引字段，详细操作请参考[创建LTS日志索引](#)。索引更新完成后，返回仪表盘页面，查看图表显示正常。

- 过滤日志组ID，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select log_group_id  
from log group by log_group_id limit 100000
```

- 过滤日志组名称，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select  
log_group_name_alias from log group by log_group_name_alias limit 100000
```

- 过滤日志流ID，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select log_stream_id  
from log group by log_stream_id limit 100000
```

- 过滤日志流名称，所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select  
log_stream_name_alias from log group by log_stream_name_alias limit 100000
```

- 原始日志流量/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
metric_name:"storage_traffic" | select sum(counter)/1024/1024 as "原始日志流量/MB"
```

- 索引流量(标准型)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
metric_name:"index_traffic" | select sum(counter)/1024/1024 as "索引流量/MB"
```

- 索引流量(搜索型)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
metric_name:"search_index_traffic" | select sum(counter)/1024/1024 as "索引流量/MB"
```

- DSL加工流量/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
metric_name:"dsl_traffic" && counter:* | select sum(counter)/1024/1024 as "日志DSL加工流量/MB"
```

- 写流量(压缩后)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
metric_name:"write_traffic" | select sum(counter)/1024/1024 as "写流量/MB"
```

- 标准存储量(最新值)/MB图表所关联的查询分析语句如下所示：

```
not log_group_id:"__default__" and not log_group_name_alias:"__default__" and  
metric_name:"storage" && counter:* | select sum(max_storage) as "标准存储量 (最新值) /MB" from  
(select max_by(counter,__time)/1024/1024 as max_storage group by log_stream_id limit 10000 )
```


- 冷存储量(最新值)/MB图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" and not log_group_name_alias:"__default__" and metric_name:"cold_storage" && counter:* | select sum(max_cold_storage) as "冷存储量 (最新值) /MB" from (select max_by(counter,_time)/1024/1024 as max_cold_storage group by log_stream_id limit 10000 )`
- 基础转储流量/MB图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" and not log_group_name_alias:"__default__" and (metric_name:"obs_transfer" or metric_name:"dms_transfer" or metric_name:"dis_transfer") | select sum(counter)/1024/1024 as "基础转储流量/MB"`
- 高级转储流量/MB图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" and not log_group_name_alias:"__default__" and (metric_name:"dws_transfer" or metric_name:"dli_transfer" or metric_name:"obs_orc_transfer") | select sum(counter)/1024/1024 as "高级转储流量/MB"`
- TOP日志组用量图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select log_group_id as "日志组ID", sum(case when metric_name = 'storage_traffic' then counter else 0 end)*1.0/1024/1024 as "原始日志流量/MB", sum(case when metric_name = 'index_traffic' then counter else 0 end)*1.0/1024/1024 as "索引流量 (标准型) /MB", sum(case when metric_name = 'search_index_traffic' then counter else 0 end)*1.0/1024/1024 as "索引流量 (搜索型) /MB", sum(case when metric_name = 'write_traffic' then counter else 0 end)*1.0/1024/1024 as "写流量 (压缩后) /MB", sum(case when metric_name = 'storage' then counter else 0 end)*1.0/1024/1024 as "标准存储量 (最新值) /MB", sum(case when metric_name = 'cold_storage' then counter else 0 end)*1.0/1024/1024 as "冷存储量 (最新值) /MB" group by log_group_id order by "原始日志流量/MB" desc`
- TOP日志流用量图表所关联的查询分析语句如下所示:
`not log_group_id:"__default__" and not log_group_name_alias:"__default__" | select log_stream_id as "日志流ID", sum(case when metric_name = 'storage_traffic' then counter else 0 end)*1.0/1024/1024 as "原始日志流量/MB", sum(case when metric_name = 'index_traffic' then counter else 0 end)*1.0/1024/1024 as "索引流量 (标准型) /MB", sum(case when metric_name = 'search_index_traffic' then counter else 0 end)*1.0/1024/1024 as "索引流量 (搜索型) /MB", sum(case when metric_name = 'write_traffic' then counter else 0 end)*1.0/1024/1024 as "写流量 (压缩后) /MB", sum(case when metric_name = 'storage' then counter else 0 end)*1.0/1024/1024 as "标准存储量 (最新值) /MB", sum(case when metric_name = 'cold_storage' then counter else 0 end)*1.0/1024/1024 as "冷存储量 (最新值) /MB" group by log_stream_id order by "原始日志流量/MB" desc`

----结束

7.3.4.15 METRIC 仪表盘模板

用户在LTS页面只需按照业务需要创建指标规则即可生成自己的统计报表，设置单个日志过滤条件或通过添加关联关系和添加组设置多个日志过滤条件，保留符合条件的日志，对用户特定时间范围内已结构化的日志进行动态统计，并将统计结果动态呈现到aom的Prometheus实例，操作简单且功能强大。

前提条件

- 已创建日志生成指标，详情请参见[日志生成指标（邀测）](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

日志生成指标任务监控中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“METRIC仪表盘模板 > 日志生成指标任务监控中心”，查看图表详情。

- 过滤规则ID，所关联的查询分析语句如下所示：

```
select distinct(task_set)
```

- **输入行数图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN "input" < 1000 THEN concat( cast( "input" AS VARCHAR ), '行' ) WHEN  
"input" < 1000 * 1000 THEN concat( cast( round( "input"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN  
"input" < 1000000000 THEN concat( cast( round( "input"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )  
WHEN "input"/ 1000.0 < 1000000000 THEN concat( cast( round( "input"/ 1000 / 1000000.0, 1 ) AS  
VARCHAR ), '十亿行' ) ELSE concat( cast( round( "input"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS  
VARCHAR ), '万亿行' ) END AS "total" from (select sum("input") as "input")
```

- **输出行数图表所关联的查询分析语句如下所示**

```
SELECT CASE WHEN "output" < 1000 THEN concat( cast( "output" AS VARCHAR ), '行' ) WHEN  
"output" < 1000 * 1000 THEN concat( cast( round( "output"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN  
"output" < 1000000000 THEN concat( cast( round( "output"/ 1000000.0, 1 ) AS VARCHAR ), '百万  
行' ) WHEN "output"/ 1000.0 < 1000000000 THEN concat( cast( round( "output"/ 1000 / 1000000.0,  
1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "output"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS  
VARCHAR ), '万亿行' ) END AS "total" from (select sum("output") as "output")
```

- **满足过滤条件行数图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN "filters" < 1000 THEN concat( cast( "filters" AS VARCHAR ), '行' ) WHEN  
"filters" < 1000 * 1000 THEN concat( cast( round( "filters"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN  
"filters" < 1000000000 THEN concat( cast( round( "filters"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )  
WHEN "filters"/ 1000.0 < 1000000000 THEN concat( cast( round( "filters"/ 1000 / 1000000.0, 1 ) AS  
VARCHAR ), '十亿行' ) ELSE concat( cast( round( "filters"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS  
VARCHAR ), '万亿行' ) END AS "total" from (select sum("filters") as "filters")
```

- **不满足过滤条件行数图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN "filter_drops" < 1000 THEN concat( cast( "filter_drops" AS VARCHAR ), '行' )  
WHEN "filter_drops" < 1000 * 1000 THEN concat( cast( round( "filter_drops"/ 1000, 1 ) AS  
VARCHAR ), '千行' ) WHEN "filter_drops" < 1000000000 THEN concat( cast( round( "filter_drops"/  
1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "filter_drops"/ 1000.0 < 1000000000 THEN  
concat( cast( round( "filter_drops"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE  
concat( cast( round( "filter_drops"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END  
AS "total" from (select sum("filter_drops") as "filter_drops")
```

- **采样行数图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN "samples" < 1000 THEN concat( cast( "samples" AS VARCHAR ), '行' ) WHEN  
"samples" < 1000 * 1000 THEN concat( cast( round( "samples"/ 1000, 1 ) AS VARCHAR ), '千行' )  
WHEN "samples" < 1000000000 THEN concat( cast( round( "samples"/ 1000000.0, 1 ) AS  
VARCHAR ), '百万行' ) WHEN "samples"/ 1000.0 < 1000000000 THEN  
concat( cast( round( "samples"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE  
concat( cast( round( "samples"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS  
"total" from (select sum("samples") as "samples")
```

- **采样丢弃行数图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN "sample_drops" < 1000 THEN concat( cast( "sample_drops" AS VARCHAR ),  
'行' ) WHEN "sample_drops" < 1000 * 1000 THEN concat( cast( round( "sample_drops"/ 1000, 1 ) AS  
VARCHAR ), '千行' ) WHEN "sample_drops" < 1000000000 THEN  
concat( cast( round( "sample_drops"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN  
"sample_drops"/ 1000.0 < 1000000000 THEN concat( cast( round( "sample_drops"/ 1000 / 1000000.0,  
1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "sample_drops"/ 1000.0 / 1000 / 1000 / 1000,  
1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum( "sample_drops" ) as "sample_drops")
```

- **超过日志时间范围行数图表所关联的查询分析语句如下所示：**

```
SELECT CASE WHEN "out_of_bounds" < 1000 THEN concat( cast( "out_of_bounds" AS VARCHAR ),  
'行' ) WHEN "out_of_bounds" < 1000 * 1000 THEN concat( cast( round( "out_of_bounds"/ 1000, 1 )  
AS VARCHAR ), '千行' ) WHEN "out_of_bounds" < 1000000000 THEN  
concat( cast( round( "out_of_bounds"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN  
"out_of_bounds"/ 1000.0 < 1000000000 THEN concat( cast( round( "out_of_bounds"/ 1000 /  
1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "out_of_bounds"/ 1000.0 / 1000 /  
1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("out_of_bounds") as  
"out_of_bounds")
```

- **执行记录图表所关联的查询分析语句如下所示：**

```
select TIME_FORMAT( "__time", 'yyyy-MM-dd HH:mm:ss:SSS', '+08:00') as "统计时间", sum("input") as  
"输入行数",sum("output") as "输出行数",sum("filters") as "满足过滤条件行数",sum("filter_drops") as  
"不满足过滤条件行数",sum("samples") as "采样行数",sum("sample_drops") as "采样丢弃行数",  
sum("out_of_bounds") as "超过日志时间范围行数" group by __time order by __time desc limit 1000
```

----结束

7.3.4.16 NGINX-INGRESS 仪表盘模板

日志服务支持采集NGINX-INGRESS日志，并进行多维度分析。云日志服务支持日志采集向导一站式采集NGINX-INGRESS日志，并为NGINX-INGRESS日志配置结构化和仪表盘。Nginx Ingress是专为Kubernetes集群设计的Ingress控制器，用于管理外部访问到集群内服务的路由规则。

NGINX-INGRESS仪表盘模板支持[查看NGINX-INGRESS秒级监控](#)、[查看NGINX-INGRESS访问中心](#)和[查看NGINX-INGRESS监控中心](#)。

前提条件

- 已采集NGINX-INGRESS日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 NGINX-INGRESS 秒级监控

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX秒级监控”，查看图表详情。

- 过滤appName，所关联的查询分析语句如下所示：

```
select appName group by appName
```

- QPS图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT(TIME_CEIL(__time,'PT1S'),'yyyy-MM-dd HH:mm:ss','+08:00') AS __time_,  
COUNT(*) as QPS from log where req_id is not null group by __time_
```

- 成功率图表所关联的查询分析语句如下所示：

```
select __time,round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select  
TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, sum(case when  
status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log where req_id is not null group  
by __time)
```

- 延迟图表所关联的查询分析语句如下所示：

```
select __time,round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟",  
round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟"  
from (select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time,  
avg(request_time)* 1000 as '访问延迟',avg(upstream_response_time)* 1000 as 'Upstream延迟' from  
log where req_id is not null group by __time)
```

- 流量图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time ,  
sum("request_length") as "请求流量", sum("body_bytes_sent") as "返回body流量" where req_id is not  
null group by __time
```

- 状态码图表所关联的查询分析语句如下所示：

```
select t.t as "time",  
CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",  
CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",  
CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",  
CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",  
CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他"  
from (select TIME_CEIL(__time,'PT5S') as t from log where req_id is not null group by t order by t  
asc ) t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from  
log WHERE "status" >= 200 and "status" < 300 and req_id is not null group by t order by t asc ) a on  
t.t =a.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from  
log WHERE "status" >= 300 and "status" < 400 and req_id is not null group by t order by t asc) b on  
t.t =b.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from  
log WHERE "status" >= 400 and "status" < 500 and req_id is not null group by t order by t asc) c on  
t.t =c.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from  
log WHERE "status" >= 500 and "status" < 600 and req_id is not null group by t order by t asc) d on  
t.t =d.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from
```

```
log WHERE "status" < 200 or "status" >= 600 and req_id is not null group by t order by t asc) e on t.t =e.t
```

- 后端响应码图表所关联的查询分析语句如下所示：

```
select t.t as "time",
CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",
CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",
CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",
CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",
CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他"
from (
select TIME_CEIL(__time,'PT5S') as t from log where req_id is not null group by t order by t asc
) t
left join(
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE
"upstream_status" >= 200 and "upstream_status" < 300 and req_id is not null group by t order by t
asc) a
on t.t = a.t
left join (
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE
"upstream_status" >= 300 and "upstream_status" < 400 and req_id is not null group by t order by t
asc) b
on t.t =b.t
left join (
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE
"upstream_status" >= 400 and "upstream_status" < 500 and req_id is not null group by t order by t
asc) c
on t.t =c.t
left join (
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE
"upstream_status" >= 500 and "upstream_status" < 600 and req_id is not null group by t order by t
asc) d
on t.t =d.t
left join (
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE
"upstream_status" < 200 or "upstream_status" >= 600 and req_id is not null group by t order by t asc)
e
on t.t =e.t
```

----结束

查看 NGINX-INGRESS 访问中心

步骤1 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX访问中心”，查看图表详情。

- 过滤appName，所关联的查询分析语句如下所示：

```
select appName group by appName
```

- PV对比昨日图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" ,
86400) as diff from (select count(1) as "pv" from log where req_id is not null))
```

- PV对比上周图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" ,
604800) as diff from (select count(1) as "pv" from log where req_id is not null))
```

- UV对比昨日图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" ,
86400) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log where req_id is
not null))
```

- UV对比上周图表所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" ,
604800) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log where req_id
is not null))
```

- 访问量PV分布(中国)图表所关联的查询分析语句如下所示：

```
select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY (remote_addr)
= '中国' and req_id is not null group by province HAVING province not in ('','保留地址','*')
```

- **访问量PV分布(世界)图表**所关联的查询分析语句如下所示:

```
SELECT ip_to_country(remote_addr) as country,COUNT(1) as PV where req_id is not null GROUP BY country HAVING country not in ('','保留地址','*')
```
- **访问量UV分布(中国)图表**所关联的查询分析语句如下所示:

```
select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as UV where IP_TO_COUNTRY (remote_addr) = '中国' and req_id is not null group by province HAVING province not in ('','保留地址','*')
```
- **访问量UV分布(世界)图表**所关联的查询分析语句如下所示:

```
select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv where req_id is not null group by country HAVING country not in ('','保留地址','*')
```
- **平均时延分布（中国）图表**所关联的查询分析语句如下所示:

```
SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(remote_addr) as province,avg(request_time) * 1000 AS "平均延迟(ms)" WHERE IP_TO_COUNTRY (remote_addr) = '中国' and req_id is not null GROUP BY province HAVING province not in ('','保留地址','*'))
```
- **平均时延分布（世界）图表**所关联的查询分析语句如下所示:

```
SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(remote_addr) as country,avg(request_time) * 1000 AS "平均延迟(ms)" where req_id is not null GROUP BY country HAVING country not in ('','保留地址','*'))
```
- **今日PV/UV图表**所关联的查询分析语句如下所示:

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(_time_,PT600S) AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE _time_ <= CURRENT_TIMESTAMP and req_id is not null group by _time_ order by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```
- **7日PV/UV图表**所关联的查询分析语句如下所示:

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(_time_,PT600S) AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE _time_ <= CURRENT_TIMESTAMP and _time_ >= INTERVAL '7' DAY and req_id is not null group by _time_ order by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```
- **区域访问TOP10(省份)图表**所关联的查询分析语句如下所示:

```
select ip_to_province(remote_addr) as "province", count(1) as "访问次数" where req_id is not null group by "province" HAVING "province" <> '-1' order by "访问次数" asc limit 10
```
- **区域访问TOP10(城市)图表**所关联的查询分析语句如下所示:

```
select ip_to_city(remote_addr) as "city", count(1) as "访问次数" where req_id is not null group by "city" HAVING "city" <> '-1' order by "访问次数" asc limit 10
```
- **Host访问TOP10图表**所关联的查询分析语句如下所示:

```
select remote_addr as "Host", count(1) as "PV" where req_id is not null group by "Host" order by "PV" asc limit 10
```
- **UserAgent访问TOP10图表**所关联的查询分析语句如下所示:

```
select http_user_agent as "UserAgent", count(1) as "PV" where req_id is not null group by "UserAgent" order by "PV" asc limit 10
```
- **设备占比(终端)图表**所关联的查询分析语句如下所示:

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then '移动端' else 'PC端' end as type , count(1) as total where req_id is not null group by type
```
- **设备占比(系统)图表**所关联的查询分析语句如下所示:

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total where req_id is not null group by type HAVING type != 'other'
```
- **TOP URL图表**所关联的查询分析语句如下所示:

```
select request_uri , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV, round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" where req_id is not null group by request_uri ORDER by PV desc
```
- **TOP 访问IP图表**所关联的查询分析语句如下所示:

```
select remote_addr as "来源IP",ip_to_country(remote_addr) as "国家",ip_to_province(remote_addr) as "省份",ip_to_city(remote_addr) as "城市",ip_to_provider(remote_addr) as "运营商",count(1) as
```

```
"PV",http_user_agent as "UserAgent采样",request_uri as "URL采样" where req_id is not null group by
remote_addr,http_user_agent,request_uri ORDER by "PV" desc
```

----结束

查看 NGINX-INGRESS 监控中心

步骤1 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX监控中心”，查看图表详情。

- 过滤appName，所关联的查询分析语句如下所示：
select appName group by appName
- 访问量PV图表所关联的查询分析语句如下所示：
SELECT TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time_,PV FROM (SELECT TIME_CEIL (__time, 'PT300S') AS _time_, count(1) AS PV FROM log where req_id is not null GROUP BY _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- 请求成功率图表所关联的查询分析语句如下所示：
select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt where req_id is not null
- 平均延迟图表所关联的查询分析语句如下所示：
select round(avg(request_time) * 1000, 3) as cnt where req_id is not null
- 4XX请求数图表所关联的查询分析语句如下所示：
SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500 and req_id is not null
- 404请求数图表所关联的查询分析语句如下所示：
SELECT COUNT(1) as cnt WHERE "status" = 404 and req_id is not null
- 429请求数图表所关联的查询分析语句如下所示：
SELECT COUNT(1) as cnt WHERE "status" = 429 and req_id is not null
- 504请求数图表所关联的查询分析语句如下所示：
SELECT COUNT(1) as cnt WHERE "status" = 504 and req_id is not null
- 5XX请求数图表所关联的查询分析语句如下所示：
SELECT TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time_,cnt FROM (SELECT TIME_CEIL (__time, 'PT300S') AS _time_, count(1) AS cnt FROM log where "status" >= 500 and req_id is not null GROUP BY _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- 状态码分布图表所关联的查询分析语句如下所示：
SELECT status, COUNT(1) AS rm where req_id is not null GROUP BY status
- 访问量UV图表所关联的查询分析语句如下所示：
SELECT TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time_,UV FROM (select TIME_CEIL(__time,'PT600S') AS _time_, APPROX_COUNT_DISTINCT(remote_addr) as UV from log where req_id is not null group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- 流量图表所关联的查询分析语句如下所示：
select TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') AS _time_,round(CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2) AS "入流量",round(CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2) AS "出流量" FROM (SELECT TIME_CEIL(__time,'PT600S') AS _time_,sum(request_length) / 1024.0 AS "入流量",sum(body_bytes_sent) / 1024.0 AS "出流量" where req_id is not null group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- 访问失败率图表所关联的查询分析语句如下所示：
SELECT TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time_,round(CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2) AS "失败率",round(CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2) AS "5XX比例" from (select TIME_CEIL(__time,'PT600S') AS _time_,sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率', sum(case when status >= 500 THEN 1 ELSE 0 END)*100.0/COUNT(1) as '5XX比例' where req_id is not null group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
- 延迟图表所关联的查询分析语句如下所示：
select TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') as _time_,round(CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2) AS "平均",round(CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2) AS "P50",round(CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2) AS "P90",round(CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2) AS "P99",round(CASE WHEN "P999" > 0 THEN

```
"P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(_time,'PT600S') as
_time_avg(request_time) * 1000 as "平均", APPROX_QUANTILE_DS("request_time", 0.50)*1000 as
"P50", APPROX_QUANTILE_DS("request_time", 0.90)*1000 as
"P90", APPROX_QUANTILE_DS("request_time", 0.99)*1000 as
"P99",APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as 'P9999' where req_id is not null group
by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **Host请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT "remote_addr", pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END,
3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出
流量(KB)" FROM ( SELECT "remote_addr", count( 1 ) AS pv, APPROX_COUNT_DISTINCT
( remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS
"访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS
"入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "remote_addr" != " and
req_id is not null GROUP BY "remote_addr" ) ORDER BY pv DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT "remote_addr", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "remote_addr", count( 1 ) AS pv, sum( CASE WHEN
"status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) *
1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != " and
req_id is not null GROUP BY "remote_addr" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT "remote_addr", pv,round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0
END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0
END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0
END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "remote_addr", count( 1 ) AS pv, sum( CASE WHEN
"status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) *
1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != " and
req_id is not null GROUP BY "remote_addr" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT request_uri, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END,
3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出
流量(KB)" FROM ( SELECT request_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr )
AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率
(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",
sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "remote_addr" != " and req_id is not null
GROUP BY request_uri ) ORDER BY pv desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT request_uri, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,
2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,
3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 )
AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS
"P99延迟(ms)" FROM (SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN
1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟
(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != " and
req_id is not null GROUP BY request_uri ) ORDER BY "平均延迟(ms)" desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END,
2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,
3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 )
AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS
"P99延迟(ms)" FROM( SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN
1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟
(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
```



```
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != "" and req_id is not null GROUP BY request_uri )ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "remote_addr" != "" and req_id is not null GROUP BY addr having length(upstream_addr) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT upstream_addr as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != "" and req_id is not null GROUP BY addr having length(upstream_addr) > 2) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "remote_addr" != "" and req_id is not null GROUP BY addr having length(upstream_addr) > 2)ORDER BY "访问失败率(%)" desc
```

----结束

7.3.4.17 NGINX 仪表盘模板

日志服务支持采集NGINX日志，并进行多维度分析。云日志服务支持日志采集向导一站式采集NGINX日志，并为NGINX日志配置结构化和仪表盘。Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。

NGINX仪表盘模板支持[查看NGINX秒级监控](#)、[查看NGINX访问中心](#)和[查看NGINX监控中心](#)。

前提条件

- 已采集NGINX日志。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 NGINX 秒级监控

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX秒级监控”，查看图表详情。

- **QPS**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT(TIME_CEIL(__time,'PT1S'),'yyyy-MM-dd HH:mm:ss','+08:00') AS _time_, COUNT(*) as QPS from log group by _time_
```


- **成功率**图表所关联的查询分析语句如下所示：

```
select __time,round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select  
TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, sum(case when  
status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group by __time)
```

- **延迟**图表所关联的查询分析语句如下所示：

```
select __time,round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟",  
round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟"  
from (select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time,  
avg(request_time)* 1000 as '访问延迟',avg(upstream_response_time)* 1000 as 'Upstream延迟' from  
log group by __time)
```

- **流量**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time ,  
sum("request_length") as "请求流量", sum("body_bytes_sent") as "返回body流量" group by __time
```

- **状态码**图表所关联的查询分析语句如下所示：

```
select t.t as "time",  
CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",  
CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",  
CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",  
CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",  
CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他"  
from (select TIME_CEIL(__time,'PT5S') as t from log group by t order by t asc ) t left join (select  
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "status" >=  
200 and "status" < 300 group by t order by t asc ) a on t.t =a.t left join (select  
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "status" >=  
300 and "status" < 400 group by t order by t asc ) b on t.t =b.t left join (select  
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "status" >=  
400 and "status" < 500 group by t order by t asc ) c on t.t =c.t left join (select  
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "status" >=  
500 and "status" < 600 group by t order by t asc ) d on t.t =d.t left join (select  
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "status" <  
200 or "status" >= 600 group by t order by t asc ) e on t.t =e.t
```

- **后端响应码**图表所关联的查询分析语句如下所示：

```
select t.t as "time",  
CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",  
CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",  
CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",  
CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",  
CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他"  
from (  
select TIME_CEIL(__time,'PT5S') as t from log group by t order by t asc  
) t  
left join(  
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE  
"upstream_status" >= 200 and "upstream_status" < 300 group by t order by t asc ) a  
on t.t = a.t  
left join (  
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE  
"upstream_status" >= 300 and "upstream_status" < 400 group by t order by t asc ) b  
on t.t =b.t  
left join (  
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE  
"upstream_status" >= 400 and "upstream_status" < 500 group by t order by t asc ) c  
on t.t =c.t  
left join (  
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE  
"upstream_status" >= 500 and "upstream_status" < 600 group by t order by t asc ) d  
on t.t =d.t  
left join (  
select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE  
"upstream_status" < 200 or "upstream_status" >= 600 group by t order by t asc ) e  
on t.t =e.t
```

----结束

查看 NGINX 访问中心

步骤1 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX访问中心”，查看图表详情。

- **PV对比昨日图表**所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "pv", 86400) as diff from (select count(1) as "pv" from log))
```
- **PV对比上周图表**所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "pv", 604800) as diff from (select count(1) as "pv" from log))
```
- **UV对比昨日图表**所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "uv", 86400) as diff from (select APPROX_COUNT_DISTINCT(my_remote_addr) as "uv" from log))
```
- **UV对比上周图表**所关联的查询分析语句如下所示：

```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from (select compare( "uv", 604800) as diff from (select APPROX_COUNT_DISTINCT(my_remote_addr) as "uv" from log))
```
- **访问量PV分布(中国)图表**所关联的查询分析语句如下所示：

```
select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','*')
```
- **访问量PV分布(世界)图表**所关联的查询分析语句如下所示：

```
SELECT ip_to_country(remote_addr) as country,COUNT(1) as PV GROUP BY country HAVING country not in ('','保留地址','*')
```
- **访问量UV分布(中国)图表**所关联的查询分析语句如下所示：

```
select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as UV where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','*')
```
- **访问量UV分布(世界)图表**所关联的查询分析语句如下所示：

```
select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv group by country HAVING country not in ('','保留地址','*')
```
- **平均时延分布（中国）图表**所关联的查询分析语句如下所示：

```
SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(remote_addr) as province,avg(request_time) * 1000 AS "平均延迟(ms)"WHERE IP_TO_COUNTRY (remote_addr) = '中国'GROUP BY province HAVING province not in ('','保留地址','*'))
```
- **平均时延分布（世界）图表**所关联的查询分析语句如下所示：

```
SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(remote_addr) as country,avg(request_time) * 1000 AS "平均延迟(ms)" GROUP BY country HAVING country not in ('','保留地址','*'))
```
- **今日PV/UV图表**所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(_time_,PT600S) AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(my_remote_addr) as UV from log WHERE _time_ <= CURRENT_TIMESTAMP and _time_ >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR group by _time_ order by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```
- **7日PV/UV图表**所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(_time_,PT600S) AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE _time_ <= CURRENT_TIMESTAMP and _time_ >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR - INTERVAL '7' DAY group by _time_ order by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```
- **区域访问TOP10(省份)图表**所关联的查询分析语句如下所示：

```
select ip_to_province(remote_addr) as "province", count(1) as "访问次数" group by "province" HAVING "province" <> '-1' order by "访问次数" asc limit 10
```
- **区域访问TOP10(城市)图表**所关联的查询分析语句如下所示：

```
select ip_to_city(remote_addr) as "city", count(1) as "访问次数" group by "city" HAVING "city" <> '-1' order by "访问次数" asc limit 10
```
- **Host访问TOP10图表**所关联的查询分析语句如下所示：

```
select host as "Host", count(1) as "PV" group by "Host" order by "PV" asc limit 10
```

- **UserAgent访问TOP10**图表所关联的查询分析语句如下所示:

```
select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" asc limit 10
```

- **设备占比(终端)**图表所关联的查询分析语句如下所示:

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type, count(1) as total group by type
```

- **设备占比(系统)**图表所关联的查询分析语句如下所示:

```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when  
regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type, count(1) as  
total group by type HAVING type != 'other'
```

- **TOP URL**图表所关联的查询分析语句如下所示:

```
select request_uri, count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV, round(sum( case  
when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group by request_uri  
ORDER by PV desc
```

- **TOP 访问IP**图表所关联的查询分析语句如下所示:

```
select remote_addr as "来源IP", ip_to_country(remote_addr) as "国家", ip_to_province(remote_addr) as  
"省份", ip_to_city(remote_addr) as "城市", ip_to_provider(remote_addr) as "运营商", count(1) as  
"PV", http_user_agent as "UserAgent采样", request_uri as "URL采样" group by  
remote_addr, http_user_agent, request_uri ORDER by "PV" desc
```

----结束

查看 NGINX 监控中心

步骤1 在仪表盘模板下方，选择“NGINX仪表盘模板 > NGINX监控中心”，查看图表详情。

- **访问量PV**图表所关联的查询分析语句如下所示:

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_, PV FROM ( SELECT  
TIME_CEIL( _time_, 'PT300S' ) AS _time_, count( 1 ) AS PV FROM log GROUP BY _time_ ) WHERE  
_time_ <= CURRENT_TIMESTAMP LIMIT 100 OFFSET 1
```

- **请求成功率**图表所关联的查询分析语句如下所示:

```
select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```

- **平均延迟**图表所关联的查询分析语句如下所示:

```
select round(avg(request_time) * 1000, 3) as cnt
```

- **4XX请求数**图表所关联的查询分析语句如下所示:

```
SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
```

- **404请求数**图表所关联的查询分析语句如下所示:

```
SELECT COUNT(1) as cnt WHERE "status" = 404
```

- **429请求数**图表所关联的查询分析语句如下所示:

```
SELECT COUNT(1) as cnt WHERE "status" = 429
```

- **504请求数**图表所关联的查询分析语句如下所示:

```
SELECT COUNT(1) as cnt WHERE "status" = 504
```

- **5XX请求数**图表所关联的查询分析语句如下所示:

```
select TIME_FORMAT(TIME_CEIL(_time_, 'PT300S'), 'yyyy-MM-dd HH:mm:ss', '+08:00') AS _time_,  
count(1) as cnt where "status" >= 500 group by _time_
```

- **状态码分布**图表所关联的查询分析语句如下所示:

```
SELECT status, COUNT(1) AS rm GROUP BY status
```

- **访问量UV**图表所关联的查询分析语句如下所示:

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_, UV FROM (select  
TIME_CEIL(_time_, 'PT600S') AS _time_, APPROX_COUNT_DISTINCT(remote_addr) as UV from log  
group by _time_ ) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **流量**图表所关联的查询分析语句如下所示:

```
select TIME_FORMAT(_time_, 'yyyy-MM-dd HH:mm:ss', '+08:00') AS _time_, round( CASE WHEN "入流  
量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量", round( CASE WHEN "出流量" > 0 THEN "出流量"  
ELSE 0 END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(_time_, 'PT600S') AS
```

```
_time_sum(request_length) / 1024.0 AS "入流量",sum(bytes_sent) / 1024.0 AS "出流量" group by  
_time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **访问失败率**图表所关联的查询分析语句如下所示:

```
SELECT TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,round( CASE WHEN "失  
败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率",round( CASE WHEN "5XX比例" > 0 THEN "5XX  
比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(_time_,PT600S) AS _time_,sum(case  
when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率', sum(case when status >=500  
THEN 1 ELSE 0 END)*100.0/COUNT(1) as '5XX比例' group by _time_) WHERE _time_ <=  
CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **延迟**图表所关联的查询分析语句如下所示:

```
select TIME_FORMAT( _time_ , 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,round( CASE WHEN "平  
均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均",round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0  
END, 2 ) AS "P50",round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90",round( CASE  
WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99",round( CASE WHEN "P9999" > 0 THEN  
"P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(_time_,PT600S) as  
_time_,avg(request_time) * 1000 as "平均", APPROX_QUANTILE_DS("request_time", 0.50)*1000 as  
"P50", APPROX_QUANTILE_DS("request_time", 0.90)*1000 as  
"P90",APPROX_QUANTILE_DS("request_time", 0.99)*1000 as  
"P99",APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as "P9999" group by _time_) WHERE  
_time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- **Host请求TOP**图表所关联的查询分析语句如下所示:

```
SELECT "host", pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,  
2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,  
3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS  
"入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量  
(KB)" FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr ) AS uv,  
sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",  
avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",  
sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY "host" ) ORDER BY pv  
DESC
```

- **Host延迟TOP**图表所关联的查询分析语句如下所示:

```
SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )  
AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0  
END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟  
(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图表所关联的查询分析语句如下所示:

```
SELECT "host", pv,round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 )  
AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS  
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99  
延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE  
0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " " GROUP BY  
"host" ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图表所关联的查询分析语句如下所示:

```
SELECT request_uri, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0  
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0  
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END,  
3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出  
流量(KB)" FROM ( SELECT request_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr )  
AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率  
(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",  
sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " " GROUP BY request_uri ) ORDER BY  
pv desc
```

- **URL延迟TOP**图表所关联的查询分析语句如下所示:

```
SELECT request_uri, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,  
2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,  
3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 )
```

```
AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS  
"P99延迟(ms)" FROM (SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN  
1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟  
(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " GROUP BY  
request_uri ) ORDER BY "平均延迟(ms)" desc
```

- **URL失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END,  
2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,  
3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 )  
AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS  
"P99延迟(ms)" FROM( SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN  
1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟  
(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " GROUP BY  
request_uri )ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )  
AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )  
AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入  
流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量  
(KB)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT  
( remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS  
"访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS  
"入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != " GROUP BY addr  
having length(upstream_addr) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS  
"访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS  
"平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90  
延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟  
(ms)" FROM (SELECT upstream_addr as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400 THEN  
1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延迟  
(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟  
(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != "  
GROUP BY addr having length(upstream_addr) > 2) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图表所关联的查询分析语句如下所示：

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS  
"访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS  
"平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90  
延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟  
(ms)" FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400  
THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延  
迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",  
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != " GROUP BY  
addr having length(upstream_addr) > 2)ORDER BY "访问失败率(%)" desc
```

----结束

7.3.4.18 VPC 仪表盘模板

虚拟私有云（Virtual Private Cloud，VPC）是隔离的、私密的虚拟网络环境。用户可以自由配置VPC内的IP地址段、子网、安全组等子服务，也可以申请弹性带宽和弹性公网IP搭建业务系统。VPC日志流中记录了虚拟私有云中的流量信息，可以帮助您检查和优化安全组和网络ACL控制规则、监控网络流量、进行网络攻击分析等。

云日志服务支持日志采集向导一站式采集VPC日志，并为VPC日志配置结构化和仪表盘。该仪表盘主要展示VPC日志的Action总次数，ACCEPT总字节数、ACCEPT总包数、REJECT总字节数、REJECT总包数、源地址的Action次数分布、总分钟Action次数、Action分布、流日志记录状态分布、Action次数的源地址运行商分布、Top5字节数的源地址、Top5字节数的目标地址、Top5包数的目标端口、各协议的每分钟包数、弹性网卡。

前提条件

- 已采集VPC日志，详情请参见[虚拟私有云VPC接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 VPC 流日志

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“日志管理”。

步骤3 在“日志应用”模块中，单击“VPC日志流中心”，选择“进入仪表盘”。

步骤4 在仪表盘模板下方，选择“VPC仪表盘模板 > VPC流日志”，查看图表详情。

- Action总次数**图表所关联的查询分析语句如下所示：

```
select CASE WHEN total_actions < 1000 THEN concat(cast( total_actions AS VARCHAR), '次') WHEN
total_actions < 1000 * 1000 THEN concat(cast(round(total_actions / 1000.0, 2) AS VARCHAR), '千次')
WHEN total_actions < 1000000000 THEN concat(cast(round(total_actions / 1000000.0, 2) AS
VARCHAR), '百万次') WHEN total_actions / 1000.0 < 1000000000 THEN
concat(cast(round(total_actions / 1000 / 1000000.0, 1) AS VARCHAR), '十亿次') ELSE
concat(cast(round(total_actions / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR), '万亿次') END AS
"total_actions" from (select count(1) as total_actions where log_status='OK' and version=1)
```

- ACCEPT总字节数**图表所关联的查询分析语句如下所示：

```
select CASE WHEN accept_bytes < 1024 THEN concat(cast( accept_bytes AS VARCHAR), 'B') WHEN
accept_bytes < 1024 * 1024 THEN concat(cast(round(accept_bytes / 1024, 2) AS VARCHAR), 'KB')
WHEN accept_bytes < 1000000000 THEN concat(cast(round(accept_bytes / 1024.0 / 1024, 2) AS
VARCHAR), 'MB') WHEN accept_bytes / 1000.0 < 1000000000 THEN concat(cast(round(accept_bytes /
1024 / 1000000.0, 2) AS VARCHAR), 'GB') ELSE concat(cast(round(accept_bytes / 1000.0 / 1000 /
1000 / 1000, 1) AS VARCHAR), 'TB') END AS "accept_bytes" from (select sum(bytes) as accept_bytes
where log_status='OK' and version=1 and action='ACCEPT')
```

- ACCEPT总包数**图表所关联的查询分析语句如下所示：

```
select CASE WHEN accept_packets < 1024 THEN concat(cast( accept_packets AS VARCHAR), 'B')
WHEN accept_packets < 1024 * 1024 THEN concat(cast(round(accept_packets / 1024, 2) AS
VARCHAR), 'KB') WHEN accept_packets < 1000000000 THEN concat(cast(round(accept_packets /
1024.0 / 1024, 2) AS VARCHAR), 'MB') WHEN accept_packets / 1000.0 < 1000000000 THEN
concat(cast(round(accept_packets / 1024 / 1000000.0, 2) AS VARCHAR), 'GB') ELSE
concat(cast(round(accept_packets / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR), 'TB') END AS
"accept_packets" from (select sum(packets) as accept_packets where log_status='OK' and version=1
and action='ACCEPT')
```

- REJECT总字节数**图表所关联的查询分析语句如下所示：

```
select CASE WHEN reject_bytes < 1024 THEN concat(cast( reject_bytes AS VARCHAR), 'B') WHEN
reject_bytes < 1024 * 1024 THEN concat(cast(round(reject_bytes / 1024, 2) AS VARCHAR), 'KB') WHEN
reject_bytes < 1000000000 THEN concat(cast(round(reject_bytes / 1024.0 / 1024, 2) AS
VARCHAR), 'MB') WHEN reject_bytes / 1000.0 < 1000000000 THEN concat(cast(round(reject_bytes /
1024 / 1000000.0, 2) AS VARCHAR), 'GB') ELSE concat(cast(round(reject_bytes / 1000.0 / 1000 /
1000 / 1000, 1) AS VARCHAR), 'TB') END AS "reject_bytes" from (select sum(bytes) as reject_bytes
where log_status='OK' and version=1 and action='REJECT')
```

- REJECT总包数**图表所关联的查询分析语句如下所示：

```
select CASE WHEN reject_packets < 1024 THEN concat(cast( reject_packets AS VARCHAR), 'B') WHEN
reject_packets < 1024 * 1024 THEN concat(cast(round(reject_packets / 1024, 2) AS VARCHAR), 'KB')
WHEN reject_packets < 1000000000 THEN concat(cast(round(reject_packets / 1024.0 / 1024, 2) AS
VARCHAR), 'MB') WHEN reject_packets / 1000.0 < 1000000000 THEN
concat(cast(round(reject_packets / 1024 / 1000000.0, 2) AS VARCHAR), 'GB') ELSE
concat(cast(round(reject_packets / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR), 'TB') END AS
"reject_packets" from (select sum(packets) as reject_packets where log_status='OK' and version=1
and action='REJECT')
```

- 源地址的Action次数分布**图表所关联的查询分析语句如下所示：

```
select IP_TO_PROVINCE(srcaddr) as province, count(1) as total_actions where IP_TO_COUNTRY
(srcaddr) = '中国' group by province HAVING province not in ('','保留地址','')
```

- 每分钟Action次数**图表所关联的查询分析语句如下所示：

```
select TIME_FORMAT(date_trunc('minute', MILLIS_TO_TIMESTAMP("start" * 1000)), 'MM-dd HH:mm')
as "t", "action", count(1) as "total_actions" where log_status='OK' and version=1 group by "t",
"action" order by t asc limit 1000
```

- **Action分布图表**所关联的查询分析语句如下所示：

```
select action, count(1) as total_actions where log_status='OK' and version=1 group by action
```
- **流日志记录状态分布图表**所关联的查询分析语句如下所示：

```
select log_status, count(1) as total_actions where version=1 group by log_status
```
- **Action次数的源地地址运营商分布图表**所关联的查询分析语句如下所示：

```
select ip_to_provider(srcaddr) as src_addr_provider, count(1) as total_actions where log_status='OK'
and version=1 group by src_addr_provider order by total_actions desc limit 5
```
- **Top5字节数的源地地址图表**所关联的查询分析语句如下所示：

```
select ip_to_provider(srcaddr) as src_addr_provider, count(1) as total_actions where log_status='OK'
and version=1 group by src_addr_provider order by total_actions desc limit 5
```
- **Top5字节数的目标地址图表**所关联的查询分析语句如下所示：

```
select dstaddr, sum(bytes) as total_bytes where log_status='OK' and version=1 group by dstaddr order
by total_bytes desc limit 5
```
- **Top5包数的目标端口图表**所关联的查询分析语句如下所示：

```
select dstport, sum(packets) as total_packets where log_status='OK' and version=1 group by dstport
order by total_packets desc limit 5
```
- **各协议的每分钟包数图表**所关联的查询分析语句如下所示：

```
select TIME_FORMAT(date_trunc('minute', MILLIS_TO_TIMESTAMP("start" * 1000)), 'MM-dd HH:mm')
as t, protocol, sum(packets) as total_packets where log_status='OK' and version=1 group by t,
protocol order by t asc limit 1000
```
- **弹性网卡图表**所关联的查询分析语句如下所示：

```
select interface_id as "ID", sum(packets) as '数据包总数量', sum(bytes) as '数据包总大小' where
log_status='OK' and version=1 group by "ID"
```

----结束

7.3.4.19 WAF 仪表盘模板

Web应用防火墙（Web Application Firewall，WAF），通过对HTTP(S)请求进行检测，识别并阻断SQL注入、跨站脚本攻击、网页木马上传、命令/代码注入、文件包含、敏感文件访问、第三方应用漏洞攻击、CC攻击、恶意爬虫扫描、跨站请求伪造等攻击，保护Web服务安全稳定。

使用[日志搜索与分析（即将下线SQL分析方式）](#)时，对应的WAF仪表盘模板支持[查看WAF安全日志中心](#)和[查看WAF访问日志中心](#)。

前提条件

- 已采集WAF日志，详情请参考[Web应用防火墙WAF接入LTS](#)。
- 日志配置结构化，详情请参见[设置云端结构化解析日志](#)。

查看 WAF 安全日志中心

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“WAF仪表盘模板 > WAF安全日志中心”，查看图表详情。

- **被攻击网站图表**所关联的查询分析语句如下所示：

```
SELECT diff [ 1 ] AS "VALUE", COALESCE ( diff [ 1 ] - diff [ 2 ], 0 ) AS "BEFORE" FROM
(
  SELECT
    compare ( "DATA", 86400 ) AS diff
```

```
FROM
( SELECT count( DISTINCT "host" ) AS "DATA" FROM log
WHERE action != "
))
```

- **攻击来源国家**图表所关联的查询分析语句如下所示：

```
SELECT
diff [ 1 ] AS
"VALUE"
,
COALESCE ( diff [ 1 ]- diff [ 2 ], 0 ) AS "BEFORE"
FROM
(
SELECT
compare ( "DATA", 86400 ) AS diff
FROM
( SELECT count( DISTINCT ip_to_country ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END) )
AS "DATA" FROM log
WHERE action != "
)
)
```

- **Web攻击拦截**图表所关联的查询分析语句如下所示：

```
SELECT
CASE
WHEN
diff [ 1 ] < 1000 THEN
concat( cast( diff [ 1 ] AS VARCHAR ), ' 次' )
WHEN diff [ 1 ] < 1000 * 1000 THEN
concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千次' )
WHEN diff [ 1 ] < 1000000000 THEN
concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万次' )
WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000000, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' )
END AS
"value"
,
CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio
FROM
( SELECT compare ( "data", 86400 ) AS diff FROM ( SELECT count( 1 ) AS "data" FROM log
WHERE action = " ) )
```

- **CC攻击拦截**图表所关联的查询分析语句如下所示：

```
SELECT
CASE
WHEN
diff [ 1 ] < 1000 THEN
concat( cast( diff [ 1 ] AS VARCHAR ), ' 次' )
WHEN diff [ 1 ] < 1000 * 1000 THEN
concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千次' )
WHEN diff [ 1 ] < 1000000000 THEN
concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万次' )
WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000000, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' )
END AS
"value"
,
CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio
FROM
( SELECT compare ( "data", 86400 ) AS diff FROM ( SELECT count( 1 ) AS "data" FROM log
WHERE attack != 'default' ) )
```

- **攻击者UV**图表所关联的查询分析语句如下所示：

```
SELECT
CASE
WHEN
diff [ 1 ] < 1000 THEN
```



```
concat( cast( cast ( diff [ 1 ] AS INTEGER ) AS VARCHAR ), ' 个' )
WHEN diff [ 1 ] < 1000 * 1000 THEN
concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千个' )
WHEN diff [ 1 ] < 1000000000 THEN
concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万个' )
WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000000, 1 ) AS VARCHAR ), ' 十亿' ) ELSE
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿' )
END AS "value",
CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio
FROM
(
SELECT
compare ( "data", 86400 ) AS diff
FROM
( SELECT count( DISTINCT CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) AS "data"
FROM log
))
```

- **攻击拦截**图表所关联的查询分析语句如下所示:

```
SELECT
CASE
WHEN
diff [ 1 ] < 1000 THEN
concat( cast( diff [ 1 ] AS VARCHAR ), ' 次' )
WHEN diff [ 1 ] < 1000 * 1000 THEN
concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千次' )
WHEN diff [ 1 ] < 1000000000 THEN
concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万次' )
WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000000, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' )
END AS
"value",
CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS "ratio"
FROM
(
SELECT
compare ( "data", 86400 ) AS diff
FROM
( SELECT count( 1 ) AS "data" FROM log WHERE action != "" )
)
```

- **CC攻击**图表所关联的查询分析语句如下所示:

```
SELECT
ip_to_province (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS province,
count( 1 ) AS "攻击次数"
WHERE attack != 'default' and ip_to_country(CASE WHEN sip = '-' THEN remote_ip ELSE sip END)
= '中国'
GROUP BY
province
```

- **攻击类型分布**图表所关联的查询分析语句如下所示:

```
SELECT time_format( MILLIS_TO_TIMESTAMP( TIMESTAMP_TO_MILLIS(__time) -
MOD(TIMESTAMP_TO_MILLIS(__time), 3600)), 'HH:mm' ) AS dt, count( 1 ) AS cnt, CASE WHEN
action = 'block' THEN '拦截' WHEN action = 'log' THEN '仅记录' WHEN action = 'captcha' THEN '人机
验证' END AS attack FROM log WHERE action != "" GROUP BY TIMESTAMP_TO_MILLIS(__time) -
MOD(TIMESTAMP_TO_MILLIS(__time), 3600), attack ORDER BY cnt DESC
```

- **Web攻击**图表所关联的查询分析语句如下所示:

```
SELECT
ip_to_province (
CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS province,
count( 1 ) AS "攻击次数"
WHERE action = 'block' and ip_to_country(CASE WHEN sip = '-' THEN remote_ip ELSE sip END) =
'中国'
GROUP BY
province
```

- **CC攻击(世界)**图表所关联的查询分析语句如下所示:

```
SELECT
  ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS country,
  count( 1 ) AS "攻击次数"
WHERE attack != 'default'
GROUP BY
  country
```

- **Web攻击(世界)**图表所关联的查询分析语句如下所示：

```
SELECT
  ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS country,
  count( 1 ) AS "攻击次数"
WHERE action = 'block'
GROUP BY
  country
```

----结束

查看 WAF 访问日志中心

步骤1 在仪表盘模板下方，选择“WAF仪表盘模板 > WAF访问日志中心”，查看图表详情。

- **PV**图表所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 1000 THEN concat( cast( diff [ 1 ] AS
VARCHAR ), ' 次' ) WHEN diff [ 1 ] < 1000 * 1000 THEN concat( cast( round( diff [ 1 ] / 1000,
1 ) AS VARCHAR ), ' 千次' ) WHEN diff [ 1 ] < 1000000000 THEN concat( cast( round( diff
[ 1 ] / 1000000.0, 1 ) AS VARCHAR ), ' 百万次' ) WHEN diff [ 1 ] / 1000.0 < 1000000000 THEN
concat( cast( round( diff [ 1 ] / 1000 / 1000000.0, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast( round( diff [ 1 ] / 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' ) END
AS "VALUE" , CASE WHEN diff [ 2 ] = 0 THEN 0 ELSE round( diff [ 3 ] - 1, 2 ) END AS
ratio FROM ( SELECT compare ( DATA, 86400 ) AS diff FROM ( SELECT
count( 1 ) AS DATA FROM log ) )
```

- **UV**图表所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 1000 THEN concat( cast( diff [ 1 ] AS
VARCHAR ), ' 次' ) WHEN diff [ 1 ] < 1000 * 1000 THEN concat( cast( round( diff [ 1 ] / 1000,
1 ) AS VARCHAR ), ' 千次' ) WHEN diff [ 1 ] < 1000000000 THEN concat( cast( round( diff
[ 1 ] / 1000000.0, 1 ) AS VARCHAR ), ' 百万次' ) WHEN diff [ 1 ] / 1000.0 < 1000000000 THEN
concat( cast( round( diff [ 1 ] / 1000 / 1000000.0, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast( round( diff [ 1 ] / 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' ) END
AS "VALUE" , CASE WHEN diff [ 2 ] = 0 THEN 0 ELSE round( diff [ 3 ] - 1,
2 ) END AS ratio FROM ( SELECT compare ( DATA, 86400 ) AS diff
FROM ( SELECT count( DISTINCT CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) AS
"DATA" FROM log ) )
```

- **流入流量**图表所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 102 THEN concat( cast( diff [ 1 ] AS
VARCHAR ), ' B' ) WHEN diff [ 1 ] < 1024 * 1024 THEN concat( cast( round( diff [ 1 ] / 1024,
1 ) AS VARCHAR ), ' KB' ) WHEN diff [ 1 ] < 1024 * 1024 * 1024 THEN
concat( cast( round( diff [ 1 ] / 1024.0 / 1024, 1 ) AS VARCHAR ), ' MB' ) WHEN diff [ 1 ] / 1024.0
< 1024 * 1024 * 1024 THEN concat( cast( round( diff [ 1 ] / 1024.0 / 1024 / 1024, 1 ) AS
VARCHAR ), ' GB' ) ELSE concat( cast( round( diff [ 1 ] / 1024.0 / 1024 / 1024 / 1024, 1 ) AS
VARCHAR ), ' TB' ) END AS "VALUE" , CASE WHEN diff [ 2 ] = 0 THEN
0 ELSE round( diff [ 3 ] - 1, 2 ) END AS ratio FROM ( SELECT compare ( "DATA",
86400 ) AS diff FROM ( SELECT COALESCE ( sum( request_length ), 0 ) AS "DATA" FROM
log ) )
```

- **网络in带宽峰值**图表所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 102 THEN concat( cast( round( diff [ 1 ], 2 ) AS
VARCHAR ), ' B/s' ) WHEN diff [ 1 ] < 1024 * 1024 THEN concat( cast( round( diff [ 1 ] /
1024, 1 ) AS VARCHAR ), ' KB/s' ) WHEN diff [ 1 ] < 1024 * 1024 * 1024 THEN
concat( cast( round( diff [ 1 ] / 1024.0 / 1024, 1 ) AS VARCHAR ), ' MB/s' ) WHEN diff [ 1 ] /
1024.0 < 1024 * 1024 * 1024 THEN concat( cast( round( diff [ 1 ] / 1024.0 / 1024 / 1024, 1 ) AS
VARCHAR ), ' GB/s' ) ELSE concat( cast( round( diff [ 1 ] / 1024.0 / 1024 / 1024 / 1024, 1 ) AS
VARCHAR ), ' TB/s' ) END AS "VALUE" , CASE WHEN diff [ 2 ] = 0
THEN 0 ELSE round( diff [ 3 ] - 1, 2 ) END AS ratio FROM ( SELECT compare
( "DATA", 86400 ) AS diff FROM ( SELECT COALESCE ( max( "DATA" ), 0 ) AS
"DATA" FROM ( SELECT TIME_FLOOR( _time, 'PT1M' ) AS dt, sum( request_length ) / 60.0 AS
"DATA" FROM log GROUP BY dt LIMIT 10000 ) )
```

- **网络out带宽峰值**图表所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 102 THEN concat( cast( round( diff [ 1 ], 2 ) AS VARCHAR ), ' B/s' )
WHEN diff [ 1 ] < 1024 * 1024 THEN concat( cast( round( diff [ 1 ]/ 1024, 1 ) AS VARCHAR ), ' KB/s' )
WHEN diff [ 1 ] < 1024 * 1024 * 1024 THEN concat( cast( round( diff [ 1 ]/ 1024.0 / 1024, 1 ) AS
VARCHAR ), ' MB/s' ) WHEN diff [ 1 ]/ 1024.0 < 1024 * 1024 * 1024 THEN concat( cast( round( diff
[ 1 ]/ 1024.0 / 1024 / 1024, 1 ) AS VARCHAR ), ' GB/s' ) ELSE concat( cast( round( diff [ 1 ]/ 1024.0 /
1024 / 1024 / 1024, 1 ) AS VARCHAR ), ' TB/s' ) END AS "value", case when diff [ 2 ] = 0 then 0 else
round( diff [ 3 ] - 1, 2 ) END AS "ratio" FROM ( SELECT compare ( "DATA", 86400 ) AS diff FROM
( SELECT COALESCE ( max( bytes_out ), 0 ) AS "DATA" FROM ( SELECT time_ceil( __time,'PT1M' ) AS
dt, sum( body_bytes_sent )/ 60.0 AS bytes_out FROM log GROUP BY dt LIMIT 10000 )))
```

- **流量带宽趋势**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( MILLIS_TO_TIMESTAMP( TIMESTAMP_TO_MILLIS(__time) -
MOD(TIMESTAMP_TO_MILLIS(__time), 600000)), 'HH:mm' ) AS dt,
round( sum( request_length )/ 1024.0 / 600, 2 ) AS "流入流量(KB/s)",
round( sum( body_bytes_sent )/ 1024.0 / 600, 2 ) AS "流出流量(KB/s)" where request_length is not
null GROUP BY TIMESTAMP_TO_MILLIS(__time) - MOD(TIMESTAMP_TO_MILLIS(__time),
600000) ORDER BY dt LIMIT 1000
```

- **PV/UV趋势**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT(MILLIS_TO_TIMESTAMP( TIMESTAMP_TO_MILLIS(__time) -
MOD(TIMESTAMP_TO_MILLIS(__time), 3600000)), 'HH:mm' ) AS dt, count( 1 ) AS PV,
APPROX_COUNT_DISTINCT( CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) AS UV
FROM log GROUP BY TIMESTAMP_TO_MILLIS(__time) -
MOD(TIMESTAMP_TO_MILLIS(__time), 3600000) ORDER BY dt LIMIT 1000
```

- **访问状态分布**图表所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT(MILLIS_TO_TIMESTAMP( TIMESTAMP_TO_MILLIS(__time) -
MOD(TIMESTAMP_TO_MILLIS(__time), 3600000)), 'HH:mm' ) AS dt, count( 1 ) AS cnt,
concat( cast( "response_code" / 100 AS VARCHAR ), 'XX' ) AS "status" GROUP BY
TIMESTAMP_TO_MILLIS(__time) - MOD(TIMESTAMP_TO_MILLIS(__time), 3600000),
"response_code" / 100 ORDER BY dt DESC LIMIT 10000
```

- **访问来源**图表所关联的查询分析语句如下所示：

```
SELECT ip_to_province (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS country,
count( 1 ) AS "访问次数" where ip_to_country(CASE WHEN sip = '-' THEN remote_ip ELSE sip END)
= '中国' GROUP BY country
```

- **流入流量来源（中国）**图表所关联的查询分析语句如下所示：

```
SELECT ip_to_province (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS region,
round( sum( request_length )/ 1024.0 / 1024, 4 ) AS "流入流量(MB)" where ip_to_country(CASE
WHEN sip = '-' THEN remote_ip ELSE sip END) = '中国' GROUP BY region
```

- **流入流量来源（世界）**图表所关联的查询分析语句如下所示：

```
SELECT ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS region,
round( sum( request_length )/ 1024.0 / 1024, 4 ) AS "流入流量(MB)" where request_length is not
null GROUP BY region
```

- **来源网络提供商**图表所关联的查询分析语句如下所示：

```
SELECT ip_to_provider (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS provider,
round( sum( request_length )/ 1024.0 / 1024.0, 3 ) AS mb_in GROUP BY provider
HAVING ip_to_provider (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) != '*' ORDER
BY mb_in DESC LIMIT 10
```

- **访问域名**图表所关联的查询分析语句如下所示：

```
SELECT http_host, count( 1 ) AS "被访问次数" GROUP BY http_host ORDER
BY "被访问次数" DESC LIMIT 30
```

- **响应最慢的URL**图表所关联的查询分析语句如下所示：

```
SELECT http_host AS "网站",url_extract_path (COALESCE ( url, '/' )) AS URL,sum( request_time )/
count( 1 ) AS "响应时间(毫秒)",count( 1 ) AS "访问次数" GROUP BY http_host, url ORDER BY "响应时
间(毫秒)" DESC LIMIT 100
```

- **访问最多的客户端**图表所关联的查询分析语句如下所示：

```
SELECT ip AS "客户端", client AS "地理网络", concat( cast( (CASE WHEN pv IS NULL THEN 0 ELSE
pv END) AS VARCHAR ), ' (', cast( case when head_pv = 'null' then 0 else (case when head_pv > 0
then head_pv else 0 end) end AS VARCHAR ), '/', cast( case when get_pv = 'null' then 0 else (case
when get_pv > 0 then get_pv else 0 end) end AS VARCHAR ), '/', cast( case when put_pv = 'null'
then 0 else (case when put_pv > 0 then put_pv else 0 end) end AS VARCHAR ), '/', cast( case when
post_pv = 'null' then 0 else (case when post_pv > 0 then post_pv else 0 end) end AS VARCHAR ), '/',
cast( case when delete_pv = 'null' then 0 else (case when delete_pv > 0 then delete_pv else 0 end)
end AS VARCHAR ), '/', ')' ) AS "PV (Head, Get, Put, Post, Delete方法)", error_count AS "错误访问次
```

```
数" FROM ( SELECT ip, client, sum( CASE WHEN "method" = 'PUT' AND "status" < 400 THEN pv  
ELSE 0 END ) AS put_pv, sum( CASE WHEN "method" = 'GET' AND "status" < 400 THEN pv ELSE 0  
END ) AS get_pv, sum( CASE WHEN "method" = 'POST' AND "status" < 400 THEN pv ELSE 0 END )  
AS post_pv, sum( CASE WHEN "method" = 'DELETE' AND "status" < 400 THEN pv ELSE 0 END ) AS  
delete_pv, sum( CASE WHEN "method" = 'HEAD' AND "status" < 400 THEN pv ELSE 0 END ) AS  
head_pv, sum( throughput ) AS throughput, sum( pv ) AS pv, sum( CASE WHEN "status" < 400  
THEN 1 ELSE 0 END ) AS error_count FROM ( SELECT CASE WHEN sip = '-' THEN remote_ip  
ELSE sip END AS ip, "method", CASE WHEN ip_to_country ( CASE WHEN sip = '-' THEN  
remote_ip ELSE sip END ) = '上海' THEN '中国上海' WHEN ip_to_province ( CASE WHEN sip = '-'  
THEN remote_ip ELSE sip END ) = '*' THEN '未知IP' WHEN ip_to_provider ( CASE WHEN sip = '-'  
THEN remote_ip ELSE sip END ) = '内网IP' THEN '内网IP' ELSE concat( ip_to_country ( CASE WHEN  
sip = '-' THEN remote_ip ELSE sip END ), '/', ip_to_province ( CASE WHEN sip = '-' THEN remote_ip  
ELSE sip END ), '/', CASE WHEN ip_to_city ( CASE WHEN sip = '-' THEN remote_ip ELSE sip  
END ) = '*' THEN '' ELSE ip_to_city ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) END, ' ',  
ip_to_provider ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END )) END AS client, sum( CASE  
WHEN "response_code" < 400 THEN 1 ELSE 0 END ) AS pv, round( sum( request_length ) / 1024.0 /  
1024, 1 ) AS throughput, "response_code" AS "status" FROM log GROUP BY ip, client,  
"method", "response_code" ORDER BY pv DESC, client, "method" LIMIT 1000 ) GROUP BY ip,  
client ORDER BY pv DESC ) LIMIT 100
```

---结束

7.3.4.20 采集诊断仪表盘模板

采集诊断仪表盘模板支持[查看ICAgent采集监控](#)、[查看ICAgent整体状态](#)和[查看ICAgent异常监控](#)。

前提条件

在LTS控制台配置中心页面的“ICAgent采集开关”页签，开启ICAgent诊断开关，请参考[设置ICAgent日志采集开关](#)。

查看 ICAgent 采集监控

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“采集诊断仪表盘模板 > ICAgent采集监控”，查看图表详情。

- 过滤日志组ID，所关联的查询分析语句如下所示：

```
select loggroup from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm' group  
by loggroup
```

- 过滤日志流ID，所关联的查询分析语句如下所示：

```
select logstream from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm'  
group by logstream
```

- 原始数据流量图表所关联的查询分析语句如下所示：

```
SELECT  
case  
when diff [ 1 ] is null then '0'  
when diff [ 1 ] > 1024 and diff [ 1 ] <= 1024*1024 then concat(round(diff [ 1 ]*1.0/1024,4),' KB')  
when diff [ 1 ] > 1024*1024 and diff [ 1 ] < 1024*1024*1024 then concat(round(diff  
[ 1 ]*1.0/1024/1024,4),' MB')  
when diff [ 1 ] > 1024*1024*1024 and diff [ 1 ] < 1024*1024*1024*1024 then concat(round(diff  
[ 1 ]*1.0/1024/1024/1024,4),' GB')  
when diff [ 1 ] > 1024*1024*1024*1024 then concat(round(diff [ 1 ]*1.0/1024,4),' TB')  
else concat(round(diff [ 1 ]*1.0,2),' B')  
END AS "原始流量",  
case  
when diff [ 3 ] is null then '昨日无数据'  
else round(diff [ 3 ] - 1,2)  
END AS "同比昨日"
```

```
FROM  
(  
  SELECT  
    report_topic,  
    compare ( traffic, 86400 ) AS diff  
  FROM  
    ( SELECT report_topic, sum( read_bytes ) AS traffic FROM log WHERE report_topic = 'icagent_profile'  
    GROUP BY report_topic )  
  GROUP BY  
    report_topic)
```

- **采集文件数**图表所关联的查询分析语句如下所示：

```
select diff[1] as "采集文件数", case when diff[3] is not null then round(diff[3] -1 ,2) else '昨日无数据'  
end as "同比昨天" from (select compare(uv,86400) as diff from (select report_topic,count(distinct  
concat(file_name,host_ip)) as uv from log where report_topic = 'icagent_profile' group by  
report_topic) group by report_topic)
```

- **采集机器数/同比昨天**图表所关联的查询分析语句如下所示

```
select diff[1] as "采集机器数", case when diff[3] is not null then round(diff[3] -1 ,2) else '昨日无数据'  
end as "同比昨天" from (select compare(uv,86400) as diff from (select report_topic,count(distinct  
host_ip) as uv from log where report_topic = 'icagent_profile' group by report_topic) group by  
report_topic)
```

- **数据发送流量**图表所关联的查询分析语句如下所示：

```
SELECT  
  "time",  
  case  
    when traffic is null then 0  
    else round(traffic*1.0/1024/1024,2)  
  END AS "发送流量 MB"  
from  
(SELECT  
  time_format( time_floor ( __time, 'PT5M' ), 'yyyy-MM-dd HH:mm' ) AS "time",  
  sum( read_bytes ) AS "traffic"  
FROM  
  log  
WHERE  
  report_topic = 'icagent_profile'  
GROUP BY  
  "time")
```

- **ICAgent写入次数**图表所关联的查询分析语句如下所示：

```
select time_floor(__time,'PT5M') as \"time\",sum(read_count) as \"写入次数\" where report_topic =  
'icagent_profile' group by \"time\"
```

- **采集机器数**图表所关联的查询分析语句如下所示：

```
select time_floor(__time,'PT5M') as "time", count(distinct host_ip) as "采集机器数" where report_topic  
= 'icagent_profile' group by "time"
```

- **采集文件分布**图表所关联的查询分析语句如下所示：

```
SELECT  
  file_name AS "采集路径",  
  host_ip AS "IP",  
  case  
    when traffic is null then '0'  
    when traffic > 1024 and traffic <= 1024*1024 then concat(round(traffic*1.0/1024,2),' KB')  
    when traffic > 1024*1024 and traffic < 1024*1024*1024 then concat(round(traffic*1.0/1024,2),' MB')  
    when traffic > 1024*1024*1024 and traffic < 1024*1024*1024*1024 then  
concat(round(traffic*1.0/1024,2),' GB')  
    when traffic > 1024*1024*1024*1024 then concat(round(traffic*1.0/1024,2),' TB')  
    else concat(round(traffic*1.0,2),' B')  
  END AS "采集流量"  
FROM  
(SELECT  
  file_name,  
  host_ip,  
  sum( read_bytes ) AS "traffic"  
WHERE  
  "report_topic" = 'icagent_profile'  
GROUP BY
```

```
file_name,  
host_ip)
```

----结束

查看 ICAgent 整体状态

步骤1 在仪表盘模板下方，选择“采集诊断仪表盘模板 > ICAgent整体状态”，查看图表详情。

- **活跃ICAgent数**图表所关联的查询分析语句如下所示：

```
select diff[1] as "活跃ICAgent数",case when diff[2] is not null then diff[2] else '昨日无数据' end as "昨日活跃ICAgent数" from (select report_topic,compare(uv,86400) as diff from (SELECT report_topic,COUNT(DISTINCT ip) as uv FROM log where report_topic = 'icagent_status' group by report_topic) group by report_topic)
```
- **发送延迟/次数趋势**图表所关联的查询分析语句如下所示：

```
select time_floor(__time,'PT5M') as \"time\", sum(\"metric.lts_cost.below_100_ms\") as \"below_100_ms\", sum(\"metric.lts_cost.100to500ms\") as \"100to500ms\", sum(\"metric.lts_cost.500msto1s\") as \"500msto1s\", sum(\"metric.lts_cost.1sto10s\") as \"1sto10s\", sum(\"metric.lts_cost.10ston\") as \"10ston\" from log where \"report_topic\" = 'icagent_status' group by \"time\"
```
- **运行状态分布**图表所关联的查询分析语句如下所示：

```
select status,count(DISTINCT ip) as pv from log where report_topic = 'icagent_status' group by status
```
- **CPU趋势**图表所关联的查询分析语句如下所示：

```
select ip,time_floor(__time,'PT5M') as \"time\",avg(\"metric.cpu_usage\") as \"CPU占用率\" from log where report_topic = 'icagent_status' and \"metric.cpu_usage\" is not null group by \"time\",ip order by \"time\"
```
- **ICAgent整体状态**图表所关联的查询分析语句如下所示：

```
select host_name as "主机名",ip as "IP", version as "版本号", os as "操作系统", time_format(MILLIS_TO_TIMESTAMP(ANY_VALUE("metric.start_time")), 'yyyy/MM/dd HH:mm:ss ZZZ') as "启动时间",avg("metric.cpu_usage") as "CPU",avg("metric.mem_used")*1.0 as "内存(MB)",status as "运行状态" where report_topic = 'icagent_status' group by host_name,ip,version,os,status
```

----结束

查看 ICAgent 异常监控

步骤1 在仪表盘模板下方，选择“采集诊断仪表盘模板 > ICAgent异常监控”，查看图表详情。

- **过滤日志组ID**，所关联的查询分析语句如下所示：

```
select loggroup from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm' group by loggroup limit 10000
```
- **过滤日志流ID**，所关联的查询分析语句如下所示：

```
select logstream from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm' group by logstream limit 10000
```
- **关键错误数**图表所关联的查询分析语句如下所示：

```
select diff[1] as "错误数", case when diff[3] is not null then round(diff[3] - 1, 2) else '昨日无数据' end as "错误数对比昨天" from (select report_topic,compare(pv, 86400) as diff from (select report_topic,count(1) as pv from log where report_topic = 'icagent_alarm' group by report_topic) group by report_topic)
```
- **丢弃超大行**图表所关联的查询分析语句如下所示：

```
select diff[1] as "丢弃行数", case when diff[3] is not null then round(diff[3] - 1, 2) else '昨日无数据' end as "丢弃行数对比昨天" from (select report_topic,compare(pv, 400) as diff from (select report_topic,count(1) as pv from log where report_topic = 'icagent_alarm' and alarm_type = 'DISCARD_BIG_LINE' group by report_topic) group by report_topic)
```
- **请求LTS失败**图表所关联的查询分析语句如下所示：

```
select diff[1] as "请求失败数", case when diff[3] is not null then round(diff[3] - 1, 2) else '昨日无数据' end as "请求失败数对比昨天" from (select report_topic,compare(pv, 86400) as diff from (select
```

```
report_topic,count(1) as pv from log where report_topic = 'icagent_alarm' and alarm_type = 'HTTP_REQUEST_ALARM' group by report_topic) group by report_topic)
```

- **文件超过上限问题数**图表所关联的查询分析语句如下所示:

```
select diff[1] as "文件超过上限问题数",case when diff[3] is not null then round(diff[3] - 1, 2) else '昨日无数据' end as "文件超过上限问题数对比昨天" from (select report_topic,compare(pv, 86400) as diff from (select report_topic,count(1) as pv from log where report_topic = 'icagent_alarm' and alarm_type = 'COLLECT_FILE_EXCEED' group by report_topic) group by report_topic)
```

- **关键错误数(必需处理)**图表所关联的查询分析语句如下所示:

```
select TIME_FORMAT(MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)), 'yyyy/MM/dd HH:mm:ss ZZ') as "最近发生时间", loggroup as "日志组ID", logstream as "日志流ID", host_ip as "IP", alarm_type as "告警类型", os as "系统类型", alarm_message as "告警详情" where report_topic = 'icagent_alarm' group by loggroup,logstream,host_ip,alarm_type,os,alarm_message limit 10000
```

- **丢弃超大行详情**图表所关联的查询分析语句如下所示:

```
select TIME_FORMAT(MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)), 'yyyy/MM/dd HH:mm:ss ZZ') as "最近发生时间", loggroup as "日志组ID", logstream as "日志流ID", host_ip as "IP", alarm_type as "告警类型", os as "系统类型", alarm_message as "告警详情" where report_topic = 'icagent_alarm' and alarm_type = 'DISCARD_BIG_LINE' group by loggroup,logstream,host_ip,alarm_type,os,alarm_message limit 10000
```

- **请求LTS失败详情**图表所关联的查询分析语句如下所示:

```
select TIME_FORMAT(MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)), 'yyyy/MM/dd HH:mm:ss ZZ') as "最近发生时间", loggroup as "日志组ID", logstream as "日志流ID", host_ip as "IP", alarm_type as "告警类型", os as "系统类型", alarm_message as "告警详情" where report_topic = 'icagent_alarm' and alarm_type = 'HTTP_REQUEST_ALARM' group by loggroup,logstream,host_ip,alarm_type,os,alarm_message limit 10000
```

- **文件超过上限问题数详情**图表所关联的查询分析语句如下所示:

```
select TIME_FORMAT(MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)), 'yyyy/MM/dd HH:mm:ss ZZ') as "最近发生时间", loggroup as "日志组ID", logstream as "日志流ID", host_ip as "IP", alarm_type as "告警类型", os as "系统类型", alarm_message as "告警详情" where report_topic = 'icagent_alarm' and alarm_type = 'COLLECT_FILE_EXCEED' group by loggroup,logstream,host_ip,alarm_type,os,alarm_message limit 10000
```

----结束

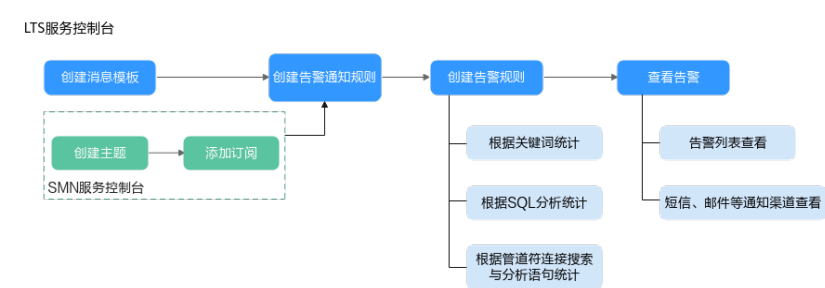
8 日志告警

8.1 日志告警概述

云日志服务支持创建搜索分析类型、关键词统计类型、SQL统计类型的日志告警规则，根据设置的告警规则触发告警，可以在告警列表查看上报的告警详情，实时监控日志数据。或者通过告警通知规则将上报告警以短信，邮件等多种形式发送告警通知，方便用户及时处理告警问题。具体的配置请参考图8-1。

了解更多日志告警的概念信息，请参考[基本概念](#)。

图 8-1 日志告警操作流程



1. **消息模板**是告警通知消息的固定格式，系统发送告警通知消息必须使用消息模板向订阅者发送。支持内置消息模板，不同协议的订阅者优先选择模板名称对应的协议模板。如果对应的协议模板不存在，则可以创建消息模板，更多信息请参考[在LTS页面创建消息模板](#)。
2. **消息通知服务SMN主题**是消息发布或客户端订阅通知的特定事件类型，更多信息请参考[创建主题](#)。
3. **告警通知规则**用于关联消息通知服务SMN主题和消息模板，也是告警规则中的一个参数。当触发告警后，自动根据消息模板内容以短信，邮件等多种形式发送告警通知。更多信息请参考[创建告警通知规则](#)。
4. **告警规则**用于设置触发告警的配置规则，例如查询条件、检测规则、统计周期、通知频率和通知渠道等信息。更多信息请参考[配置日志告警规则](#)。

5. 当设置的匹配条数或条件表达式满足时，告警将会上报，用户可以在LTS控制台[查看告警](#)，更多信息请参考[查看LTS告警列表](#)，也可以根据告警通知规则的配置在短信、邮箱等通知渠道查看告警。

使用限制

使用限制请参考[日志告警](#)。

8.2 配置日志告警规则

LTS支持对日志流中的日志数据设置告警规则，实时监控服务运行状态。目前每个账号可以创建告警规则的总数不超过200个。

支持批量创建多个告警规则，请参考[配置多个告警规则](#)。

前提条件

- 已创建日志组、日志流，请参考[管理日志组](#)和[管理日志流](#)。
- 日志接入LTS后，如果需要使用字段索引进行搜索分析统计，请确保索引配置准确，详情请参考[创建LTS日志索引](#)。准确的索引配置可以帮助您更高效地查询和分析日志数据，方便您基于特定字段（如日志级别、错误码、响应时间等）配置日志告警规则。

配置告警规则

支持以下统计类型的告警规则：搜索分析（管道符方式）、关键词统计及SQL统计。

配置单个搜索分析告警（管道符方式）

云日志服务支持将日志数据进行结构化，通过新SQL引擎配置告警规则，支持使用管道符（搜索|分析），定时查询结构化数据，当且仅当条件表达式返回为true的时候，将告警进行上报，用户可以在LTS控制台查看SQL告警。

说明

管道符搜索方式仅支持白名单用户使用，如有需要，请提工单申请开通。详细操作请参考[提交工单](#)。

- 步骤1** 在云日志服务管理控制台，单击“日志告警”。
- 步骤2** 单击“告警规则”。
- 步骤3** 在“告警规则”页签，单击“创建”，在界面右侧弹出“新建告警规则”页面。
- 步骤4** 在“新建告警规则”页面，参考[表8-1](#)配置告警规则相关参数。

表 8-1 搜索分析告警参数说明

参数类别	参数名称	参数说明
基本信息	规则名称	告警规则的名称，建议您根据业务自定义告警规则名称。创建成功后将鼠标悬浮在规则名称上，显示规则名称和原始名称。支持修改规则名称，不支持修改首次创建的原始名称。 命名规范如下： 只支持输入英文、数字、中文、中划线、下划线，且不能中划线、下划线开头或结尾。长度为1-128个字符。
	企业项目	选择业务需要的企业项目，默认为default。也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。 - 企业项目需要开通后才能使用，请参考如何开通企业项目。 - 支持将该企业项目资源迁出，详细请参考迁出企业项目资源。
	描述	对该规则进行简要描述，长度不能超过128个字符。
统计分析	统计类型	勾选搜索分析：使用新SQL引擎配置告警，支持使用管道符（搜索 分析）。 告警规则创建成功后无法修改统计类型，请根据业务需求规划好统计类型。
	查询条件（支持添加3条查询语句。）	日志组名称：选择已创建的日志组。
		日志流名称：选择已创建的日志流。
		查询时间：指定语句的查询周期。查询语句的时间范围：从当前时间往前推一个周期。例如：查询时间设置为1小时，当前时间为9:00，则查询语句的时间范围为8:00-9:00。 - 如果查询时间单位为分钟，则取值范围是1-60； - 如果查询时间单位为小时，则取值范围是1-24。
		查询语句：语句格式如下。 搜索语句 SQL分析语句 LTS会根据设置的语句对日志流中的日志进行监控。输入查询语句后，单击“预览”，在预览下方正常显示查询结果，根据预览结果的字段设置检测规则。

参数类别	参数名称	参数说明
		添加多个查询条件：重复执行以上步骤即可，多个查询条件的编号请参考图8-2查看。 图 8-2 查看查询条件编号  其中，第一个查询条件编号为0，第二个查询条件编号为1，第三个查询条件编号为2。

参数类别	参数名称	参数说明
	校验规则	当满足条件表达式（ ）时，告警级别为（ 紧急、重要、次要、提示 ）And（ ）次统计周期内，至少满足（ ）次以上条件触发告警。 输入具体的条件表达式，当条件表达式返回为true的时候，产生告警，否则不产生告警。 ● 条件表达式支持中文。条件表达式不支持纯数字，不支持以数字开头的。 ● 触发告警级别包括“紧急”、“重要”、“次要”、“提示”，默认“紧急”。 ● 统计周期次数指上面设置的统计周期；满足条件次数指设置的条件表达式。配置的统计周期次数须大于等于满足触发条件次数。统计周期次数最小值为1，最大值为10。 单击+增加条件表达式（ or ），最多支持增加20条。 单击  删除条件表达式。 条件表达式支持的基础语法和多表组合语法。关联多个查询条件时，条件表达式中需要使用如下格式来引用字段： <code>#{查询条件编号}.{查询语句字段}</code> 例如，\$0.pv 表示第一个图表0的查询语句 “* select count(*) as pv” 中的pv字段。 注意 使用多个图表时，条件表达式中必须存在第一个图表0，即\$0。 ● 基础语法： - 基础运算符：支持加（ + ）、减（ - ）、乘（ * ）、除（ / ）、取模运算（ % ）。示例：x * 10 + y > 100。 - 比较运算符：支持大于（ > ）、大于等于（ >= ）、小于（ < ）、小于等于（ <= ）、等于（ == ）、不等于（ != ）。示例：x >= 100。 - 逻辑运算符：支持与（ && ）、或（ ）。示例：x > 0 && y < 200。 - 取反前缀：支持取反前缀（ ! ）。示例：!(x < 1 && x > 100)。 - 数值常量：支持数值常量，并作为64位浮点数处理。示例：x > 10。 - 字符串常量：支持字符串常量（ "字符串" ），例如 "string"。示例：str == "string"。 - 布尔常量：支持布尔常量(true、false)。示例：(x < 100)!=true。 - 括号：支持使用括号改变计算的优先级。示例：x *(y + 10)<200。

参数类别	参数名称	参数说明
		- contains函数：支持使用contains函数判断是否包含子串，例如contains(str, "hello")返回true则表示str中包含hello子串。 • 多表组合语法： - 基础运算符：（+*/%）。 - 比较运算符：大于（>）、大于等于（>=）、小于（<）、小于等于（<=）、等于（==）、不等于（!=）。 - 逻辑运算符：与（&&）、或（ ）。 - 取反前缀（!）。 - contains函数。 - 括号（）。 示例： \$0.pv > 10 && \$1.uv < 2 \$0.pv 表示第一个图表0的查询语句“* select count(*) as pv”中的pv字段，\$1.uv表示第二个图表1的查询语句“* select count(*) as uv”的uv字段，这两个条件需要同时满足。
高级设置	统计周期	条件表达式查询的频率可以设置为： • 每小时：表示整点小时查询。 • 每天：需要指定几点整查询。 • 每周：需要指定周几的几点整查询。 • 固定间隔：自定义间隔周期，需要指定1-60分钟/1-24小时。例如：当前时间为9:00，固定间隔设置为5分钟，则第一次查询时间为9:00，第二次查询时间为9:05，第三次查询时间为9:10.....当查询时间大于1小时，固定间隔时间最小取值为5分钟。 • CRON表达式：按照CRON表达式执行查询任务，CRON表达式最小精度为分钟，格式为24小时制。
	恢复策略	配置恢复策略，即满足该策略时，会发送告警恢复通知。 配置的最近统计周期次数内，如果不满足触发条件且开启恢复时通知开关，则会发送恢复告警通知。 最近统计周期次数最小值为1，最大值为10。
	通知场景	• 告警触发时：用于发送触发告警通知。开启该按钮，当满足触发条件时，会发送告警通知；未开启该按钮，当满足触发条件时，不会发送告警通知。 • 告警恢复时：用于发送恢复告警通知。开启该按钮，当满足恢复策略时，会发送恢复告警通知；未开启该按钮，当满足恢复策略时，不会发送恢复告警通知。

参数类别	参数名称	参数说明
	通知频率	支持选择立即通知、每5分钟、每10分钟、每15分钟、每30分钟、每1小时、每3小时、每6小时发送告警。 立即通知指只要产生告警就发送通知，每10分钟指的是两次通知之间最小时间间隔为10分钟，可避免告警轰炸。
	告警通知规则	请从下拉列表中选择已创建的告警通知规则。 若没有，请单击右侧“创建告警通知规则”，详细操作请见 创建告警通知规则 。
	语言	发送告警的语言，支持中文（简体）和英文。
	标签	按照业务需求对不同的告警规则添加对应的标签。单击“添加标签”，分别填写标签键key和标签值value。 如需添加多个标签可重复该步骤，最多支持添加20个标签。 标签键key限制条件： • 标签键可以包含任意语种的字母、数字和空格，以及_.:+=-@字符，但首尾不能包含空格，且不能以_sys_开头。 • 标签键长度不能超过128个字符；标签值长度不能超过255个字符。 • 标签键名称不可重复。 标签值value限制条件： • 标签值可以包含任意语种的字母、数字和空格，以及_.:+=-@字符。 • 标签值长度不能超过255个字符。 删除标签： 警告 删除标签无法恢复，请谨慎操作。 如需删除标签可单击标签操作列的“删除”。

步骤5 单击“确定”，搜索分析告警规则创建成功。

告警规则创建完成后，告警状态默认显示“已开启”。关闭告警规则后，告警状态显示“已关闭”，临时关闭告警后，告警状态显示“临时关闭到2023/05/30 16:21:24.000 GMT+08:00”。（临时关闭的时间仅供参考，请以设置临时关闭告警的时间为准）

当开启告警规则且关联日志流满足告警规则时，会触发告警；当关闭告警规则时，即使有满足该告警规则的情况，也不会触发告警。

----结束

配置关键词告警规则

LTS支持对日志流中的日志数据进行关键词统计，通过设置告警规则，监控日志中的关键词，统计一定时间段内的日志中关键字出现的次数，实时监控服务运行状态。

- 步骤1** 登录[云日志服务控制台](#)。
- 步骤2** 左侧导航选择“日志告警”。
- 步骤3** 单击“告警规则”。
- 步骤4** 在“告警规则”页签，单击“创建”，在界面右侧弹出“新建告警规则”页面。
- 步骤5** 在“新建告警规则”页面，配置告警规则相关参数。

表 8-2 关键词告警参数说明

参数类别	参数名称	参数说明
基本信息	规则名称	告警规则的名称，建议您根据业务自定义告警规则名称。创建成功后将鼠标悬浮在规则名称上，显示规则名称和原始名称。支持修改规则名称，不支持修改首次创建的原始名称。 命名规范如下： 只支持输入英文、数字、中文、中划线、下划线，且不能以中划线、下划线开头或结尾。长度为1-128个字符。
基本信息	企业项目	选择业务需要的企业项目，默认为default。也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。 - 企业项目需要开通后才能使用，请参考如何开通企业项目。 - 支持将该企业项目资源迁出，详细请参考迁出企业项目资源。
基本信息	描述	对该规则进行简要描述。长度不能超过128个字符。
统计分析	统计类型	勾选 关键词统计 ：适用于使用关键词搜索配置日志告警的场景。 告警规则创建成功后无法修改统计类型，请根据业务需求规划好统计类型。
	查询条件	日志组名称 ：选择已创建的日志组。
		日志流名称 ：选择已创建的日志流。 当日志组下有多个日志流时，支持选择多个日志流，即可批量创建关键词告警。
		查询时间 ：指定语句的查询周期。查询语句的时间范围：从当前时间往前推一个周期。例如：查询时间设置为1小时，当前时间为9:00，则查询语句的时间范围为8:00-9:00。 - 如果查询时间单位为分钟，则取值范围是1-60； - 如果查询时间单位为小时，则取值范围是1-24。

参数类别	参数名称	参数说明
		关键词：根据业务需要，选取在日志搜索能查到的日志关键字，LTS会根据设置的关键词对日志流中的日志进行监控。 关键词支持精确匹配和模糊匹配，输入长度不超过1024个字符。如何设置关键词搜索请参考搜索语法介绍。 在配置日志字段索引时，默认关闭“大小写敏感”功能，此时设置关键字不区分大小写。若启用“大小写敏感”功能，则需确保告警关键字的大小写匹配准确。详细请参考创建LTS日志索引。
	检测规则	配置触发条件，即满足该条件时，会触发告警。 ● 匹配条数：当关键词搜索结果的日志条数达到设定的条数时，会触发告警。支持大于（>）、大于等于（>=）、小于（<）、小于等于（<=）4种比较运算符。 ● 触发告警级别包括“紧急”、“重要”、“次要”、“提示”，默认“紧急”。 ● 统计周期次数指高级设置的统计周期；满足条件次数是指设置的关键词。配置的统计周期次数须大于等于满足触发条件次数。统计周期次数最小值为1，最大值为10。 单击+增加条件表达式（or），最多支持增加20条。 单击  删除条件表达式。
高级设置	统计周期	条件表达式查询的频率可以设置为： ● 每小时：表示整点小时查询。 ● 每天：需要指定几点整查询。 ● 每周：需要指定周几的几点整查询。 ● 固定间隔：自定义间隔周期，需要指定1-60分钟/1-24小时。例如：当前时间为9:00，固定间隔设置为5分钟，则第一次查询时间为9:00，第二次查询时间为9:05，第三次查询时间为9:10.....当查询时间大于1小时，固定间隔时间最小取值为5分钟。 ● CRON表达式：按照CRON表达式执行查询任务，CRON表达式最小精度为分钟，格式为24小时制。
高级设置	恢复策略	配置恢复策略，即满足该策略时，会发送告警恢复通知。 配置的最近统计周期次数内，如果不满足触发条件且开启恢复时通知开关，则会发送恢复告警通知。 最近统计周期次数最小值为1，最大值为10。

参数类别	参数名称	参数说明
高级设置	通知场景	● 告警触发时：用于发送触发告警通知。开启该按钮，当满足触发条件时，会发送告警通知；未开启该按钮，当满足触发条件时，不会发送告警通知。 ● 告警恢复时：用于发送恢复告警通知。开启该按钮，当满足恢复策略时，会发送恢复告警通知；未开启该按钮，当满足恢复策略时，不会发送恢复告警通知。
高级设置	通知频率	支持选择立即通知、每5分钟、每10分钟、每15分钟、每30分钟、每1小时、每3小时、每6小时发送告警。 立即通知指只要产生告警就发送通知，每10分钟指的是两次通知之间最小时间间隔为10分钟，可避免告警轰炸。
高级设置	告警通知规则	请从下拉列表中选择已创建的告警通知规则。 若没有，请单击右侧“创建告警通知规则”。
高级设置	语言	发送告警的语言。
高级设置	标签	按照业务需求对不同的告警规则添加对应的标签。单击“添加”，分别填写标签键key和标签值value。 如需添加多个标签可重复该步骤，最多支持添加20个标签。 标签键key限制条件： ● 标签键可以包含任意语种的字母、数字和空格，以及_!:=+-@字符，但首尾不能包含空格，且不能以_sys_开头。 ● 标签键长度不能超过128个字符。 ● 标签键名称不可重复。 标签值value限制条件： ● 标签值可以包含任意语种的字母、数字和空格，以及_!:=+-@字符。 ● 标签值长度不能超过255个字符。 删除标签： 警告 删除标签无法恢复，请谨慎操作。 如需删除标签可单击标签操作列的“删除”。

步骤6 单击“确定”，关键词告警规则创建成功。详细示例请查看[参考示例：出现关键字即触发告警](#)。

告警规则创建完成后，告警状态默认显示“已开启”。关闭告警规则后，告警状态显示“已关闭”，临时关闭告警后，告警状态显示“临时关闭到2023/05/30 16:21:24.000 GMT+08:00”。（临时关闭的时间仅供参考，请以设置临时关闭告警的时间为准）

当开启告警规则且关联日志流满足告警规则时，会触发告警；当关闭告警规则时，即使有满足该告警规则的情况，也不会触发告警。

----结束

配置 SQL 告警规则

云日志服务支持将日志数据进行结构化，通过配置SQL告警规则，定时查询结构化数据，当且仅当条件表达式返回为true的时候，将告警进行上报，用户可以在LTS控制台查看SQL告警。每条SQL告警规则可以关联1到3个图表，每个图表包含一条查询某个日志流的SQL查询语句。

旧版SQL引擎即将下线，推荐您使用新版SQL引擎（搜索分析），功能更丰富。新版SQL引擎的语法兼容SQL92标准，如果查询失败，您需要手动修改SQL语句。

此功能支持全部用户使用的局点有：华南-广州、华北-北京四、华北-乌兰察布一、华东-上海一、中国-香港、西南-贵阳一、亚太-新加坡、华北-北京一，支持部分白名单用户使用的局点有：华东-上海二、亚太-曼谷、华南-深圳、中东-利雅得、亚太-雅加达，其他局点暂不支持该功能。

- 步骤1 在云日志服务管理控制台，单击“日志告警”。
- 步骤2 单击“告警规则”。
- 步骤3 在“告警规则”页签，单击“创建”，在界面右侧弹出“新建告警规则”页面。
- 步骤4 在“新建告警规则”页面，参考表8-3配置告警规则相关参数。

表 8-3 SQL 告警参数说明

参数类别	参数名称	参数说明
基本信息	规则名称	告警规则的名称，建议您根据业务自定义告警规则名称。创建成功后将鼠标悬浮在规则名称上，显示规则名称和原始名称。支持修改规则名称，不支持修改首次创建的原始名称。 命名规范如下： 只支持输入英文、数字、中文、中划线、下划线，且不能以中划线、下划线开头或结尾。长度为1-128个字符。
基本信息	企业项目	选择业务需要的企业项目，默认为default。也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。 - 企业项目需要开通后才能使用，请参考如何开通企业项目。 - 支持将该企业项目资源迁出，详细请参考迁出企业项目资源。
基本信息	描述	对该规则进行简要描述。长度不能超过128个字符。
统计分析	统计类型	勾选SQL统计：使用SQL分析配置告警。 告警规则创建成功后无法修改统计类型，请根据业务需求规划好统计类型。

参数类别	参数名称	参数说明
	相关图表	配置告警关联的图表。 当前有两种添加方式：直接添加和从图表导入。若所选日志流未配置结构化规则，请先设置云端结构化解析日志。最多支持添加3个图表，图表不能为空，图表中的sql查询语句也不能为空。 ● 直接添加 单击“直接添加”，具体的参数配置信息如下： - 日志组名称：在下拉框中选择有日志上报的日志组。 - 日志流名称：在下拉框中选择目标日志组下的日志流。 - 查询时间：当前所选日志的查询时间，查询时间（1 ~ 60分钟/1 ~ 24小时），单位为分钟或小时。 - 查询语句：填写可视化查询语句。具体请参考SQL分析语法介绍。 ● 从图表导入 单击“从图表导入”，进入“添加可视化图表”页面，选择对应日志组、日志流下的可视化图表，单击“确定”。 若该日志流下没有图表或没有所需的图表，单击界面上的“前往添加图表”，进入可视化界面，设置完成后单击“保存并返回”返回到告警规则界面，自动打开创建规则弹框，填充新创建的图表及图表的查询语句。 可以指定图表的查询时间（1 ~ 60分钟/1 ~ 24小时），单位为分钟或小时，每个图表最多可以查询最近一天的数据，当统计周期选择1~4分钟时，图表查询时间不能超过1小时。 - 单击 ：跳转到日志流的可视化查看详情界面。 - 单击 ：删除该直接添加的图表。 - 单击“预览”：查看可视化分析后的数据。必须要执行“预览”，否则将无法保存该告警规则。 若想添加多个图表，重复执行以上步骤即可。多个图表的编号请参考图8-3查看。 其中，第一个图表编号为0，第二个图表编号为1，第三个图表编号为2。

参数类别	参数名称	参数说明
		图 8-3 查看图表编号 配置项 日志组名称 日志流名称 日志组 日志流 查询时间 5 分钟 查询范围 SELECT 配置项 日志组名称 日志流名称 日志组 日志流 查询时间 5 分钟 查询范围 SELECT 配置项 日志组名称 日志流名称 日志组 日志流 查询时间 5 分钟 查询范围 SELECT

参数类别	参数名称	参数说明
	检测规则	输入具体的条件表达式，当条件表达式返回为true的时候，产生告警，否则不产生告警。 ● 条件表达式支持中文。不支持纯数字，不支持以数字开头的。 ● 触发告警级别包括“紧急”、“重要”、“次要”、“提示”，默认“紧急”。 ● 统计周期次数指上面设置的统计周期；满足条件次数指设置的条件表达式。配置的统计周期次数须大于等于满足触发条件次数。统计周期次数最小值为1，最大值为10。 单击+增加条件表达式（or），最多支持增加20条。 单击删除条件表达式。 条件表达式支持的基础语法和多表组合语法。关联多个图表时，条件表达式中需要使用如下格式来引用字段： <code>\$(图表编号).{查询语句字段}</code> 例如，\$0.pv 表示第一个图表0的查询语句“select count(*) as pv”中的pv字段。 注意 使用多个图表时，条件表达式里必须存在第一个图表0，即\$0。 ● 基础语法： - 基础运算符：支持加（+）、减（-）、乘（*）、除（/）、取模运算（%）。示例：x * 10 + y > 100。 - 比较运算符：支持大于（>）、大于等于（>=）、小于（<）、小于等于（<=）、等于（==）、不等于（!=）。示例：x >= 100。 - 逻辑运算符：支持与（&&）、或（ ）。示例：x > 0 && y < 200。 - 取反前缀：支持取反前缀（!）。示例：!(x < 1 && x > 100)。 - 数值常量：支持数值常量，并作为64位浮点数处理。示例：x > 10。 - 字符串常量：支持字符串常量（"字符串"），例如"string"。示例：str == "string"。 - 布尔常量：支持布尔常量(true、false)。示例：(x < 100)!=true。 - 括号：支持使用括号改变计算的优先级。示例：x *(y + 10)<200。 - contains函数：支持使用contains函数判断是否包含子串，例如contains(str, "hello")返回true则表示str中包含hello子串。 ● 多表组合语法： - 基础运算符：（+ - * / %）。

参数类别	参数名称	参数说明
		- 比较运算符：大于（>）、大于等于（>=）、小于（<）、小于等于（<=）、等于（==）、不等于（!=）。 - 逻辑运算符：与（&&）、或（ ）。 - 取反前缀（!）。 - contains函数。 - 括号（ ）。 示例： \$0.pv > 10 && \$1.uv < 2 \$0.pv 表示第一个图表0的查询语句“select count(*) as pv”中的pv字段，\$1.uv表示第二个图表1的查询语句“select count(*) as uv”的uv字段，这两个条件需要同时满足。
高级设置	统计周期	条件表达式查询的频率可以设置为： ● 每小时：表示整点小时查询。 ● 每天：需要指定几点整查询。 ● 每周：需要指定周几的几点整查询。 ● 固定间隔：自定义间隔周期，需要指定1-60分钟/1-24小时。例如：当前时间为9:00，固定间隔设置为5分钟，则第一次查询时间为9:00，第二次查询时间为9:05，第三次查询时间为9:10..... 当查询时间大于1小时，固定间隔时间最小取值为5分钟。 ● CRON表达式：按照CRON表达式执行查询任务，CRON表达式最小精度为分钟，格式为24小时制。
高级设置	恢复策略	配置恢复策略，即满足该策略时，会发送告警恢复通知。 配置的最近统计周期次数内，如果不满足触发条件且开启恢复时通知开关，则会发送恢复告警通知。 最近统计周期次数最小值为1，最大值为10。
高级设置	通知场景	● 告警触发时：用于发送触发告警通知。开启该按钮，当满足触发条件时，会发送告警通知；未开启该按钮，当满足触发条件时，不会发送告警通知。 ● 告警恢复时：用于发送恢复告警通知。开启该按钮，当满足恢复策略时，会发送恢复告警通知；未开启该按钮，当满足恢复策略时，不会发送恢复告警通知。
高级设置	通知频率	支持选择立即通知、每5分钟、每10分钟、每15分钟、每30分钟、每1小时、每3小时、每6小时发送告警。 立即通知指只要产生告警就发送通知，每10分钟指的是两次通知之间最小时间间隔为10分钟，可避免告警轰炸。

参数类别	参数名称	参数说明
高级设置	告警通知规则	请从下拉列表中选择已创建的告警通知规则。 若没有，请单击右侧“创建告警通知规则”，详细操作请见 创建告警通知规则 。
高级设置	语言	发送告警的语言，支持中文（简体）和英文。
高级设置	标签	按照业务需求对不同的告警规则添加对应的标签。单击“添加标签”，分别填写标签键key和标签值value。如需添加多个标签可重复该步骤，最多支持添加20个标签。 标签键key限制条件： • 标签键可以包含任意语种的字母、数字和空格，以及_!:=+-@字符，但首尾不能包含空格，且不能以_sys_开头。 • 标签键长度不能超过128个字符。 • 标签键名称不可重复。 标签值value限制条件： • 标签值可以包含任意语种的字母、数字和空格，以及_!:=+-@字符。 • 标签值长度不能超过255个字符。 删除标签： 警告 删除标签无法恢复，请谨慎操作。 如需删除标签可单击标签操作列的“删除”。

步骤5 单击“确定”，SQL告警规则创建成功。详细示例请查看[参考示例：根据关键字出现的次数设置告警](#)。

----结束

配置多个告警规则

支持批量创建多个告警规则。

步骤1 在“告警规则”页面，批量导入告警规则。

1. 单击“导入/导出 > 导入”，进入导入告警规则页面。
2. 下载告警模板到本地填写完成。
3. 单击“选择文件”，选择本地填写好的文件。
4. 确认导入的规则信息无误后，单击“导入”。
5. 导入成功后，在规则列表下方显示告警规则明细。

步骤2 单击“批量创建/编辑”，进入批量创建/编辑告警规则页面。

步骤3 在规则列表下方，输入告警规则数量，单击“添加”。

- 在规则列表下方默认已有1个告警规则，最多支持再添加199个数量，因此支持同时添加200个告警规则。

- 在右侧配置项下方自定义填写“规则名称”，配置项设置完成后，也可以双击左侧告警规则配置的名称自定义命名。规则名称只支持输入英文、数字、中文、中划线、下划线，且不能以中划线、下划线开头或结尾。长度为1-128个字符。
- 鼠标放在告警规则配置上面，单击即可直接复制告警规则配置。
- 鼠标放在告警规则配置上面，单击，在弹出的提示框单击“确定”删除告警规则配置。
- 若规则原始名称已存在，提交后会编辑该条规则，否则会创建一条新规则，不能修改统计类型，否则会导致修改失败。
- 单击“导入”，在导入告警规则页面，提前下载告警模板，本地填写完成后，即可选择文件，单击“导入”完成批量导入告警规则。

步骤4 在配置项下方，设置告警规则。

步骤5 设置完成后，单击“参数检查”。

步骤6 检查成功后，单击“提交”，批量告警规则设置完成。

- 一个告警规则设置完成后，单击“应用于其他告警规则”即可将该告警规则复制到其他告警规则。
- 例如添加了4个告警规则，批量创建成功后，在告警规则页签下方，就会显示4条告警规则。

----结束

告警规则后续操作

创建告警规则后，支持对告警规则进行修改、开启/关闭、复制、删除等操作，可能会导致原有告警规则发生变化，请谨慎操作。

- 支持对单个告警规则进行如下操作：
修改告警规则：单击目标告警规则操作列的“修改”，在“修改告警规则”页面，修改规则名称、查询条件、检测规则等信息，修改完成后，单击“确定”。
临时关闭告警规则：单击目标告警规则操作列的“更多 > 临时关闭”，设置临时关闭的截止时间，单击“确定”。
复制告警规则：单击目标告警规则操作列的“更多 > 复制”，即可直接复制告警规则。
删除告警规则：单击目标告警规则操作列的“删除”，在弹出的对话框中，单击“确定”删除该告警规则。

说明

删除告警规则后不可恢复，请谨慎操作。

- 告警恢复通知状态：开启该按钮，当满足恢复策略时，会发送恢复告警通知；未开启该按钮，当满足恢复策略时，不会发送恢复告警通知。
- 告警规则开关状态：开启该按钮，该告警规则才会生效；未开启该按钮，告警规则失效。
- 标签管理：单击目标告警规则操作列的“更多 > 标签管理”，在打开的“编辑标签”页面添加标签。
- 勾选多个告警规则后，支持对多个告警进行批量操作：开启告警规则、关闭告警规则、临时关闭告警规则、取消临时关闭、开启告警恢复通知、关闭告警恢复通知、删除告警规则、导入告警规则、导出已选中数据到XLSX、导出全部数据到XLSX。

- 鼠标悬浮在规则名称上，显示规则名称和原始名称。不支持修改首次创建的原始名称。

CRON 表达式

CRON表达式的格式组成为：分钟（0-59）、小时（0-23）、日（1-31）、月（1-12）、星期（0-6，0=SUN，6=SAT），这几个字段使用空格顺序连接并且均为必填字段。除了基础取值，在实际使用场景还可以结合一些特殊字符来输出更复杂时间规则，示例如下：

- 从00:00开始，每隔整10分钟查询一次，分别为10分钟、20分钟、30分钟、40分钟、50分钟、60分钟。*表示该字段的任意值，/表示指定时间的间隔。

```
0/10 * * * *
```

例如，当前时间为16:37，下一次查询时间为16:50。

- 从00:00开始，每隔5小时查询一次，分别为0时、5时、10时、15时、20时。*表示该字段的任意值，/表示指定时间的间隔。

```
0 0/5 * * * *
```

例如，当前时间为16:37，下一次查询时间为20:00。

- 每天14:00查询一次。

```
0 14 * * * *
```

- 每月10日00:00查询一次。

```
0 0 10 * * *
```

参考示例：出现关键字即触发告警

如果您希望日志中出现目标关键字时，就能触发告警，则您可以参考本示例设置查询语句和关键词告警规则。以下示例仅供参考，请以实际业务为准。

设置的关键字一定是日志流存在的关键字，例如Error关键字。

图 8-4 查询结果



- 查询语句：选择查询时间范围为15分钟，然后执行如下语句，查询包含Error关键字的日志。更多搜索语法请参考[搜索语法介绍](#)。

图 8-5 查询语句

统计分析

统计类型

搜索分析

使用新SQL引擎配置告警，支持使用管道符（搜索分析）

关键词统计

使用关键词搜索配置告警

SQL统计

使用SQL分析配置告警

查询条件

日志组名称

Its-group

日志流名称

Its-

查询时间

15

分钟

关键词

Error

- 告警通知：创建上述告警规则后，只要日志中出现Error关键字，您就可以在“告警列表”收到告警通知。您还可以单击告警名称，查看告警详情，进行溯源。

图 8-6 告警通知

出现Error就报错

告警级别

紧急

发生时间

2024/08/23 11:36:09.911 GMT+08:00

持续时长

1分钟48秒

告警详情

规则名称

出现Error就报错

规则原始名称

出现Error就报错

条件表达式

匹配条数 > 1

当前值

{ "Error": 90 }

统计周期

每5分钟

告警状态

触发

日志组名称

lt-

日志流名称

lt-

关键词

Error

查询时间

15分钟

告警对象

修复建议

名称

出现Error就报错

资源类型

日志组/流

日志组/流

lt-

参考示例：根据关键字出现的次数设置告警

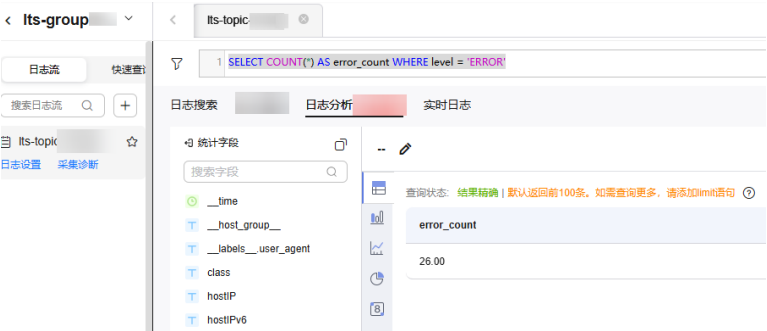
如果您希望在一定时间范围内日志关键字出现的次数达到指定次数时，才触发告警，则您可以参考本示例设置查询分析语句和SQL告警规则。以下示例仅供参考，请以实际业务为准。

SQL语句：

```
SELECT COUNT(*) AS error_count WHERE level = 'ERROR'
```

在目标日志流可查询到当前日志等级为“ERROR”的日志总计出现26次。

图 8-7 查询结果



- 查询语句：选择查询时间范围为5分钟，然后执行如下语句“SELECT count(*) as Error”，统计5分钟内出现Error关键字的次数。更多搜索语法请参考[SQL分析语法介绍](#)。

图 8-8 查询语句

统计分析

统计类型

- ☐ 搜索分析
使用新SQL引擎配置告警，支持使用通符(搜索)分析
- ☐ 关键词统计
使用关键词搜索配置告警
- ☒ SQL统计
使用SQL分析配置告警

相关图表

日志组名称: its-group

日志流名称: its-topic

查询时间: 5 分钟

查询语句: SELECT COUNT(*) AS error_count WHERE level = 'ERROR'

+ 直接添加 + 从图表导入

检测规则

当满足条件表达式 error_count > 10 时，告警级别为 紧急

1 次统计周期内，至少满足 1 次以上条件触发告警

- 告警通知：创建上述告警规则后，只要日志中出现Error关键字超过2次，您就可以在“告警列表”收到告警通知。您还可以单击告警名称，查看告警详情，进行溯源。

图 8-9 告警通知

5分钟产生10条日志级别为ERROR告警		
告警级别	紧急	
发生时间	2025/07/08 19:35:01.715 GMT+08:00	
持续时长	1分钟04秒	
告警详情	规则名称	5分钟产生10条日志级别为ERROR告警
	规则原始名称	5分钟产生10条日志级别为ERROR告警
	条件表达式	error_count > 10
	当前值	{ "error_count": 15 }
	统计周期	每分钟
	告警状态	触发
	日志组/流名称	lts-group- / lts-topic-
	查询语句	SELECT COUNT(*) AS error_count WHERE level = 'ERROR'
	查询时间	5分钟
告警对象	修复建议	
名称	5分钟产生10条日志级别为ERROR告警	
资源类型	日志组/流	
日志组/流	lts-group- / lts-topic-	

参考示例：使用搜索 | 分析语句设置告警

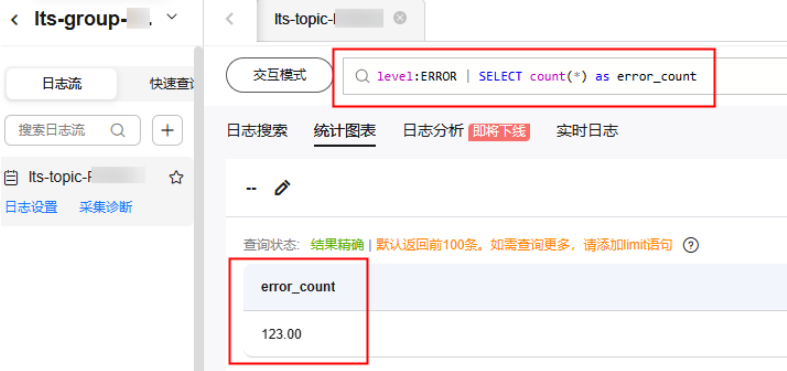
如果您希望查询一定时间范围内日志级别为“ERROR”的日志数据并统计日志级别为“ERROR”的日志条数，到达指定条数后触发告警，则您可以参考本示例设置查询分析语句和告警规则。以下示例仅供参考，请以实际业务为准。

查询与分析语句：

```
level:ERROR | SELECT count(*) as error_count
```

在目标日志流可查询到当前日志等级为“ERROR”的日志总计出现123次。

图 8-10 查询与分析结果



- 查询语句：选择查询时间范围为5分钟，然后执行查询与分析语句，统计5分钟内日志告警级别为“ERROR”的日志条数。更多管道符语句语法请参考[日志搜索与分析（管道符方式）](#)。

图 8-11 查询语句

统计分析

统计类型

搜索分析

使用新SQL引擎配置告警，支持使用普通符
(搜索)分析

关键词统计

使用关键词搜索配置告警

SQL统计

使用SQL分析配置告警

查询条件

日志组名称

Its-group

日志流名称

Its-topic

查询时间

5

分钟

查询语句

level ERROR | SELECT count(*) as error_count

格式为：搜索语句 | SQL分析语句

检测规则

当满足条件表达式

error_count > 10

时，告警级别为

紧急

1

次统计周期内，至少满足

1

次以上条件触发告警

条件表达式示例：

单个条件示例：pv > 10，pv 表示查询语句“select count(*) as pv”中的pv字段

多个条件示例：\$0.pv > 10 && \$1.uv < 2，\$0.pv 表示条件0“select count(*) as pv”中的pv字段，\$1.uv表示条件1“select count(*) as uv”的uv字段。

了解更多用法

- 告警通知：创建上述告警规则后，只要日志中出现Error关键字超过2次，您就可以在“告警列表”收到告警通知。您还可以单击告警名称，查看告警详情，进行溯源。

图 8-12 告警通知

5分钟产生10条日志告警级别为ERROR告警

告警级别

紧急

发生时间

2025/07/02 21:04:10.833 GMT+08:00

持续时长

1分钟07秒

告警详情

规则名称

5分钟产生10条日志告警级别为ERROR告警

规则原始名称

条件表达式

error_count > 10

当前值

[{"error_count":15}]

统计周期

每分钟

告警状态

触发

日志组/流名称

Its- 6 / Its-

查询语句

level ERROR | SELECT count(*) as error_count

查询时间

5分钟

告警对象

修复建议

名称

5分钟产生10条日志告警级别为ERROR告警

资源类型

日志组/流

日志组/流

Its-group / Its-to

文档版本 01 (2025-09-17)

版权所有 © 华为云计算技术有限公司

750

相关文档

- LTS支持通过调用API接口实现创建关键词告警规则、查询关键词告警规则、删除关键词告警规则等，详细请参考[关键词告警规则](#)。
- LTS支持通过调用API接口实现创建SQL告警规则、查询SQL告警规则、删除SQL告警规则等，详细请参考[SQL告警规则](#)。

8.3 配置日志告警通知规则

8.3.1 在 LTS 页面创建消息模板

消息模板是告警通知消息的固定格式，系统发送告警通知消息必须使用消息模板向订阅者发送。支持内置消息模板，不同协议的订阅者优先选择模板名称对应的协议模板，如果对应的协议模板不存在，则采用自定义的消息模板。使用消息模板发送告警通知消息时，系统会自动将模板变量替换为告警规则中的内容。

创建消息模板

- 步骤1 登录[云日志服务控制台](#)。
- 步骤2 左侧导航选择“日志告警”，进入告警页面，选择“告警通知规则”。
- 消息模板默认有以下内置模板，当您所选择的消息模板中未配置消息内容时，云日志服务默认使用内置模板。
- 关键词模板：关键词告警模板
 - keywords_template：关键词告警英文模板
 - sql模板：sql告警模板
 - sql_template：sql告警英文模板
- 步骤3 在消息模板页签，单击“创建”，在界面右侧弹出的“创建消息模板”页面中，配置消息模板的相关参数。
- 邮件内容支持html标签和消息预览。
 - 企业微信、钉钉、飞书支持markdown语法和消息预览。
 - 针对AOM和LTS，最多可以创建100（包含）条消息模板，如果消息模板数量已达上限100个时，请删除不需要的消息模板后重新创建。

表 8-4 配置消息模板参数

参数名称	说明	校验规则	样例
模板名称	消息模板的名称	输入内容只能是数字、字母、下划线、汉字、中划线，且不能以下划线、中划线等特殊符号开头和结尾。长度不能超过100个字符。	LTS-test

参数名称	说明	校验规则	样例
模板描述	对消息模板的描述	输入内容只能是数字、字母、下划线、汉字，且不能以下划线等特殊符号开头和结尾。长度不能超过1024个字符。	-
消息头语言	系统在发送消息时会默认添加消息头	• 中文（简体） • 英文	• 中文：“尊敬的用户...” • 英文：“Dear User...”
通知方式	消息的通知方式类型	• 邮件 • 短信 • HTTP/HTTPS • 钉钉 • 飞书 • 企业微信 • 语音，此功能仅针对白名单用户提交工单申请使用。详细操作请参考提交工单。 • welink，此功能仅针对白名单用户提交工单申请使用。详细操作请参考提交工单。	-
主题	消息的主题	支持自定义主题名称和使用变量命名主题两种方式。主题名称长度不能超过512个字符。 仅邮件类型支持配置消息主题。	test

参数名称	说明	校验规则	样例
正文	消息的内容	添加变量： - 规则原始名称：\${event_name} - 告警级别：\${event_severity} - 发生时间：\${starts_at} - 发生区域：\${region_name} - 华为云账号：\${domain_name} - 告警源： \$event.metadata.resource_provider - 资源类型： \$event.metadata.resource_type - 资源标识：\${resources} - 告警状态： \$event.annotations.alarm_status - 表达式： \$event.annotations.condition_expression - 当前值： \$event.annotations.current_value - 表达式带值： \$event.annotations.condition_expression_with_value - 统计周期： \$event.annotations.frequency - 规则名称： \$event.annotations.alarm_rule_alias - 通知频率： \$event.annotations.notification_frequency - 日志组原始名称： \$event.annotations.results[0].log_group_name - 日志流原始名称： \$event.annotations.results[0].log_stream_name	 \${event_name} \${event_severity} \${starts_at} \${region_name}

参数名称	说明	校验规则	样例
		- 日志组名称： \$event.annotations.results[0].log_group_name_alias - 日志流名称： \$event.annotations.results[0].log_stream_name_alias - 关键词告警支持的变量 - 查询时间： \$event.annotations.results[0].time - 查询日志：（日志长度最多2KB，超过2KB被截断丢弃） \$event.annotations.results[0].raw_results - 查询URL： \$event.annotations.results[0].url - 日志组/日志流名称： \$event.annotations.results[0].resource_id 只支持添加首次创建的日志组/日志流原始名称，不支持添加修改后的日志组/日志流名称。 - 日志流的企业项目ID： \$event.annotations.results[0].eps_id - 查询自定义字段 \$event.annotations.results[0].fields.xxx xxx表示原始日志的结构化字段和内置字段（hostIP、hostName等），日志字段长度最多1KB，超过1KB被截断丢弃。 - SQL告警支持的变量 - 图表0的日志组/流名称： \$event.annotations.results[0].resource_id 只支持添加首次创建的日志组/日志流原始名称，不支持添加修改后的日志组/日志流名称。 0代表第一个图表，1代表第二个图表，以此类推。	

参数名称	说明	校验规则	样例
		2. 图表0的查询语句： \$event.annotations.results[0].sql 3. 图表0的查询时间： \$event.annotations.results[0].time 4. 图表0的查询URL： \$event.annotations.results[0].url 5. 图表0的查询日志： \$event.annotations.results[0].raw_results 6. 图表0的日志流的企业项目ID： \$event.annotations.results[0].eps_id 复制模板： - keywords_template - sql_template - sql模板 - 关键词模板 - 自定义模板（用户通过添加变量创建的消息模板）	

步骤4 配置完成后，单击“确定”。

在LTS页面创建消息模板时提示“创建消息模板失败，名称不能重复”，可以参考如下步骤进行排查处理。

1. 请检查消息模板名称是否有重复。
若有重复，请重新修改消息模板名称。因为，LTS的消息模板数据源与AOM消息模板数据源为同一个，消息模板名称不能重复，如果在AOM中创建了名称为test的消息模板，在LTS页面不能创建同名的消息模板。
2. 请检查消息模板数量是否超过100个。
针对AOM和LTS，最多可以创建100（包含）条消息模板，当消息模板数量已达上限100个时，请删除不需要的消息模板后重新创建。

----结束

编辑消息模板

步骤1 在消息模板列表中，单击消息模板名称行后的“修改”，根据表8-4进行修改，其中“模板名称”不可修改。内置消息模板不支持编辑。

步骤2 编辑完成后，单击“确定”。

----结束

复制消息模板

步骤1 在消息模板列表中，单击消息模板名称行后的“复制”，在新打开的页面，参考[创建消息模板](#)设置消息模板参数。

步骤2 设置完成后，单击“确定”。

----结束

删除消息模板



删除后无法恢复，请谨慎操作。

步骤1 在消息模板列表中，单击消息模板名称行后的“删除”。内置消息模板不支持删除。

步骤2 在弹出的对话框中，单击“确认”删除该消息模板。

----结束

批量删除消息模板



删除后无法恢复，请谨慎操作。

步骤1 在消息模板列表中，勾选待删除的消息模板，单击列表左上方“批量删除”。

步骤2 在弹出的删除消息模板页面，单击“确定”，删除所勾选的消息模板。

----结束

导出消息模板

步骤1 在消息模板列表中，勾选待导出的消息模板，单击列表左上方“导出”。

步骤2 选择“导出全部数据到XLSX”或“导出已选中数据到XLSX”，导出成功后即可到本地查看消息模板数据。

----结束

消息模板定制

邮件内容支持html标签和消息预览，企业微信、钉钉、飞书支持markdown语法和消息预览。由于每种语法不同，云日志服务支持您在配置消息模板时，根据实际情况定制通知内容信息。

- 邮件
邮件渠道的内容支持HTML标签。例如：
使用
换行。

使用[\\${event_severity} 添加颜色。]($event.annotations.alarm_rule_url)

- 企业微信

企业微信渠道的内容支持Markdown语法。目前支持的元素参考如下：

标题

```
# 一级标题
## 二级标题
### 三级标题
#### 四级标题
##### 五级标题
##### 六级标题
```

文字加粗

```
**告警**
```

链接

```
[查询url](http://example.com)
```

字体颜色

```
<font color="#FF0000">${event_severity}</font>
```

- 钉钉

钉钉渠道的内容支持Markdown语法。目前支持的元素参考如下：

标题

```
# 一级标题
## 二级标题
### 三级标题
#### 四级标题
##### 五级标题
##### 六级标题
```

文字加粗

```
**告警**
```

链接

```
[查询url](http://example.com)
```

字体颜色

```
<font color="#FF0000">${event_severity}</font>
```

- 飞书

飞书渠道的内容支持Markdown语法。目前支持的元素参考如下：

加粗

```
**粗体**
```

超链接

```
<a>https://open.feishu.cn</a>
```

相关文档

LTS支持通过调用API接口实现创建消息模板、查询消息模板、删除消息模板等，详细请参考[消息模板管理](#)。

8.3.2 创建告警通知规则

LTS提供告警通知规则定制功能，通过告警通知规则关联SMN主题与消息模板。当触发告警后，自动根据消息模板内容以短信，邮件等多种形式发送告警通知。

前提条件

- 已创建一个主题。详细操作请参考[创建主题](#)。
- 已设置主题策略。详细操作请参考[设置主题策略](#)。
- 已为主题添加相关的订阅者，即通知的接收人（例如：邮件或短信）。详细操作请参考[订阅主题](#)。

注意事项

您最多可创建1000个告警通知规则，如果告警通知规则数量已达上限1000时，请删除不需要的通知规则。

创建告警通知规则

- 步骤1 登录[云日志服务控制台](#)。
- 步骤2 在左侧导航栏中选择“日志告警”。
- 步骤3 单击“告警通知规则”页签。
- 步骤4 单击“创建”。设置通知规则名称、通知规则配置等信息。

图 8-13 创建告警通知规则

创建告警通知规则 ⓘ

基本信息

• 通知规则名称

输入内容只能是数字、字母、下划线、汉字、中划线、小数点，且不能以下划线中划线小数点等特殊符号开头和结尾。

• 企业项目

default

C

描述

请输入描述

0/1,024

通知规则配置

• 主题

请选择告警主题

C [创建主题](#)

主题是消息发布或客户端订阅通知的特定事件类型。如果没有可选主题，请到SMN界面创建主题，并设置主题策略。

• 消息模板 ⓘ

请选择消息模板

C [创建消息模板](#)

表 8-5 告警通知规则参数说明

参数名称	说明
通知规则名称	自定义通知规则名称，创建成功后不支持修改。命名要求如下： 只能是数字、字母、中文、下划线、中划线组成，且不能以下划线、中划线开头结尾，长度为1到64个字符。

参数名称	说明
企业项目	选择已创建的企业项目。 如果当前账号未开通企业项目则不显示该参数。
描述	自定义通知规则的描述，字符长度0-1024个字符。 只能是数字、字母、特定字符（_*）、空格和中文组成，不能以下划线开头结尾。
主题	SMN主题，请从下拉列表中选择。 若没有合适的主题，请单击主题选择栏下方“创建主题”，在SMN界面创建。 详细操作请参考 创建主题 。
消息模板	通知消息的模板，请从下拉列表中选择。 若没有合适的消息模板，请单击消息模板选择栏右侧“创建消息模板”，新建消息模板。详细操作请参考 创建消息模板 。

步骤5 设置完成后，单击“确定”。

----结束

更多操作

告警通知规则创建完成后，您还可以执行[表8-6](#)中的相关操作。

表 8-6 相关操作

操作	说明
修改告警通知规则	单击“操作”列的“修改”。
导出告警通知规则	- 选中单个或多个告警通知规则，单击导出已选中数据到XLSX。 - 若没有选中告警通知规则，单击导出全部数据到XLSX。
删除告警通知规则	删除告警通知规则前需要先删除该通知规则绑定的告警规则。 警告 删除操作无法恢复，请谨慎操作。 - 删除单条规则：单击对应规则“操作”列的“删除”，随后在提示页面单击“确定”即可删除。 - 删除单条或多条规则：勾选对应规则前的复选框，单击“批量删除”，随后在提示页面单击“确定”即可删除。
搜索告警通知规则	在表格上方按照属性类型筛选通知规则。

8.4 查看 LTS 告警列表

云日志服务支持对日志数据进行监控，通过配置告警规则，定时查询日志数据。当设置的匹配条数或条件表达式满足时，将告警进行上报，用户可以在LTS控制台查看告警。

前提条件

已创建告警规则。

查看告警

步骤1 登录[云日志服务控制台](#)。

步骤2 左侧导航选择“日志告警”。

步骤3 默认显示“告警列表”页面，在该页面默认显示30分钟（相对）的所有告警列表及其趋势图。

步骤4 输入查询条件后进行搜索，页面会展示该条件下的所有告警信息及这些告警的趋势图，具体查询条件如下：

- 在页面上方搜索框中可根据日志组、日志流、告警级别、规则名称进行搜索。
- 设置时间范围，默认时间范围为30分钟（相对）。
时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。
 - 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
 - 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
 - 自定义：表示查询指定时间范围的日志数据。

步骤5 查询的告警默认显示在“活动告警”页签下，将鼠标放在目标告警所在行中的“告警详情”可查看告警详情。单击告警列表中对应的“名称”，界面右侧弹出该告警的详细信息。

告警故障已经解除时，可单击列表中告警所在行后的删除按钮对该告警进行清除，被执行清除操作后的告警将会显示在“历史告警”页签。

针对已设置好的搜索条件，告警列表默认需要手动刷新，如需设置自动刷新可单击告警界面右上角 ，在弹出的下拉列表中选择“30秒自动刷新”、“1分钟自动刷新”或“5分钟自动刷新”，若在设置自动刷新后需要手动刷新，也可在下拉列表重新选择“手动刷新”。

----结束

相关文档

- LTS支持通过调用API接口实现查询活动或历史告警列表，详细请参考[查询活动或历史告警列表](#)。

- LTS支持通过调用API接口实现删除活动告警，详细请参考[删除活动告警](#)。

9 日志转储

9.1 日志转储概述

主机和云服务的日志数据上报至云日志服务LTS后，LTS会根据配置的日志存储时间定时清理日志内容。例如日志存储时间为30天，上报到LTS的日志只保存30天，30天后开始删除日志内容。请以创建日志组或日志流时设置的日志存储时间为准，详情请参考[管理日志组](#)和[管理日志流](#)。超出存储时间的日志数据将会被自动删除，对于需要长期存储的日志数据（日志持久化），LTS提供转储功能，可以将日志转储至其他云服务中进行长期保存。

日志转储功能只能拷贝已有日志，不会删除日志。LTS根据用户配置的日志存储时间定时清理日志文件，不会影响转储后的日志。

当前LTS支持以下[表9-1](#)，请根据您的业务场景进行日志转储服务选择。

表 9-1 转储服务类型

转储对象服务类型	使用场景
对象存储服务 OBS	对象存储服务（Object Storage Service，OBS）提供海量、安全、高可靠、低成本的数据存储能力，可供用户存储任意类型和大小的数据。适合企业备份/归档、视频点播、视频监控等多种数据存储场景。 - 创建周期性日志转储时，LTS支持OBS桶的存储类别为标准存储、已恢复的归档存储的OBS桶。 - 创建一次性日志转储时，LTS支持OBS桶的存储类别为标准存储。

转储对象服务类型	使用场景
分布式消息服务Kafka版（DMS）	分布式消息服务Kafka版（Distributed Message Service for Kafka）是一款基于开源社区版Kafka提供的消息队列服务，具备高效可靠的消息异步传递机制，向用户提供计算、存储和带宽资源独占式的Kafka专享实例。 主要用于不同系统间的数据交流和传递，在企业解决方案、金融支付、电信、电子商务、社交、即时通信、视频、物联网、车联网等众多领域都有广泛应用。 Kafka可以应对大量日志传输场景，应用通过异步方式将日志消息同步到消息服务，再通过其他组件对日志做实时或离线分析，也可用于关键日志信息收集进行应用监控。
数据接入服务 DIS	数据接入服务（Data Ingestion Service，简称DIS）为处理或分析流数据的自定义应用程序构建数据流管道，主要解决云服务外的数据实时传输到云服务内的问题。数据接入服务每小时可从数十万种数据源（如IoT数据采集、日志和定位追踪事件、网站点击流、社交媒体源等）中连续捕获、传送和存储数TB数据。
数据仓库服务 GaussDB(DWS)	数据仓库服务（Data Warehouse Service，简称DWS）是完全托管的企业级云上数据仓库服务，具备高性能、低成本、易扩展等特性，满足大数据时代企业数据仓库业务诉求。适用于对海量日志进行存储，分析等场景。
数据湖探索 DLI	数据湖探索（Data Lake Insight，简称DLI）是完全兼容Apache Spark、Apache Flink、Trino生态，提供一站式的流处理、批处理、交互式分析的Serverless融合处理分析服务。 适用于需要对海量日志内容进行分析，大数据处理的场景，例如游戏公司不同部门日常通过游戏数据分析平台，分析每日新增日志获取所需指标，通过数据来辅助决策等场景。

9.2 日志转储至 OBS

对象存储服务OBS提供日志存储功能；您可以将日志转储至OBS，并在OBS控制台下载日志文件。支持将日志周期性或一次性的转储至对象存储服务OBS中长期保存。

- [配置日志转储（周期性）](#)
- [配置日志转储（一次性）](#)

费用说明

日志转储成功后，LTS根据您实际的日志转储量进行收费，转储费用详情请参见[LTS价格详情](#)。OBS存储独立收费，费用详情请参见：[OBS价格详情](#)。

前提条件

- 日志已接入LTS。详细请参考[日志接入](#)。

- 已创建OBS桶。详细请参考[创建桶](#)。

配置日志转储

LTS支持将日志周期性或一次性转储至对象存储服务OBS中长期保存。

配置日志转储（周期性）

- 创建日志转储时，除需拥有LTS使用权限外，还需要拥有以下OBS桶相关授权项：设置桶ACL（obs:bucket:PutBucketAcl）、获取桶列表（obs:bucket:ListAllMyBuckets）、获取桶元数据（obs:bucket:HeadBucket）、获取桶ACL（obs:bucket:GetBucketAcl）、获取桶的加密配置（obs:bucket:GetEncryptionConfiguration）。更多信息请参见[桶相关授权项](#)。
- 云日志服务配置的日志转储是将最新产生的日志转储到OBS桶中，不会对历史日志进行转储。

步骤1 登录[云日志服务控制台](#)。

步骤2 左侧导航选择“日志转储”。

步骤3 在“日志转储”页面右上角，单击“配置转储”。

步骤4 在“配置转储”页面，设置转储日志相关参数。

图 9-1 周期性转储

The screenshot shows the 'Configure Log Archiving' (配置转储) page. At the top, there is a blue banner with a warning icon and text: '云日志服务只收取日志基础转储流量和日志高级转储流量费用，所依赖的OBS/DIS/DMS等转储对象的费用参考各服务收费标准。了解计费详情' (CloudLog Service only charges basic log archiving traffic and advanced log archiving traffic fees. The fees for the dependent OBS/DIS/DMS etc. archiving objects are subject to the pricing of each service. Learn more about billing details). Below the banner, the settings are organized into sections:

- 日志源 (Log Source):** Includes buttons for '当前账号' (Current Account) and '其他账号' (Other Account).
- 转储方式 (Archiving Method):** Includes buttons for '周期性转储' (Periodic Archiving) and '一次性转储' (One-time Archiving).
- 是否开启转储 (Enable Archiving):** A toggle switch that is currently turned on.
- 转储对象 (Archiving Object):** Includes buttons for 'OBS桶' (OBS Bucket), 'DIS通道' (DIS Channel), 'DWS集群' (DWS Cluster), and 'DLI集群' (DLI Cluster). The 'OBS桶' button is highlighted.
- 日志组名称 (Log Group Name):** A dropdown menu with a 'C' icon.
- 日志流名称 (Log Stream Name):** A dropdown menu with a 'D' icon. Below it, a note says: '请确保所选日志流上传了原始日志，否则转储后文件将没有内容' (Please ensure that the selected log stream has uploaded the original logs, otherwise the files after archiving will be empty).
- OBS桶 (OBS Bucket):** A dropdown menu with a 'C' icon. Below it, a note says: '选中的桶会将读写策略授权给云日志服务，请谨慎修改桶策略，防止转储失败。' (The selected bucket will authorize read and write policies to CloudLog Service. Please be cautious in modifying the bucket policy to prevent archiving failure).
- 自定义转储路径 (Custom Archiving Path):** A toggle switch that is currently turned off.
- 日志文件前缀 (Log File Prefix):** A text input field with the placeholder '请输入日志文件前缀' (Please enter the log file prefix).
- 转储格式 (Archiving Format):** A dropdown menu with '原始日志格式' (Original Log Format) selected.
- 转储周期 (Archiving Cycle):** A dropdown menu with '3 小时' (3 hours) selected.
- 文件名时区 (File Name Time Zone):** A dropdown menu with '(UTC) 协调世界时间' (UTC Coordinated Universal Time) selected.
- 是否投递tag (Deliver Tag):** A toggle switch that is currently turned off.
- 压缩格式 (Compression Format):** A dropdown menu with '压缩(gzip)' (Compression (gzip)) selected.

表 9-2 配置转储参数说明

参数名称	说明	示例
日志源	- 当前账号：对用户所在账号下所产生的日志进行转储。 - 其他账号：对委托人账号下所产生的日志进行转储，如需转储其他账号日志，需该账号使用者在IAM中创建委托。	当前账号
转储方式	日志源选择当前账号时，选择周期性转储。 周期性转储：日志将周期性的转储至对象存储服务（OBS）中长期保存。	周期性转储
委托名称	当转储其他账号时，需填写委托人在IAM中创建的委托名称。	-
委托人账号名称	当转储其他账号时，需填写委托人的账号名称。	-
是否开启转储	默认开启转储。	开启
转储对象	选择转储的云服务。	OBS
日志组名称	选择已创建的日志组。	-
日志流名称	选择已创建的日志流。已配置过OBS转储的日志流不能重复配置。 请确保所选日志流上传了原始日志，否则转储后文件将没有内容。	-
OBS桶	- 选择已创建的OBS桶。 - 如果没有可选择的OBS桶，单击“查看OBS”，进入对象存储服务管理控制台，创建OBS桶。 - 如果OBS桶为加密桶，则需要选择“密钥名称”，并勾选下方的“我同意在KMS创建授权给LTS账号，对转储日志加解密”。 - 选中的桶会将读写策略授权给云日志服务，请谨慎修改桶策略，防止转储失败。 - LTS支持OBS桶的存储类别为标准存储、已恢复的归档存储的OBS桶。	-
密钥名称	对于加密的OBS桶，选择密钥名称。如果没有可选择的密钥，单击“创建密钥并授权”，进入数据加密控制台，创建密钥。	-

参数名称	说明	示例
自定义转储路径	- 开启：将日志转储至自定义路径中，用于区分不同日志流之间的转储日志文件。 格式为：/LogTanks/RegionName/%GroupName/%StreamName/<i>自定义转储路径</i>。自定义转储路径默认为 lts/%Y/%m/%d，其中%Y代表年，%m代表月，%d代表日，格式需要符合如下规范： “/LogTanks/RegionName”为系统默认路径，不可以修改。 - 新增%GroupName代表日志组名称，%StreamName代表日志流名称。 - 名称只能由英文字母、数字及特殊字符“&”“\$”“@”“.”“.”“.”“=”“+”“?”“-”“.”“-”“/”和“%”组成，且“%”后只可跟Y（年）、m（月）、d（日）、H（时）、M（分），在%Y、%m、%d、%H和%M前后可以添加任意长度字符，并且可对其先后顺序进行调换。 - 自定义转储路径名称不允许为空，长度限制为1~128个字符。 - 示例： - 输入LTS-test/%Y/%m/%d/%H/%M，则日志转储路径为：LogTanks/RegionName/LTS-test/Y/m/d/H/M/日志文件名称。 - 输入LTS-test/%d/%H/%m/%Y，则日志转储路径为：LogTanks/RegionName/LTS-test/d/H/m/Y/日志文件名称。 - 不开启：将日志转储至系统默认路径中。系统默认路径为：LogTanks/RegionName/2019/01/01/日志组/日志流/日志文件名称。	LTS-test/ %GroupName/ %StreamName/ %Y/%m/%d/ %H/%M
日志文件前缀	转储至OBS桶中的日志文件前缀。 日志文件前缀需符合如下规范： - 名称长度限制为0~64个字符。 - 名称只能由英文大小写字母、数字、中划线“-”、下划线“_”和小数点“.”组成。 示例：输入LTS-log，则日志文件名称为：LTS-log_日志文件名称。	LTS-log

参数名称	说明	示例
转储格式	用于配置日志的转储格式，可选择“原始日志格式”、“Json格式”和ORC。 ORC转储格式（包括ORC字段和无效字段填充）仅支持华东-上海一的白名单用户使用，其他局点暂不支持该功能。 - 原始日志格式示例： 云日志服务控制台展示的日志内容的格式为原始日志格式。 Sep 30 07:30:01 ecs-bd70 CRON[3459]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1) - JSON格式示例： { "host_name": "ecs-bd70", "ip": "192.168.0.54", "line_no": 249, "message": "Sep 30 14:40:01 ecs-bd70 CRON[4363]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)\n", "path": "/var/log/syslog", "time": 1569825602303 }	Json
ORC字段	转储格式选择ORC时需要设置ORC字段。 支持自动配置字段或者单击添加，在下拉框选择或输入键值、选择类型。 - 键：日志字段名称，同一个字段名在ORC字段中只能配置一次，不支持多次使用。 - 类型：string、boolean、int、long、float、double。 日志转储过程中，会将云日志服务中的日志字段由string类型转换为ORC目标类型。如果转换到非string类型失败，int/long/float/double类型会置为默认值0。	-
无效字段填充	转储格式选择ORC时支持设置无效字段填充。 若日志中没有值与上方配置的键值对应，则会用无效字段填充。	-
转储周期	日志自动转储至OBS桶的时间间隔，支持2分钟、5分钟、30分钟、1小时、3小时、6小时、12小时。	3小时
文件名时区	日志自动转储至OBS桶时，按照UTC时间生成转储目录及文件名称。	(UTC)协调世界时间

参数名称	说明	示例
是否投递tag	如主机日志，转储时会增加采集器收集的tag字段。 - 不开启：不会投递tag。 - 开启：默认的投递tag有：主机信息（hostIP、hostId、hostName、pathFile、collectTime）；kubernetes信息（clusterName、clusterId、nameSpace、podName、appName、containerName）。可选择公共tag有：regionName、projectId、logStreamName、logGroupName。当开启投递tag后，转储格式必须是JSON格式。 - 转储标签：开启后，会将日志流标签添加至转储内容。	开启
压缩格式	支持不压缩、压缩gzip、压缩zip、压缩snappy。 说明 压缩格式支持白名单用户申请 提交工单 开通。	gzip

步骤5 单击“确定”，完成配置。当转储任务状态为“正常”时，表示转储任务创建成功。

步骤6 单击“转储对象”列的OBS桶名称，可以跳转至OBS控制台，查看转储的日志文件。

转储到OBS后的日志，支持从OBS下载到本地进行查看。转储至OBS的日志支持下载的格式：原始日志、JSON格式。

图 9-2 转储至 OBS

☐ 规则ID	☐ 日志源	☐ 日志流	☐ 日志源	☐ 转储类型	☐ 转储对象	☐ 转储格式	☐ 转储间隔	☐ 状态	☐ 创建时间
☐ c1a5-abc	1a-	obs_2025	obs_202501	当前转储	OBS桶	原始日志格式	2分钟	正常	2025/04/01 20:15:3 GMT+08:00

----结束

配置日志转储（一次性）

说明

目前此功能支持华东-上海一、华东-上海二、华北-北京四、华南-广州、亚太-新加坡，其他局点如有需要请[提交工单](#)申请开通。

步骤1 在“日志转储”页面右上角，单击“配置转储”。

步骤2 在“配置转储”页面，设置转储日志相关参数。

图 9-3 一次性转储

配置转储

云日志服务只收取日志基础转储流量和日志高级转储流量费用。所依赖的OBS/DIS/DMS等转储对象的费用参考各服务收费标准。[了解详情详情](#)

转储方式

周期性转储

一次性转储

单次一次性转储日志最多上限2000万条。转储文件个数上限200个。同一时间内仅支持执行一个转储任务。

转储对象

OBS桶

日志组名称

请选择-

C

日志流名称

请选择-

?

请确保所选日志流上传了原始日志。否则转储后文件将没有内容

过滤条件

关键词

请输入转储采集相关的关键词等过滤条件

转储时间范围

15分钟 (相对)

日志总条数

0

查询

转储文件个数

-

1

+

?

OBS桶

请选择-

C

首次配置一次性转储到未授权的OBS桶中时，LTS服务会授权给OBS桶ACL规则，授权生效需要15分钟，如果您第一次配置一次性转储后失败，任务会在15分钟后自动重试。
请谨慎修改桶策略，防止转储失败。

所属桶目录

请输入所属OBS桶目录

转储文件名称

请输入日志文件前缀

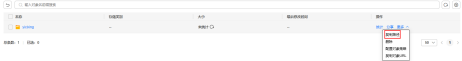
转储格式

原始日志格式

表 9-3 配置转储参数说明

参数名称	说明	示例
转储方式	一次性转储：日志将一次性的转储至对象存储服务（OBS）中长期保存。	一次性转储
转储对象	选择转储的云服务。	OBS
日志组名称	选择已创建的日志组。	-
日志流名称	选择已创建的日志流。已配置过OBS转储的日志流不能重复配置。 请确保所选日志流上传了原始日志，否则转储后文件将没有内容。	-
过滤条件	默认关键词过滤，在输入框填写需要过滤的关键词。	-

参数名称	说明	示例
转储时间范围	时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。 ● 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。 ● 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。 ● 自定义：表示查询指定时间范围的日志数据。	-
日志总条数	选择的转储时间范围内的日志总条数。 单击日志总条数后面的查询，LTS自动计算并显示转储时间范围内的日志总条数。若没有日志条数，请修改转储时间范围。	-
转储文件个数	单次一次性转储日志条数上限2000万条，转储文件个数上限200个。	-
OBS桶	● 选择已创建的OBS桶。 如果没有可选择的OBS桶，单击“查看OBS”，进入对象存储服务管理控制台，创建OBS桶。 ● LTS目前仅支持存储类别为“标准存储”的OBS桶。 ● 不支持转储到存储类型为归档存储或配置了跨区域复制的OBS桶。 ● 首次配置一次性转储到未授权的OBS桶中时，LTS服务会授权给OBS桶ACL规则，授权生效需要15分钟，如果您第一次配置一次性转储后失败，任务会在15分钟后重试。请谨慎修改桶策略，防止转储失败。	-

参数名称	说明	示例
所属桶目录	所属OBS桶目录路径，不要以/开头或结尾，只能由英文字母、数字、中划线、下划线、小数点组成。 您可以参考如下方法获取所属桶目录： - 在OBS控制台桶列表页面，单击目标桶名称进入对象页面。 - 在“对象”页面，找到需要存放LTS日志的OBS文件名称，单击该文件夹操作列的“更多 > 复制路径”。（以下示例仅供参考，请以实际路径为准） 如果需要将日志存入该文件夹test/yicixing，复制路径结果为“test/yicixing”，复制的路径就是桶目录。 图 9-4 复制路径 	test/yicixing
转储文件名称	自定义转储文件名称，只能由英文字母、数字、中划线、下划线、小数点组成。	-
转储格式	用于配置日志的转储格式，可选择原始日志格式、Json格式、CSV格式。 - 原始日志格式示例： 云日志服务控制台展示的日志内容的格式为原始日志格式。 Sep 30 07:30:01 ecs-bd70 CRON[3459]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1) - Json格式示例： { "host_name": "ecs-bd70", "ip": "192.168.0.54", "line_no": 249, "message": "Sep 30 14:40:01 ecs-bd70 CRON[4363]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)", "path": "/var/log/syslog", "time": 1569825602303 } - CSV格式：以表格的形式展示日志内容。	Json

步骤3 单击“确定”，完成配置。当转储任务状态为“正常”时，表示转储任务创建成功。

日志转储任务创建成功后无法修改，请确保参数已填写正确。

步骤4 单击“转储对象”列的OBS桶名称，可以跳转至OBS控制台，查看转储的日志文件。
转储到OBS后的日志，支持从OBS下载到本地进行查看。

图 9-5 一次性转储至 OBS



----结束

修改日志转储

只支持修改周期性转储任务，一次性转储任务不支持修改。

1. 在日志转储列表中，单击待修改配置转储任务所在行的“修改”，弹出“修改转储”对话框，进行修改。
2. 修改完成后，单击“确定”。

查看转储详情

1. 在日志转储列表中，单击待查看配置转储任务所在行的“更多>详情”。
2. 在弹出的“转储详情”页面中，可查看日志转储详情。

删除转储任务

如果日志不再需要转储，可以删除转储任务。



警告

删除转储任务将不再对日志进行转储，且删除后无法恢复，请谨慎操作。

- 删除转储任务后，之前已经转储日志将会继续保存在OBS。
 - 创建转储任务时，选中的OBS桶会将读写策略授权给云日志服务。当多个转储任务使用同一OBS桶时，如您需要删除转储任务，请按如下操作：
 - 如果仅使用该OBS桶创建了一个转储任务，删除该转储任务时，请在对象存储服务（Object Storage Service, OBS）中“权限控制 > 桶ACL”里删除特定用户的桶访问权限。
 - 如果使用该OBS桶创建了多个转储任务，请勿删除桶访问权限，否则会导致转储失败。
1. 删除单个转储任务。
 - a. 单击待删除的转储任务所在行操作列的删除。
 - b. 在弹出的“删除转储”页面，单击“确定”，删除转储任务。
 2. 批量删除转储任务。
 - a. 批量勾选待删除的转储任务，单击列表左上方“批量删除”。
 - b. 在弹出的“删除转储”页面，单击“确定”，删除所勾选的转储任务。

查看转储状态

日志转储任务的转储状态共分为正常、异常、关闭三种状态。

- 正常：日志转储任务正常进行。
- 异常：日志转储任务异常，可能是如下原因导致：
 - OBS桶策略异常，请您在对象存储服务中设置访问控制策略。
 - OBS加密桶的密钥被删除或被取消授权，请您确保授权密钥的合法性。
- 关闭：日志转储任务停止。

相关文档

- 如果在配置日志转储时遇到问题，您可以参考[日志转储常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志转储、查询日志转储、删除日志转储等，详细请参考[日志转储](#)。

9.3 日志转储至 DIS

DIS提供丰富的大数据分析能力，可以将大量日志文件传输到云端做备份，进行离线分析、存储查询及机器学习，还能用于数据丢失或异常后的恢复和故障分析。同时大量小文本文件可合并转储为大文件，提高数据处理性能。您可以根据业务场景选择是否使用DIS进行日志转储。

说明

- 建议您优先使用[转储至DMS](#)。
- 目前此功能仅支持华北-北京四、华北-北京一、华东-上海二、华南-广州、中国-香港、亚太-新加坡、华东-上海一局点，其他局点需要给DIS服务提交工单申请开通才能使用。详细操作请参考[提交工单](#)。

费用说明

日志转储成功后，LTS根据您的实际的日志转储量进行收费，转储费用详情请参见[LTS价格详情](#)。DIS存储独立收费，费用详情请参见：[DIS价格详情](#)。

前提条件

- 日志已接入LTS，详细请参考[日志接入](#)。
- 已开通DIS，详细请参考[开通DIS通道](#)。
- 建议您优先使用[转储至DMS](#)。
- 目前此功能仅支持华东-上海一、华北-北京四、华北-北京一、华东-上海二、华南-广州、中国-香港、亚太-新加坡局点，其他局点需要给DIS服务提交工单申请开通才能使用。详细操作请参考[提交工单](#)。

日志转储至 DIS

步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。

步骤2 左侧导航栏中，单击“日志转储”。

步骤3 在“日志转储”页面右上角，单击“配置转储”。

步骤4 在“配置转储”页面，设置转储日志相关参数。

表 9-4 配置转储参数说明

参数名称	说明	示例
日志源	- 当前账号：对用户所在账号下所产生的日志进行转储。 - 其他账号：对委托人账号下所产生的日志进行转储，如需转储其他账号日志，需该账号使用者在IAM中创建委托，创建完成后，请记下委托名称和委托人账号名称。	当前账号
委托名称	当转储其他账号时，需填写委托人在IAM中创建的委托名称。	-
委托人账号名称	当转储其他账号时，需填写委托人的账号名称。	-
是否开启转储	选择开启转储。	开启
转储对象	选择转储的云服务。	DIS
日志组名称	选择已创建的日志组。	-
日志流名称	选择已创建的日志流。已配置过DIS转储的日志流不能重复配置。	-
通道名称	选择已创建的DIS通道。如果没有可选择的通道，单击“查看DIS通道”，进入数据接入服务管理控制台，创建接入通道。	-
转储格式	用于配置日志的转储格式，可选择“原始日志格式”和“JSON格式”。 - 原始日志格式示例： 云日志服务控制台展示的日志内容的格式为原始日志格式。 Sep 30 07:30:01 ecs-bd70 CRON[3459]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1) - JSON格式示例： { "host_name": "ecs-bd70", "ip": "192.168.0.54", "line_no": 249, "message": "Sep 30 14:40:01 ecs-bd70 CRON[4363]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)", "path": "/var/log/syslog", "time": 1569825602303 }	JSON
转储周期	日志将实时转储至DIS通道中。	实时

参数名称	说明	示例
是否投递tag	如主机日志，转储时会增加采集器收集的tag字段。 - 不开启：不会投递tag。 - 开启：默认的投递tag有：主机信息（hostIP、hostId、hostName、pathFile、collectTime）；kubernetes信息（clusterName、clusterId、nameSpace、podName、appName、containerName）。可选择公共tag有：regionName、projectId、logStreamName、logGroupName。当开启投递tag后，转储格式必须是JSON格式。 - 转储标签：开启后，会将日志流标签添加至转储内容。	开启

步骤5 单击“确定”，完成配置。当转储任务状态为“正常”时，表示转储任务创建成功。

步骤6 单击“转储对象”列的DIS通道名称，可以跳转至DIS控制台，查看转储的日志文件。

转储后的日志，支持下载到本地进行查看。

 警告

删除转储任务将不再对日志进行转储，且删除后无法恢复，请谨慎操作。

- 删除转储任务后，之前已经转储日志将会继续保存在DIS。
- 当删除该转储任务时，请在数据接入服务（Data Ingestion Service，DIS）中，单击“通道管理”，选择该DIS实例进入实例详情页面。在授权管理中，删除上传权限。
- 删除单个转储任务。
 - a. 单击待删除的转储任务所在行操作列的删除。
 - b. 在弹出的“删除转储”页面，单击“确定”，删除转储任务。
- 批量删除转储任务。
 - a. 批量勾选待删除的转储任务，单击列表左上方“批量删除”。
 - b. 在弹出的“删除转储”页面，单击“确定”，删除所勾选的转储任务。

----结束

相关文档

- 如果在配置日志转储时遇到问题，您可以参考[日志转储常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志转储、查询日志转储、删除日志转储等，详细请参考[日志转储](#)。

9.4 日志转储至 DMS

分布式消息服务 DMS提供日志实时处理管道，您可以通过分布式消息服务API实时消费处理日志。

 说明

目前此功能仅支持白名单用户提交工单申请使用。详细操作请参考[提交工单](#)。

费用说明

日志转储成功后，LTS根据您的实际的日志转储量进行收费，转储费用详情请参见[LTS价格详情](#)。DMS存储独立收费，费用详情请参见：[DMS价格详情](#)。

前提条件

- 日志已接入LTS。
- 已[购买DMS](#)。
- 在注册DMS Kafka实例前，需在安全组中，开放[入方向规则](#)198.19.128.0/17和9011端口。如果DMS的子网关联了网络ACL，需要放开网络ACL入方向规则，协议为TCP，入方向规则源地址198.19.128.0/17，端口范围为1-65535，目的地址为全放开，目的端口为9011。

日志转储至 DMS

- 步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2** 左侧导航栏中，单击“日志转储”。
- 步骤3** 在“日志转储”页面右上角，单击“配置转储”。
- 步骤4** 在“配置转储”页面，设置转储日志相关参数。

表 9-5 配置转储参数说明

参数名称	说明	示例
日志源	• 当前账号：对用户所在账号下所产生的日志进行转储。 • 其他账号：对委托人账号下所产生的日志进行转储，如需转储其他账号日志，需该账号使用者在IAM中创建委托。	当前账号
委托名称	当转储其他账号时，需填写委托人在IAM中创建的委托名称。	-
委托人账号名称	当转储其他账号时，需填写委托人的账号名称。	-
是否开启转储	默认开启转储。	开启
转储对象	选择转储的云服务。	DMS

参数名称	说明	示例
日志组名称	选择已创建的日志组。	-
日志流名称	选择已创建的日志流。 已配置过DMS转储的日志流不能重复配置。	-
Kafka实例	选择Kafka实例。如果没有可选择的实例，单击“查看Kafka实例”，进入分布式消息服务控制台，创建Kafka实例。 如果Kafka实例已注册（如果未注册，请注册Kafka实例，操作指导请参见： 注册Kafka实例 ），可以选择修改Kafka实例。 创建Kafka实例时，设置实例的访问方式：内网访问开启密文接入，“kafka安全协议”选择“SASL_SSL”，设置用户名和密码。同时开启“SASL PLAIN机制”。详细操作请参考 购买实例 。 若对Kafka实例进行变更操作（例如删除topic或扩缩容节点），可能导致Kafka实例被拉黑，从而无法将数据发送到Kafka实例。建议变更Kafka实例前先删除转储任务。待变更完成后再重新创建转储任务。	-
Topic	选择Kafka实例的Topic，如果没有可选择的topic，进入分布式消息服务控制台，在待转储的Kafka实例中的“Topic管理”页面，创建Topic。	topic-01
转储格式	用于配置日志的转储格式，可选择“原始日志格式”和“JSON格式”。 - 原始日志格式示例： 云日志服务控制台展示的日志内容的格式为原始日志格式。 Sep 30 07:30:01 ecs-bd70 CRON[3459]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1) - JSON格式示例： { "host_name": "ecs-bd70", "ip": "192.168.0.54", "line_no": 249, "message": "Sep 30 14:40:01 ecs-bd70 CRON[4363]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)\\n", "path": "/var/log/syslog", "time": 1569825602303 }	RAW
转储周期	日志将实时转储至Kafka实例中。	实时

参数名称	说明	示例
用户日志字段	当转储格式选择JSON时，需要设置该参数。 开启转储用户日志字段后，支持设置转储原始日志和转储类型。转储类型支持转储所有字段或自定义转储字段。 • 开启“转储原始日志”，转储message字段，关闭“转储原始日志”，不转储message字段。 • 转储类型选择“转储所有字段”是转储日志下所有的字段。 • 转储类型选择“自定义转储字段”是手动配置用户日志字段。 - 单击“自动配置”，云日志服务LTS会根据近15分钟的第一条日志内容、结构化字段、常见内置保留字段（例如hostIP、hostName、pathFile）自动生成内容，您可以根据需要添加或者删除字段。 - 单击“添加字段”，在日志字段和表字段的下拉框中，选择您需要转储的字段。 - 单击操作列下的  删除不需要转储的字段。	-
LTS内置字段和用户自定义Tag	当转储格式选择JSON时，需要设置该参数。 是否开启转储LTS内置字段和用户自定义Tag。开启后，转储类型支持转储所有字段或自定义转储字段。 • 转储类型选择“转储所有字段”是转储日志下所有的内置字段和用户自定义字段。 • 转储类型选择“自定义转储字段”是手动配置用户日志字段。 - 单击“自动配置”，云日志服务会根据近15分钟的第一条日志内容、结构化字段、常见内置保留字段（例如hostIP、hostName、pathFile）自动生成内容，您可以根据需要添加或者删除字段。 - 单击“添加字段”，在日志字段和表字段的下拉框中，选择您需要转储的字段。 - 单击操作列下的  ，删除不需要转储的字段。	-

参数名称	说明	示例
日志流标签字段	当转储格式选择JSON时，需要设置该参数。 是否开启转储日志流标签字段。开启后，转储类型支持转储所有字段或自定义转储字段。如何设置标签请参考管理日志流。 - 转储类型选择“转储所有字段”是转储日志下所有的日志流标签字段。 - 转储类型选择“自定义转储字段”是手动配置用户日志字段。 - 单击“自动配置”，云日志服务会根据近15分钟的第一条日志内容、结构化字段、常见内置保留字段（例如hostIP、hostName、pathFile）自动生成内容，您可以根据自己的需要添加或者删除字段。 - 单击“添加字段”，在日志字段和表字段的下拉框中，选择您需要转储的字段。 - 单击操作列下的，删除不需要转储的字段。	-
更多配置	当转储格式选择JSON时，需要设置该参数。 若日志中无值与上方配置的键值对应，则会用无效字段填充。	-

步骤5 单击“确定”，完成配置。当转储任务状态为“正常”时，表示转储任务创建成功。

步骤6 单击“转储对象”列的名称，可跳转到Kafka实例的基本信息页面。

- 创建转储任务成功后，支持查看、修改、删除转储任务。
- 删除转储任务后，之前已经转储日志将会继续保存在DMS。



警告

删除转储任务将不再对日志进行转储，且删除后无法恢复，请谨慎操作。

- 删除单个转储任务。
 - 单击待删除的转储任务所在行操作列的删除。
 - 在弹出的“删除转储”页面，单击“确定”，删除转储任务。
- 批量删除转储任务。
 - 批量勾选待删除的转储任务，单击列表左上方“批量删除”。
 - 在弹出的“删除转储”页面，单击“确定”，删除所勾选的转储任务。

----结束

注册 Kafka 实例

- 如果选择Kafka实例未注册，单击“注册”，跳转到注册Kafka实例页面。

2. 注册Kafka实例相关参数说明。

表 9-6 参数说明

参数名称	说明	示例
Kafka实例	dms实例名称。	Kafka-01
打通DMS网络	打通Kafka实例和LTS服务的网络，用户LTS服务通过该网络发送转储数据。	-
用户名	如果Kafka实例开启了sasl认证，需要输入sasl认证的用户名。	DMS
密码	如果Kafka实例开启了sasl认证，需要输入sasl认证的密码。	-

3. 单击“确定”，完成注册Kafka实例。

相关文档

- 如果在配置日志转储时遇到问题，您可以参考[日志转储常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志转储、查询日志转储、删除日志转储等，详细请参考[日志转储](#)。

9.5 日志转储至 DWS

数据仓库服务GaussDB(DWS) 是一种基于华为云基础架构和平台的在线数据处理数据库，提供即开即用、可扩展且完全托管的分析型数据库服务。转储至数据仓库服务GaussDB(DWS)，可以将日志中的结构化字段转储到DWS数据库表中，您可以根据业务场景选择是否使用DWS进行日志转储。

说明

目前此功能仅支持华北-北京四、华东-上海一、华南-广州、亚太-新加坡局点，其他局点需要提交工单申请使用。详细操作请参考[提交工单](#)。

费用说明

日志转储成功后，LTS根据您的实际的日志转储量进行收费，转储费用详情请参见[LTS价格详情](#)。DWS存储独立收费，费用详情请参见：[DWS价格详情](#)。

前提条件

- 日志已接入云日志服务（LTS）。
- 日志流已配置结构化规则。
- 购买[独享型且规格为网络型的弹性负载均衡（ELB）实例](#)。
- [创建GaussDB\(DWS\)存算一体集群](#)，并且绑定弹性负载均衡（ELB）。暂不支持DWS集群跨VPC绑定弹性负载均衡（ELB）。

日志转储至 DWS

- 步骤1 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2 在左侧导航栏中选择“日志转储”。
- 步骤3 在“日志转储”页面中，单击右上角“配置转储”。
- 步骤4 在“配置转储”页面中，选择转储对象“DWS集群”，并配置各参数信息。

表 9-7 配置转储参数说明

参数名称	说明	示例
是否开启转储	选择是否开启转储。	开启
转储对象	选择转储的云服务。	DWS集群
日志组名称	选择已创建的日志组。	-
日志流名称	选择已创建的日志流。 已配置过DWS集群转储的日志流不能重复配置。	-
集群名称	已创建的集群名称。	test
数据库名称	集群的数据库名称。有两种数据库，分别是“gaussdb”和“postgres”。默认集群数据库为“gaussdb”。	gaussdb
用户名	数据库的管理员用户名。	lts-test
密码	数据库的管理员密码。	-
schema名称	数据库对象的集合名称。	-
表名	schema中的表名称。	-

参数名称	说明	示例
字段映射	将内置字段以及日志中配置的结构化字段和类型，映射到数据库字段。 内置字段有13个，分别是hostIP、hostId、hostName、pathFile、collectTime、clusterName、clusterId、podName、containerName、regionName、projectId、logGroupName和logStreamName。 当结构化字段类型和数据库表字段类型一致时，支持将日志的结构化字段转储至数据仓库服务GaussDB(DWS)，否则转储无效。 - 在结构化字段和表字段的▼下拉框，选择您需要转储的字段。 - 单击操作列下的🗑️，删除不需要的转储字段。 - 单击➕添加，添加需要转储的字段。	hostIP

- 步骤5** 完成后单击“确定”。
- 创建转储任务成功后，支持查看、修改、删除转储任务。
 - 删除转储任务后，之前已经转储日志将会继续保存在DWS。

 **警告**

删除转储任务将不再对日志进行转储，且删除后无法恢复，请谨慎操作。

- 删除单个转储任务。
 - 单击待删除的转储任务所在行操作列的删除。
 - 在弹出的“删除转储”页面，单击“确定”，删除转储任务。
- 批量删除转储任务。
 - 批量勾选待删除的转储任务，单击列表左上方“批量删除”。
 - 在弹出的“删除转储”页面，单击“确定”，删除所勾选的转储任务。

----结束

相关文档

- 如果在配置日志转储时遇到问题，您可以参考[日志转储常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志转储、查询日志转储、删除日志转储等，详细请参考[日志转储](#)。

9.6 日志转储至 DLI

数据湖探索（Data Lake Insight，简称DLI）是完全兼容Apache Spark、Apache Flink、openLooKeng（基于Apache Presto）生态，提供一站式的流处理、批处理、交互式分析的Serverless融合处理分析服务。用户不需要管理任何服务器，即开即用。支持标准SQL/Spark SQL/Flink SQL，支持多种接入方式，并兼容主流数据格式。数据无需复杂的抽取、转换、加载，使用SQL或程序就可以对云上CloudTable、RDS、DWS、CSS、OBS、ECS自建数据库以及线下数据库的异构数据进行探索。

基于转储DLI功能，您可以轻松将LTS中的日志按照字段映射关系转储到DLI数据库表中，进行后续的大数据分析工作。

说明

目前此功能仅在华北-北京四、华南-广州、华东-上海一局点支持白名单用户提交工单申请使用，详细操作请参考[提交工单](#)，其他局点暂不支持该功能。

费用说明

日志转储成功后，LTS根据您的实际的日志转储量进行收费，转储费用详情请参见[LTS价格详情](#)。DWS存储独立收费，费用详情请参见：[DLI价格详情](#)。

前提条件

- 日志已接入云日志服务（LTS）。
- 日志流已配置结构化规则。
- 已在DLI中创建数据库和表，创建数据库表时数据位置选择OBS，数据格式选择JSON。详细操作请参见[创建数据库和表](#)。
- 已在DLI中创建队列，该队列将用来将数据导入DLI的表中。详细操作请参见[创建队列](#)。

日志转储至 DLI

- 步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2** 在左侧导航栏中选择“日志转储”。
- 步骤3** 在“日志转储”页面中，单击右上角“配置转储”。
- 步骤4** 在“配置转储”页面中，选择转储对象“DLI集群”，并配置各参数信息。

表 9-8 配置转储参数说明

参数名称	说明	示例
是否开启转储	选择是否开启转储。	开启
转储对象	选择转储的云服务。	DLI集群
日志组名称	选择已创建的日志组。	-

参数名称	说明	示例
日志流名称	选择已创建的日志流。 已配置过DLI集群转储的日志流不能重复配置。	-
DLI-数据库	需要转储的目标DLI数据库名称。更多信息请参见 DLI库表管理 。	test
DLI-数据表	需要转储的目标DLI数据库表名称。更多信息请参见 DLI库表管理 。	-
表普通列映射	将内置字段以及日志中配置的结构化字段和类型，映射到数据库表字段。转储类型支持转储所有字段和自定义转储字段。 表普通列映射和表分区列映射添加表字段的总数为创建表的总列数。 内置字段有13个，分别是hostIP、hostId、hostName、pathFile、collectTime、clusterName、clusterId、podName、containerName、regionName、projectId、logGroupName和logStreamName。 - 选择“转储所有字段”转储日志下所有的日志流标签字段。 - 选择“自定义转储字段”手动配置用户日志字段。 - 单击“自动配置”，云日志服务会根据采集时预览数据中的第一条日志字段、结构化字段、常见内置保留字段（例如hostIP、hostName、pathFile）自动生成内容，您可以根据自己的需要添加或者删除字段。 - 单击“添加”，在日志字段和表字段的下拉框中，选择您需要转储的字段。 - 单击操作列下的，删除不需要转储的字段。	-
表分区列映射	存储时根据设置的字段值进行分区。将内置字段以及日志中配置的结构化字段和类型，映射到数据库表字段。 内置字段有13个，分别是hostIP、hostId、hostName、pathFile、collectTime、clusterName、clusterId、podName、containerName、regionName、projectId、logGroupName和logStreamName - 单击 添加，在分区字段和表字段的下拉框中，选择您需要转储的字段。能添加一个表字段。 - 单击操作列下的，删除不需要的转储字段。	-

步骤5 完成后单击“确定”。

- 创建转储任务成功后，支持查看、修改、删除转储任务。
- 删除转储任务后，之前已经转储日志将会继续保存在DLI。

 警告

删除转储任务将不再对日志进行转储，且删除后无法恢复，请谨慎操作。

- 删除单个转储任务。
 - a. 单击待删除的转储任务所在行操作列的删除。
 - b. 在弹出的“删除转储”页面，单击“确定”，删除转储任务。
- 批量删除转储任务。
 - a. 批量勾选待删除的转储任务，单击列表左上方“批量删除”。
 - b. 在弹出的“删除转储”页面，单击“确定”，删除所勾选的转储任务。

----结束

相关文档

- 如果在配置日志转储时遇到问题，您可以参考[日志转储常见问题](#)获取解决方案。
- LTS支持通过调用API接口实现创建日志转储、查询日志转储、删除日志转储等，详细请参考[日志转储](#)。

10 日志消费与加工

10.1 DSL 加工（邀测）

10.1.1 DSL 加工概述

DSL（Domain Specific Language）加工是LTS为您提供的一站式日志加工平台，基于领域自定义的脚本语言和200多个内置函数，您可以在LTS控制台实现端到端的日志规整、富化、脱敏、过滤等加工任务。

了解更多日志加工的概念信息，请参考[基本概念](#)。

说明

DSL加工的功能在邀测中，仅支持华北-北京四、华北-乌兰察布一、华东-上海一、华南-广州、亚太-新加坡、中国-香港的白名单用户使用，其他局点暂不支持。

背景信息

用户对采集到LTS的日志流有二次加工的诉求，目前使用函数加工有以下缺点：

1. 用户有多条日志流需要转换成不同的日志流结构，一个函数只能转换一个日志流，使用不方便，有新增日志结构的时候就需要新增函数，功能繁琐，使用不够灵活。
2. 函数加工需要使用函数工作流服务，且还需要独立收费，长期使用增加成本。

应用场景

- 提取结构化的数据，方便后续检索分析，生成仪表盘等。
- 日志减肥瘦身，节约后续使用成本。丢弃不需要的日志数据，节约存储成本、流量成本。
- 敏感数据脱敏，例如：将用户的身份证、手机号码脱敏。
- 日志分类投递，例如：按照日志级别：ERROR、WARNING、INFO将日志分类，然后分发到不同的日志主题。

方案介绍

云日志服务支持通过创建DSL加工任务，将用户源日志流数据进行加工输出到目标日志流中，主要操作流程如下：

1. 通过协同消费组对源日志流的数据进行读取。
2. 通过加工规则对读取到的每一条数据进行加工处理。
3. 将加工后的数据写入目标日志流。数据加工完成后，您可以在目标日志流中查看加工后的数据。

功能特性

云日志服务提供数据加工功能，用于数据的规整、富化、脱敏和过滤。

- 数据规整：针对混乱格式的日志进行字段提取、格式转换，获取结构化数据以支持后续的流处理、数据仓库计算。
- 数据富化：对日志（例如订单日志）和维表（例如用户信息表）进行字段连接（JOIN），为日志添加更多维度的信息，用于数据分析。
- 数据脱敏：对数据中包含的密码、手机号、地址等敏感信息进行脱敏。
- 数据过滤：过滤出关键服务的日志，用于重点分析。

10.1.2 创建 DSL 加工任务

DSL（Domain Specific Language）加工是LTS为您提供的一站式日志加工平台，基于领域自定义的脚本语言和200多个内置函数，您可以在LTS控制台实现端到端的日志规整、富化、流转、脱敏、过滤等加工任务。详细加工语法请参考[DSL数据加工语法（邀测）](#)。

对源日志内容已完成结构化配置。若用户在日志接入LTS时没有配置ICAgent结构化解析，可以单独给目标日志流配置ICAgent结构化解析或云端结构化解析。更多内容请参考[配置ICAgent结构化解析](#)和[设置云端结构化解析日志](#)。

前提条件

已成功采集到日志。

创建 DSL 加工任务

- 步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2** 在左侧导航栏中选择“日志加工 > DSL加工”，单击“新建DSL加工任务”。
- 步骤3** 在基本信息页面中，完成如下配置后，然后单击“下一步”。

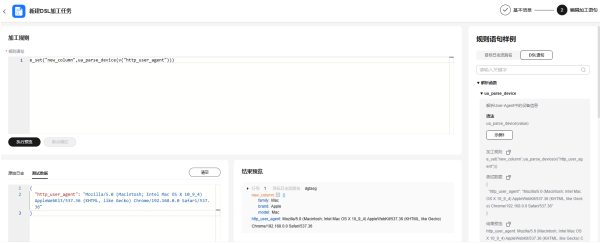
表 10-1 基本信息参数

参数	说明
任务名称	DSL加工任务的名称。只能由英文字母、数字、中划线、下划线、小数点组成，长度范围为1~64个字符。
启用状态	默认开启。

参数	说明
源日志流	- 当前账号 选择日志组、日志流，即表示源日志组、日志流中的日志内容通过DSL加工处理后，将存储到目标日志流中。 - 其他账号 填写委托名称、委托人账号名称，选择日志组和日志流。关于委托信息请参考创建委托。
目标日志流	单击“添加”，选择日志组和日志流，填写日志流别名（用于DSL加工语句编写）。最多支持添加30个目标日志流。 使用DSL加工规则将源日志流分发到多个目标日志流时，需要在加工语句中使用特定函数指定目标日志流别名，若未明确指定日志流别名，LTS系统默认只会将加工结果输出到第一个目标日志流。
加工范围	白名单功能，数据加工的时间范围。选择“某时间开始”或“特定时间范围”可以对历史日志数据进行加工。 - 所有：从源日志流接收到第一条日志的时间开始数据加工任务，直到加工任务被手动停止。 - 某时间开始：指定数据加工任务的开始时间，从该时间点开始加工，直到加工任务被手动停止。 - 特定时间范围：指定数据加工任务的起止时间，加工任务执行到指定时间后自动停止。
高级配置	对于加工语句中需要使用的密码信息（例如数据库连接密码），日志服务支持使用键值对形式保存在密钥对中。 单击“添加”，填写key值、Value值。

步骤4 在编辑加工语句页面中，参考规则语句样例填写规则语句，查看结果预览。

图 10-1 加工规则



1. 在右侧规则语句样例下方的DSL语句，根据业务需求选择函数语法，然后在规则语句下方输入加工语法。更多加工语法信息请参考[DSL数据加工语法（邀测）](#)。
2. 在“原始日志”页签，设置搜索时间框的时间范围，然后选择需要加工的日志数据，单击“加入测试数据”。
3. 单击“执行预览”，按照加工语法执行后，在结果预览下方查看结果。若需要排查或修复问题，您可以使用“断点调试”，提高开发效率。

步骤5 结果预览显示正常，单击“确定”。创建成功后在DSL加工页面生成一条任务明细。

- 单击“导出 > 导出全部数据到XLSX”，将所有加工任务导出到本地查看。

- 单击DSL加工任务名称进入详情页面，查看该任务的基本信息和执行详情。
- 单击DSL加工任务操作列的“修改”进入修改页面，按照表10-1修改相关参数。
- 单击DSL加工任务操作列的“暂停”即可暂停该条加工任务。
- 单击DSL加工任务操作列的“启动”即可启动该条加工任务。
- 单击DSL加工任务操作列的“删除”即可删除该条加工任务。



警告

删除后无法恢复，请谨慎操作。

----结束

10.2 DSL 数据加工语法（邀测）

10.2.1 操作符函数

10.2.1.1 解析函数

本文介绍User-Agent解析函数的语法规则，包括参数解释、函数示例等。

函数列表

函数	说明
ua_parse_device	解析User-Agent中的设备信息。
ua_parse_os	解析User-Agent中的操作系统信息。
ua_parse_agent	解析User-Agent中的浏览器信息。
ua_parse_all	解析User-Agent中所有信息。
url_parse	解析URL的组成部分。
url_parse_qs	解析URL中查询字符串包含的参数。

User-Agent解析函数会剔除解析结果为None的字段，例如解析的设备数据为{'brand': None, 'family': 'Other', 'model': None}，则brand字段和model字段将被剔除，最终的解析结果为{'family': 'Other'}。

ua_parse_device

解析User-Agent中的设备信息。

- **函数格式**
ua_parse_device(value)

- 参数说明

参数名称	数据类型	是否必填	说明
value	String	是	待解析的User-Agent字符串。

- 返回结果

返回JSON类型的数据集。

- 函数示例

- 测试数据

```
{
  "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36"
}
```

- 加工规则

```
e_set("new_column",ua_parse_device(v("http_user_agent")))
```

- 加工结果

```
http_user_agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
new_column: {"family": "Mac", "brand": "Apple", "model": "Mac"}
```

ua_parse_os

解析User-Agent中的操作系统信息。

- 函数格式

```
ua_parse_os(value)
```

- 参数说明

参数名称	数据类型	是否必填	说明
value	String	是	待解析的User-Agent字符串。

- 返回结果

返回JSON类型的数据集。

- 函数示例

- 测试数据

```
{
  "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36"
}
```

- 加工规则

```
e_set("new_column",ua_parse_os(v("http_user_agent")))
```

- 加工结果

```
http_user_agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
new_column: {"family": "Mac OS X",
              "major": "10",
              "minor": "9",
              "patch": "4"}
```

ua_parse_agent

解析User-Agent中的浏览器信息。

- 函数格式

ua_parse_agent(value)

- 参数说明

参数名称	数据类型	是否必填	说明
value	String	是	待解析的User-Agent字符串。

- 返回结果

返回JSON类型的数据集。

- 函数示例

- 测试数据

```
{
  "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36"
}
```

- 加工规则

```
e_set("new_column",ua_parse_agent(v("http_user_agent")))
```

- 加工结果

```
http_user_agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
new_column: { 'family': 'Chrome', 'major': '192', 'minor': '168', 'patch': '0' }
```

ua_parse_all

解析User-Agent中的所有信息。

- 函数格式

ua_parse_all(value)

- 参数说明

参数名称	数据类型	是否必填	说明
value	String	是	待解析的User-Agent字符串。

- 返回结果

返回JSON类型的数据集。

- 函数示例

- 测试数据

```
{
  "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36"
}
```

- 加工规则

```
e_set("new_column",ua_parse_all(v("http_user_agent")))
```

- 加工结果

```
http_user_agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
```

```
new_column: {
  "user_agent": {
    "family": "Chrome",
    "major": "192",
    "minor": "168",
    "patch": "0"
  },
  "os": {
    "family": "Mac OS X",
    "major": "10",
    "minor": "9",
    "patch": "4"
  },
  "device": {
    "family": "Mac",
    "brand": "Apple",
    "model": "Mac"
  }
}
```

url_parse

解析URL的组成部分。

- 函数格式
url_parse(url, scheme="", allow_fragments=true)
- 参数说明

参数名称	数据类型	是否必填	说明
value	String	是	待解析的URL。
scheme	String	否	网络协议，默认为空字符。仅在URL中未指定网络协议时，返回结果中的scheme字段才会使用此处设置的值。
allow_fragments	Boolean	否	是否解析URL中的fragment部分。 - true（默认值）：解析URL中的fragment部分，返回结果中的fragment字段为具体值。 - false：不解析URL中的fragment部分，返回结果中的fragment字段为空字符串。

- 返回结果
返回解析后的JSON数据，具体参数说明如下表所示。

字段	说明
scheme	网络协议
netloc	网络位置
path	分层路径标识
query	查询组件

fragment	片段标识符

● 函数示例

- a. 示例1：使用默认参数，返回URL的各个组成部分。
- 测试数据

```
{
  "content": "https://username:username@example.com:8083/hello/asdah/?type=docx?filename=python3.docx#urlib"
}
```

■ 加工规则

```
e_set("url",url_parse(v("content")))
```

■ 加工结果

```
content:https://username:username@example.com:8083/hello/asdah/?type=docx?filename=python3.docx#urlib
url:{
  "scheme": "https",
  "netloc": "username:username@example.com:8083",
  "path": "/hello/asdah/",
  "params": "type=docx",
  "query": "filename=python3.docx",
  "fragment": "urlib",
}
```
- b. 示例2：设置allow_fragments为false，返回结果中的fragment参数值为空。
- 测试数据

```
{
  "content": "https://username:username@example.com:8083/hello/asdah/?type=docx?filename=python3.docx#urlib"
}
```

■ 加工规则

```
e_set("url",url_parse(v("content"),allow_fragments=false))
```

■ 加工结果

```
content:https://username:username@example.com:8083/hello/asdah/?type=docx?filename=python3.docx#urlib
url:{
  "scheme": "https",
  "netloc": "username:username@example.com:8083",
  "path": "/hello/asdah/",
  "params": "type=docx",
  "query": "filename=python3.docx",
  "fragment": "",
}
```
- c. 示例3：设置scheme为https，allow_fragments为false，返回结果中scheme参数值为https，fragment参数值为空。
- 测试数据

```
{
  "content": "://username:username@example.com:8083/hello/asdah/?type=docx?filename=python3.docx#urlib"
}
```

■ 加工规则

```
e_set("url",url_parse(v("content"),scheme="https", allow_fragments=false))
```

■ 加工结果

```
content://username:username@example.com:8083/hello/asdah/?type=docx?filename=python3.docx#urlib
```



```
url:{
  "scheme": "https",
  "netloc": "username:username@example.com:8083",
  "path": "/hello/asdah/",
  "params": "type=docx",
  "query": "filename=python3.docx",
  "fragment": "",
}
```

url_parse_qs

解析URL中查询字符串的组成部分。

函数格式

```
url_parse_qs(
    url_qs,
    keep_blank_values=false,
    strict_parsing=false,
    encoding="utf-8",
    errors="replace",
    ignore_multi_fields=true,
)
```

参数说明

参数名称	数据类型	是否必填	说明
url_qs	String	是	待解析的URL查询字符串。
keep_blank_values	Boolean	否	是否返回值为空的参数。 - false（默认值）：不返回。 - true：返回，且将空值处理为空字符串。
strict_parsing	Boolean	否	是否处理解析错误。 - true：解析报错会引发 ValueError 异常。 - false（默认值）：忽略错误。
encoding	String	否	指定编码方式，将含有百分号（%）的转义字符解析为 Unicode 字符，默认为 utf-8。支持 ASCII。
errors	String	否	按照编码方式无法识别字符时的处理方案。取值包括： - ignore：直接忽略。 - strict：直接报错，丢弃此条日志数据。 - replace（默认值）：使用半角问号（?）替换无法识别部分。 - xmlcharrefreplace：使用对应 XML 字符替换无法识别部分。

ignore_multi_fields	Num	否	指定单个返回参数的值的个数。 - true（默认值）：每个参数只返回第一个值，类型为String。 - false：每个参数都返回所有值，类型为List。

- 返回结果**
返回解析后的JSON数据，具体参数说明如下表所示。

字段	说明
logType	日志类型
uid	日志的唯一标识
time	日志的时间
msg	日志中的信息

- 函数示例**
 - a. 示例1：设置keep_blank_values为true，返回结果中包含值为空的参数。

- 测试数据

```
{
  "content": "logType=net_wheel_log&uid=62452****&vid=6.1.0_gf_pc&asb=1206427&git=&time=22-11-3+
%e4%b8%8a11%e6%97%b649%e5%88%8633%e7%a7%92&operatingSystem=Windows
+10++(10.0.0)+64bit&deviceModel=System+Product+Name+(System
+manufacturer)&graphicsDeviceName=NVIDIA+GeForce+GTX
+1650&graphicsDeviceType=Direct3D11&graphicsDeviceVendor=NVIDIA&graphicsDeviceVersion=Direct3D+11.0+%5blevel
+11.1%5d&graphicsMemorySize=3962&systemMemorySize=8127&processorCount=6&processorFrequency=3000&processorType=Intel(R)+Core(TM)+i5-9500F+CPU+
%40+3.00GHz&deviceId=96da5902a042a5f84118995f88373f73650e76be166589726****&guessUID=62452****&networkReachability=wifi&msg=GetAuthkeyRsp"
}
```
 - 加工规则

```
e_set("url",url_parse_qs(v("content"), keep_blank_values=true))
```
 - 加工结果

```
content:logType=net_wheel_log&uid=62452****&vid=6.1.0_gf_pc&asb=1206427&git=&time=22-11-3+
%e4%b8%8a11%e6%97%b649%e5%88%8633%e7%a7%92&operatingSystem=Windows
+10++(10.0.0)+64bit&deviceModel=System+Product+Name+(System
+manufacturer)&graphicsDeviceName=NVIDIA+GeForce+GTX
+1650&graphicsDeviceType=Direct3D11&graphicsDeviceVendor=NVIDIA&graphicsDeviceVersion=Direct3D+11.0+%5blevel
+11.1%5d&graphicsMemorySize=3962&systemMemorySize=8127&processorCount=6&processorFrequency=3000&processorType=Intel(R)+Core(TM)+i5-9500F+CPU+
%40+3.00GHz&deviceId=96da5902a042a5f84118995f88373f73650e76be166589726****&guessUID=62452****&networkReachability=wifi&msg=GetAuthkeyRsp
url:{
  "logType": "net_wheel_log",
  "uid": "62452****",
  "vid": "6.1.0_gf_pc",
  "asb": "1206427",
  "git": "",
}
```

```
"time": "22-11-3 上11时49分33秒",
"operatingSystem": "Windows 10 (10.0.0) 64bit",
"deviceModel": "System Product Name (System manufacturer)",
"graphicsDeviceName": "NVIDIA GeForce GTX 1650",
"graphicsDeviceType": "Direct3D11",
"graphicsDeviceVendor": "NVIDIA",
"graphicsDeviceVersion": "Direct3D 11.0 [level 11.1]",
"graphicsMemorySize": "3962",
"systemMemorySize": "8127",
"processorCount": "6",
"processorFrequency": "3000",
"processorType": "Intel(R) Core(TM) i5-9500F CPU @ 3.00GHz",
"deviceId": "96da5902a042a5f84118995f88373f73650e76be166589726****",
"guessUID": "62452****",
"networkReachability": "wifi",
"msg": "GetAuthkeyRsp",
}
```

- b. 示例2: 设置keep_blank_values为默认值 (false) , 返回结果无值为空的参数。

■ 测试数据

```
{
  "content": "logType=net_wheel_log&uid=62452****&vid=6.1.0_gf_pc&asb=1206427&git=&time=22-11-3+
%e4%b8%8a11%e6%97%b649%e5%88%8633%e7%a7%92&operatingSystem=Windows
+10++(10.0.0)+64bit&deviceModel=System+Product+Name+(System
+manufacturer)&graphicsDeviceName=NVIDIA+GeForce+GTX
+1650&graphicsDeviceType=Direct3D11&graphicsDeviceVendor=NVIDIA&graphicsDeviceVersion=Direct3D+11.0+%5blevel
+11.1.%5d&graphicsMemorySize=3962&systemMemorySize=8127&processorCount=6&processorFrequency=3000&processorType=Intel(R)+Core(TM)+i5-9500F+CPU+
%40+3.00GHz&deviceId=96da5902a042a5f84118995f88373f73650e76be166589726****&guessUID=62452****&networkReachability=wifi&msg=GetAuthkeyRsp"
}
```

■ 加工规则

```
rule_set("url",url_parse_qs(v("content")))
```

■ 加工结果

```
content:logType=net_wheel_log&uid=62452****&vid=6.1.0_gf_pc&asb=1206427&git=&time=22-11-3+
%e4%b8%8a11%e6%97%b649%e5%88%8633%e7%a7%92&operatingSystem=Windows
+10++(10.0.0)+64bit&deviceModel=System+Product+Name+(System
+manufacturer)&graphicsDeviceName=NVIDIA+GeForce+GTX
+1650&graphicsDeviceType=Direct3D11&graphicsDeviceVendor=NVIDIA&graphicsDeviceVersion=Direct3D+11.0+%5blevel
+11.1.%5d&graphicsMemorySize=3962&systemMemorySize=8127&processorCount=6&processorFrequency=3000&processorType=Intel(R)+Core(TM)+i5-9500F+CPU+
%40+3.00GHz&deviceId=96da5902a042a5f84118995f88373f73650e76be166589726****&guessUID=62452****&networkReachability=wifi&msg=GetAuthkeyRsp
url:{
  "logType": "net_wheel_log",
  "uid": "62452****",
  "vid": "6.1.0_gf_pc",
  "asb": "1206427",
  "time": "22-11-3 上11时49分33秒",
  "operatingSystem": "Windows 10 (10.0.0) 64bit",
  "deviceModel": "System Product Name (System manufacturer)",
  "graphicsDeviceName": "NVIDIA GeForce GTX 1650",
  "graphicsDeviceType": "Direct3D11",
  "graphicsDeviceVendor": "NVIDIA",
  "graphicsDeviceVersion": "Direct3D 11.0 [level 11.1]",
  "graphicsMemorySize": "3962",
  "systemMemorySize": "8127",
  "processorCount": "6",
  "processorFrequency": "3000",
  "processorType": "Intel(R) Core(TM) i5-9500F CPU @ 3.00GHz",
}
```

```
"deviceId": "96da5902a042a5f84118995f88373f73650e76be166589726****",
"guessUID": "62452****",
"networkReachability": "wifi",
"msg": "GetAuthkeyRsp",
}
```

c. 示例3：设置ignore_multi_fields为默认值（true），每个参数只返回第一个值。

■ 测试数据

```
{
  "content": "logType=net_log&uid=62452****&x=1&x=2&x=3&asb=123&asb=456"
}
```

■ 加工规则

```
e_set("url",url_parse_qs(v("content")))
```

■ 加工结果

```
content:logType=net_log&uid=62452****&x=1&x=2&x=3&asb=123&asb=456
url:{
  "logType": "net_log",
  "uid": "62452****",
  "x": "1",
  "asb": "123"
}
```

d. 示例4：设置ignore_multi_fields为false，每个参数返回所有值。

■ 测试数据

```
{
  "content": "logType=net_log&uid=62452****&x=1&x=2&x=3&asb=123&asb=456"
}
```

■ 加工规则

```
e_set("url",url_parse_qs(v("content"),ignore_multi_fields=false))
```

■ 加工结果

```
content:logType=net_log&uid=62452****&x=1&x=2&x=3&asb=123&asb=456
url:{
  "logType": ["net_log"],
  "uid": ["62452****"],
  "x": ["1", "2", "3"],
  "asb": ["123", "456"],
}
```

10.2.1.2 资源函数

本文介绍资源函数的语法规则，包括参数解释、函数示例等。

函数列表

使用如下资源函数时，必须配置高级预览才能拉取到目标数据。

函数	说明
res_local	从当前数据加工任务中获取高级配置中的密钥信息。 支持和其他函数组合使用。

函数	说明
res_obs_file	从OBS中获取特定Bucket下的文件内容，支持定期更新数据。 支持和其他函数组合使用。

res_local

使用res_local函数从当前数据加工任务获取高级配置中的密钥信息。支持和其他函数组合使用。

- 函数格式**
res_local(param, default=None, type="auto")

- 参数说明**

参数名称	数据类型	是否必填	说明
param	String	是	对应高级参数配置中的Key。
default	String	否	当param参数的值不存在时，返回该参数的值，默认值为None。
type	String	否	数据输出时的数据格式。 - auto（默认值）：将原始值转化为JSON格式。如果转换失败则返回原始值。 - JSON：将原始值转化为JSON格式。如果转换失败则返回default参数的值。 - raw：返回原始值。

- 返回结果**
根据参数配置，返回JSON格式数据或者原始值。

表 10-2 成功示例

原始值	返回值	返回值类型
1	1	整数
1.2	1.2	浮点
true	true	布尔
false	false	布尔
"123"	123	字符串
null	None	None

原始值	返回值	返回值类型
["v1", "v2", "v3"]	["v1", "v2", "v3"]	列表
["v1", 3, 4.0]	["v1", 3, 4.0]	列表
{"v1": 100, "v2": "good"}	{"v1": 100, "v2": "good"}	字典
{"v1": {"v11": 100, "v2": 200}, "v3": "good"}	{"v1": {"v11": 100, "v2": 200}, "v3": "good"}	字典

表 10-3 失败示例

原始值	返回值	说明
(1,2,3)	"(1,2,3)"	不支持元组，需使用列表形式。
true	"true"	只支持true、false（小写）这两种布尔类型。
{1: 2, 3: 4}	"{1: 2, 3: 4}"	字典的关键字只能是字符串。

- 函数示例**
从高级配置的密钥参数中获取信息并赋值给local。添加密钥时的Key为endpoint，Value为hangzhou。
 - 测试数据

```
{
  "content": "1"
}
```
 - 加工规则

```
e_set("local", res_local('endpoint'))
```
 - 加工结果

```
content: 1
local: hangzhou
```

res_obs_file

使用res_obs_file函数从OBS Bucket中获取文件内容，并支持定期刷新。支持和其他函数组合使用。

建议日志服务Project和OBS Bucket处于同一地域，使用华为云内网获取数据。

- 函数格式**
res_obs_file(endpoint,bucket,file)
- 参数说明**

参数名称	数据类型	是否必填	说明
endpoint	String	是	OBS中保存的文件地址。如obs.cn-north-4.huawei.com

bucket	String	是	OBS中使用的桶名，如lts-dsl。
file	String	是	目标OBS文件的路径。例如test/data.txt，不能以正斜线（/）开头。

- **返回结果**

返回字节流形式或文本形式的文件数据。

- **函数示例**

a. 示例1：从OBS中拉取JSON格式的数据。

- **JSON内容**

```
{
  "users": [
    {
      "name": "user1",
      "login_historys": [
        {
          "date": "2019-10-10 0:0:0",
          "login_ip": "203.0.113.10"
        },
        {
          "date": "2019-10-10 1:0:0",
          "login_ip": "203.0.113.10"
        }
      ]
    },
    {
      "name": "user2",
      "login_historys": [
        {
          "date": "2019-10-11 0:0:0",
          "login_ip": "203.0.113.20"
        },
        {
          "date": "2019-10-11 1:0:0",
          "login_ip": "203.0.113.30"
        },
        {
          "date": "2019-10-11 1:1:0",
          "login_ip": "203.0.113.50"
        }
      ]
    }
  ]
}
```

- **测试数据**

```
{
  "content": "123"
}
```

- **加工规则**

```
e_set(
  "json_parse",
  json_parse(
    res_obs_file(
      "https://obs.test.huawei.com",
      "lts-dsl",
      "als_dsl.json"
    )
  ),
)
```

■ 加工结果

```
content: 123
json_parse:
{'
  "users": [
    {
      "name": "user1",
      "login_historys": [
        {
          "date": "2019-10-10 0:0:0",
          "login_ip": "203.0.113.10"
        },
        {
          "date": "2019-10-10 1:0:0",
          "login_ip": "203.0.113.10"
        }
      ]
    },
    {
      "name": "user2",
      "login_historys": [
        {
          "date": "2019-10-11 0:0:0",
          "login_ip": "203.0.113.20"
        },
        {
          "date": "2019-10-11 1:0:0",
          "login_ip": "203.0.113.30"
        },
        {
          "date": "2019-10-11 1:1:0",
          "login_ip": "203.0.113.50"
        }
      ]
    }
  ]
}
```

b. 示例2：从OBS中拉取文本内容。

■ 文本内容

```
Test bytes
```

■ 测试数据

```
{
  "content": "123"
}
```

■ 加工规则

```
e_set(
  "test_txt",
  res_obs_file(
    "https://obs.test.huawei.com",
    "lts-dsl",
    "als_dsl.json"
  )
)
```

■ 加工结果

```
content: 123
test_txt: Test bytes
```

10.2.1.3 字典函数

本文介绍字典函数的语法规则，包括参数解释、函数示例等。

函数列表

函数	说明
<code>dct_make</code>	构建字典。
<code>dct_update</code>	更新字典。
<code>dct_delete</code>	删除字典值。
<code>dct_keys</code>	获取字典关键字列表。
<code>dct_values</code>	获取字典值列表。
<code>dct_get</code>	获取字典中某关键字的值。

dct_make

构建字典。

- 函数格式

```
dct_make(key1, value1, key2, value2, ...)
```

- 参数说明

参数名称	数据类型	是否必填	说明
key	String	是	作为字典key的字符串。
value	String	是	作为字典value的字符串。

- 返回结果

返回构建的字典。

- 函数示例

- 测试数据

```
{
  "content": "test"
}
```

- 加工规则

```
e_set("hello", dct_make("k1", "v1", "k2", "v2"))
```

- 加工结果

```
content:test
hello:{"k1": "v1", "k2": "v2"}
```

dct_update

更新字典。

- 函数格式

```
dct_update(dict1, dict2)
```

- 参数说明

参数名称	数据类型	是否必填	说明
dict1	dict	是	需要被更新的目标字典。
dict2	dict	是	补充新的字典信息。

- **返回结果**

返回更新的字典。

- **函数示例**

- 测试数据

```
{
  "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
}
```

- 加工规则

```
e_set("hello", dct_update(v("ctx"), {"k3": "v3"}))
```

- 加工结果

```
ctx: {"k1":"v1","k2":"v2"}
hello: {"k1": "v1", "k2": "v2", "k3": "v3"}
```

dct_delete

删除字典。

- **函数格式**

```
dct_delete(dict, key1, key2, ...)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
dict	dict	是	需要删除键值对的目标字典。
key1	String	是	要删除的键值对的关键字。
key2	String	否	要删除的键值对的关键字。

- **返回结果**

返回删除后的字典。

- **函数示例**

- 测试数据

```
{
  "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
}
```

- 加工规则

```
e_set("hello", dct_delete(v("ctx"), "k2"))
```

- 加工结果

```
ctx: {"k1":"v1","k2":"v2"}
hello: {"k1":"v1"}
```

dct_keys

获取字典关键字列表。

- 函数格式

dct_keys(dict)

- 参数说明

参数名称	数据类型	是否必填	说明
dict	dict	是	字典数据。

- 返回结果

返回获取的字典关键词列表。

- 函数示例

- 测试数据

```
{
  "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
}
```

- 加工规则

```
e_set("hello", dct_keys(v("ctx")))
```

- 加工结果

```
ctx: {"k1":"v1","k2":"v2"}
hello: ["k1","k2"]
```

dct_values

获取字典值列表。

- 函数格式

dct_values(dict)

- 参数说明

参数名称	数据类型	是否必填	说明
dict	dict	是	字典数据。

- 返回结果

返回获取的字典词列表。

- 函数示例

- 测试数据

```
{
  "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
}
```

- 加工规则

```
e_set("hello", dct_values(v("ctx")))
```

- 加工结果

```
ctx: {"k1":"v1","k2":"v2"}
hello: ["v1","v2"]
```

dct_get

获取字典中某关键字的值。

- **函数格式**
dct_get(dict,key,default=None)
- **参数说明**

参数名称	数据类型	是否必填	说明
dict	dict	是	字典数据。
key	String	是	要获取值的关键字。
default	String	否	key不存在时，返回该值。
- **返回结果**
返回字典关键字的值。
- **函数示例**
 - a. 示例1：
 - 测试数据

```
{
  "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
}
```
 - 加工规则

```
e_set("hello", dct_get(v("ctx"), "k1"))
```
 - 加工结果

```
ctx: {"k1":"v1","k2":"v2"}
hello: v1
```
 - b. 示例2：
 - 测试数据

```
{
  "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
}
```
 - 加工规则

```
e_set("hello", dct_get(v("ctx"), "k3",default="123"))
```
 - 加工结果

```
ctx: {"k1":"v1","k2":"v2"}
hello: 123
```

10.2.1.4 列表函数

本文介绍列表函数的语法规则，包括参数解释、函数示例等。

函数列表

函数	说明
lst_make	构建列表。
lst_insert	在列表特定位置插入元素。
lst_append	在列表结尾追加元素。

函数	说明
<code>lst_delete_at</code>	在列表特定位置删除元素。
<code>lst_reverse</code>	反向排序列表。
<code>lst_get</code>	获取列表、元组中的一个元素。

lst_make

构建一个列表。

- 函数格式

```
lst_make(value1, value2, ...)
```

- 参数说明

参数名称	数据类型	是否必填	说明
value1	String	是	列表的元素。
value2	String	是	列表的元素。

- 返回结果

返回构建后的列表。

- 函数示例

- 测试数据

```
{
  "content": "test"
}
```

- 加工规则

```
e_set("hello", lst_make("k1", "k2"))
```

- 加工结果

```
content:test
hello:["k1", "k2"]
```

lst_insert

在列表特定位置插入元素。

- 函数格式

```
lst_insert(list_string, location, value1, value2, ...)
```

- 参数说明

参数名称	数据类型	是否必填	说明
list_string	List	是	传入一个列表。
location	Number	是	要插入的位置。
value1	String	是	要插入的元素。

value2	String	否	要插入的元素。

- **返回结果**

返回插入元素后的列表。

- **函数示例**

- 测试数据

```
{
  "ctx": ["k1", "k2"]
}
```

- 加工规则

```
e_set("hello", lst_insert(v("ctx"), 0, "k0"))
```

- 加工结果

```
ctx: ["k1", "k2"]
hello: ["k0", "k1", "k2"]
```

lst_append

在列表结尾追加元素。

- **函数格式**

```
lst_append(list_string, value1, value2, ...)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
list_string	List	是	传入一个列表。
value1	String	是	要添加的元素。
value2	String	否	要添加的元素。

- **返回结果**

返回添加元素后的列表。

- **函数示例**

- 测试数据

```
{
  "ctx": [
    "k1",
    "k2"
  ]
}
```

- 加工规则

```
e_set("hello", lst_append(v("ctx"), "k3"))
```

- 加工结果

```
ctx: ["k1", "k2"]
hello: ["k1", "k2", "k3"]
```

lst_delete_at

返回删除元素后的列表。

- **函数格式**

```
lst_delete_at(list_string, location)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
list_string	list	是	传入一个列表。
location	Number	是	要删除元素的位置。第一个元素从位置0开始。

- **返回结果**

返回获取的字典关键词列表。

- **函数示例**

- 测试数据

```
{  
  "ctx": ["k1", "k2"]  
}
```

- 加工规则

```
e_set("hello", lst_delete_at(v("ctx"),1))
```

- 加工结果

```
ctx: ["k1", "k2"]  
hello: ["k1"]
```

lst_reverse

反向排序列表。

- **函数格式**

```
lst_reverse(list_string)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
list_string	List	是	传入一个列表。

- **返回结果**

返回反转后的列表。

- **函数示例**

- 测试数据

```
{  
  "ctx": ["v1", "v2"]  
}
```

- 加工规则

```
e_set("hello", lst_reverse(v("ctx")))
```

- 加工结果

```
ctx: ["v1", "v2"]  
hello: ["v2", "v1"]
```

lst_get

获取列表、元组中的一个元素。

- 函数格式**
lst_get(list_string, location)
- 参数说明**

参数名称	数据类型	是否必填	说明
ist_string	List	是	传入一个列表。
location	Int	是	从0开始计数，传入数字。例如数据为["a","b","c"]，对应获取的元素位置分别为0，1，2。
- 返回结果**
返回列表的其中一个元素。
- 函数示例**
 - 测试数据

```
{
  "ctx":["v1","v2"]
}
```
 - 加工规则

```
e_set("hello", lst_get(v("ctx"),1))
```
 - 加工结果

```
ctx: ["v1","v2"]
hello: "v2"
```

10.2.1.5 编码解码函数

本文介绍编码解码函数的语法规则，包括参数解释、函数示例等。

函数列表

分类	类型	函数	说明
编码与解码	字符串类型	str_encode	对数据进行编码。
		str_decode	对数据进行解码。
	Base64类型	base64_encoding	对数据进行Base64编码。
		base64_decoding	对数据进行Base64解码。
	HTML类型	html_encoding	对数据进行HTML编码。
		html_decoding	对数据进行HTML解码。

分类	类型	函数	说明
	URL类型	url_encoding	对数据进行URL编码。
		url_decoding	对数据进行URL解码。
压缩与解压缩	Gzip压缩库	gzip_compress	对数据进行压缩并编码。
		gzip_decompress	将压缩数据解压缩。
	Zlib压缩库	zlib_compress	对数据进行压缩并编码。
		zlib_decompress	将压缩数据解压缩。
哈希摘要	MD5哈希	md5_encoding	对数据进行MD5编码。
	SHA1哈希	sha1_encoding	对数据进行SHA1编码。
	循环冗余校验	crc32_encoding	计算数据的循环冗余校验码。

str_encode

按照指定的编码格式对字符串进行编码。

- 函数格式
`str_encode(value, "utf8", errors="ignore")`
- 参数说明

参数名称	数据类型	是否必填	说明
value	任意（自动转为String）	是	需要被编码的值。
encoding	String	否	编码格式，默认为utf8。支持ASCII。
errors	String	否	按照编码格式无法识别字符时的处理方式。取值包括： - ignore（默认值）：忽略不做编码。 - strict：直接报错，丢弃此条日志数据。 - replace：使用半角问号（?）替换无法识别部分。 - xmlcharrefreplace：使用对应XML字符替换无法识别部分。

- **返回结果**

返回编码后的字符串。

- **函数示例**

- a. 示例1:

- 测试数据

```
{  
  "value": "test 测试"  
}
```

- 加工规则

```
e_set("result", str_encode(v("value"), "ascii", errors="ignore"))
```

- 加工结果

```
value: test 测试  
result: test
```

- b. 示例2:

- 测试数据

```
{  
  "value": "test 测试"  
}
```

- 加工规则

```
e_set("result", str_encode(v("value"), "ascii", errors="strict"))
```

- 加工结果：执行时直接报错

- c. 示例3:

- 测试数据

```
{  
  "value": "test 测试"  
}
```

- 加工规则

```
e_set("result", str_encode(v("value"), "ascii", errors="replace"))
```

- 加工结果

```
value: test 测试  
result: test ??
```

- d. 示例4:

- 测试数据

```
{  
  "value": "test 测试"  
}
```

- 加工规则

```
e_set("result", str_encode(v("value"), "ascii", errors="xmlcharrefreplace"))
```

- 加工结果

```
value: test 测试  
result: test &#27979;&#35797;
```

str_decode

按照指定的编码格式对传入值进行解码。

- **函数格式**

```
str_decode(value, "utf8", errors="ignore")
```

- **参数说明**

参数名称	数据类型	是否必填	说明
value	任意（自动转为String）	是	需要被解码的值。
encoding	任意（自动转为String）	否	编码格式，默认为utf8。支持ASCII。
errors	任意（自动转为String）	否	按照编码格式无法识别字符时的处理方式。取值包括： • ignore（默认值）：忽略不做解码。 • strict：直接报错，丢弃此条日志数据。 • replace：使用半角问号（?）替换无法编解码部分。 • xmlcharrefreplace：使用对应XML字符替换无法编解码部分。

- **返回结果**

返回解码后的值。

- **函数示例**

- 测试数据

```
{
  "test": "asewds"
}
```

- 加工规则

```
e_set("encoding", str_decode(b'\xe4\xbd\xa0\xe5\xa5\xbd', "utf8", 'strict'))
```

- 加工结果

```
test: asewds
encoding: 您好
```

base64_encoding

对数据进行Base64编码。

- **函数格式**

```
base64_encoding(value, format=None)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	填入需要被编码的值。
format	String	否	Base64编码协议。默认为format=RFC3548，还可以配置为format=RFC4648。

- **返回结果**

返回编码后的字符串。

- **函数示例**

- 测试数据

```
{
  "str_en": "data to be encoded"
}
```

- 加工规则

```
e_set("str_base64",base64_encoding(v("str_en")))
```

- 加工结果

```
str_en : data to be encoded
str_base64 : ZGF0YSB0byBiZSBpbmNvZGVk
```

base64_decoding

对数据进行Base64解码。

- **函数格式**

```
base64_decoding(value, format=None)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	被解码的值。
format	String	否	Base64解码协议。默认为format=RFC3548，还可以配置为format=RFC4648。 RFC4648的Base64解码协议使用等号(=)将被解码的值填充到4字节的倍数。

- **返回结果**

返回解码后的字符串。

- **函数示例**

- 测试数据

```
{
  "str_de": "ZGF0YSB0byBiZSBpbmNvZGVk"
}
```

- 加工规则

```
e_set("str_de_base64",base64_decoding(v("str_de")))
```

- 加工结果

```
str_de: ZGF0YSB0byBiZSBpbmNvZGVk
str_de_base64: data to be encoded
```

html_encoding

对数据进行HTML编码。

- **函数格式**

```
html_encoding(value)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	被编码的值。

- **返回结果**

返回编码后字符串。

- **函数示例**

- 测试数据

```
{
  "str": "<img class='size-medium wp-image-113' style='margin-left: 15px;' title='su1' src='http://aadoc.com/wp-content/uploads/2008/10/su1-300x194.jpg' alt='\" width='300' height='194' />"
}
```
- 加工规则

```
e_set("str_html_en",html_encoding(v("str")))
```
- 加工结果

```
str : 
str_html_en : &lt;img class=&quot;size-medium wp-image-113&quot; style=&quot;margin-left: 15px;&quot; title=&quot;su1&quot; src=&quot;http://aadoc.com/wp-content/uploads/2008/10/su1-300x194.jpg&quot; alt=&quot;&quot; width=&quot;300&quot; height=&quot;194&quot; /&gt;
```

html_decoding

对数据进行HTML解码。

- **函数格式**

html_decoding(value)

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	被解码的值。

- **返回结果**

返回解码后字符串。

- **函数示例**

- 测试数据

```
{
  "str": "&lt;img class=&quot;size-medium wp-image-113&quot; style=&quot;margin-left: 15px;&quot; title=&quot;su1&quot; src=&quot;http://aadoc.com/wp-content/uploads/2008/10/su1-300x194.jpg&quot; alt=&quot;&quot; width=&quot;300&quot; height=&quot;194&quot; /&gt;"
}
```
- 加工规则

```
e_set("str_html_de",html_decoding(v("str")))
```
- 加工结果

```
str : 
str_html_de: 
```

url_encoding

对数据进行URL编码。

- 函数格式
url_encoding(value, plus=false)
- 参数说明

参数名称	参数类型	是否必填	说明
value	String	是	被编码的值。
plus	Boolean	否	是否将空格转换为加号，默认为false。 - true：将空格转换为加号。 - false：不转换空格为加号。
- 返回结果
返回编码后的字符串。
- 函数示例
 - 示例1：对数据进行URL编码。
 - 测试数据

```
{
  "content": "https://www.example.org/hello/asdah"
}
```
 - 加工规则

```
e_set("url",url_encoding(v("content")))
```
 - 加工结果

```
content : https://www.example.org/hello/asdah
url: https%3A%2F%www.example.org%2FHello%2Fasdah
```
 - 示例2：对数据进行URL编码。其中，未设置plus参数，不转换数据中的空格为加号。
 - 测试数据

```
{
  "content": "1 2+3:4"
}
```
 - 加工规则

```
e_set("URL",url_encoding(v("content")))
```
 - 加工结果

```
content:1 2+3:4
URL:1%202%2B3%3A4
```
 - 示例3：对数据进行URL编码。其中，设置plus参数为true，将数据中的空格转换为加号。
 - 测试数据

```
{
  "content": "1 2+3:4"
}
```

- 加工规则
e_set("URL", url_encoding(v("content"), plus=true))
- 加工结果
content:1 2+3:4
URL:1+2%2B3%3A4

url_decoding

对数据进行URL解码。

- 函数格式
url_decoding(value, plus=false)
- 参数说明

参数名称	参数类型	是否必填	说明
value	String	是	被解码的值。
plus	Boolean	否	是否将加号转换为空格，默认为false。 - true：将加号转换为空格。 - false：不转换加号为空格。

- 返回结果
返回解码后的字符串。
- 函数示例

a. 示例1：对数据进行URL解码。

- 测试数据

```
{
  "content": "https%3A%2F%www.example.org%2FHello%2Fasdah"
}
```

- 加工规则
e_set("URL",url_decoding(v("content")))

- 加工结果
content : https%3A%2F%www.example.org%2FHello%2Fasdah
URL : https://www.example.org/hello/asdah

b. 示例2：对数据进行URL解码。其中，未设置plus参数，不转换数据中的加号为空格。

- 测试数据

```
{
  "content": "/answer?event_date=2022-06-30+09%3A06%3A53%20123"
}
```

- 加工规则
e_set("URL",url_decoding(v("content")))

- 加工结果

```
content:/answer?event_date=2022-06-30+09%3A06%3A53%20123
URL:/answer?event_date=2022-06-30+09:06:53 123
```

- c. 示例3：对数据进行URL解码。其中，设置plus参数为true，将数据中的加号转换为空格。

■ 测试数据

```
{
  "content": "/answer?event_date=2022-06-30+09%3A06%3A53%20123"
}
```

■ 加工规则

```
e_set("URL",url_decoding(v("content"),plus=true))
```

■ 加工结果

```
content:/answer?event_date=2022-06-30+09%3A06%3A53%20123
URL:/answer?event_date=2022-06-30 09:06:53 123
```

gzip_compress

将数据进行压缩并编码。

- 函数格式

```
gzip_compress(data, compresslevel=6, to_format="base64", encoding="utf-8")
```

- 参数说明

参数名称	参数类型	是否必填	说明
data	String	是	输入需要压缩的数据。
compresslevel	Int	否	用于控制压缩等级，可配置为0~9的整数。默认值为6。 ● 1：压缩速度最快但压缩比例最小。 ● 9：压缩速度最慢但压缩比例最大。 ● 0：不压缩。
to_format	String	否	对压缩后的数据进行编码的格式，目前支持进行Base64和hex编码。
encoding	String	否	原始未压缩数据的编码格式，默认为utf-8。

- 返回结果

返回编码后的字符串。

- 函数示例（加密结果仅供参考。）

- a. 示例1：对日志字段进行压缩并Base64编码。

■ 测试数据

```
{
  "content": "I always look forward to my holidays whether I travel or stay at home."
}
```


- 加工规则

```
e_set("base64_encode_gzip_compress",gzip_compress(v("content"),to_format="base64"))
```

- 加工结果

```
content: I always look forward to my holidays whether I travel or stay at home.
base64_encode_gzip_compress: H4sIAA8JXl4C/
xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtle/7wbES3rDyRsnoyQmklgNo1/
ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
```

b. 示例2：对日志字段进行hex编码。

- 测试数据

```
{
  "content":"H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtle/
7wbES3rDyRsnoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA"
}
```

- 加工规则

```
e_set("hex_encode_gzip_compress", gzip_compress(v("content"), to_format="hex"))
```

- 加工结果

```
content:H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtle/
7wbES3rDyRsnoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
hex_encode_gzip_compress:1f8b08004a478c6202ff0dc1dd0e43301800d047aa65156e3ff52f
4a2ba17649c43255194d4a9f9e73527c64007e2e2426e81485c35628c1c42616535079bc6405
e5d1e92ef009b59c906786a879efe1c50fb55d6c5de44cb717b2dae6d4f103f8feecbf4f88a2a4
41bae618c679575d9bc0e306907876806c000000
```

gzip_decompress

将压缩数据解压缩。

- 函数格式

```
gzip_decompress(data, from_format="base64", encoding="utf-8")
```

- 参数说明

参数名称	参数类型	是否必填	说明
data	任意	是	输入需要解压的数据。
from_format	String	否	解压数据的编码格式，目前支持Base64和hex格式解码。
encoding	String	否	原始未压缩数据的编码格式，默认utf-8，其他编码格式请参见标准编码格式。

- 返回结果

返回解压后的对象。

- 函数示例

a. 示例1：对日志字段进行Base64解码。

- 测试数据

```
{
  "content": "H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtle/
7wbES3rDyRsnoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA"
}
```

- 加工规则

e_set("gzip_decompress",gzip_decompress(v("content"),from_format="base64"))
- 加工结果

content: H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtIe/7wbES3rDyRsnoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
gzip_decompress: I always look forward to my holidays whether I travel or stay at home.
- b. 示例2：对日志字段进行hex解码。
- 测试数据

{
 "content":"1f8b0800bff8856202ff0dc1dd0e43301800d047aa65156e3ff52f4a2ba17649c43255194d4a9f9e73527c64007e2e2426e81485c35628c1c42616535079bc6405e5d1e92ef009b59c906786a879efe1c50fb55d6c5de44cb717b2dae6d4f103f8feecbf4f88a2a441bae618c679575d9bc0e306907876806c000000"
}
- 加工规则

e_set("gzip_decompress", gzip_decompress(v("content"), from_format="hex"))
- 加工结果

content:1f8b0800bff8856202ff0dc1dd0e43301800d047aa65156e3ff52f4a2ba17649c43255194d4a9f9e73527c64007e2e2426e81485c35628c1c42616535079bc6405e5d1e92ef009b59c906786a879efe1c50fb55d6c5de44cb717b2dae6d4f103f8feecbf4f88a2a441bae618c679575d9bc0e306907876806c000000 gzip_decompress:H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtIe/7wbES3rDyRsnoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA

zlib_compress

将数据进行压缩并编码。

- 函数格式
zlib_compress(data, compresslevel=6, to_format="base64", encoding="utf-8")
- 参数说明

参数名称	参数类型	是否必填	说明
data	String	是	输入需要压缩的数据。
compresslevel	Int	否	用于控制压缩等级，可配置为0~9的整数。默认值为6。 1：压缩速度最快但压缩比例最小。 9：压缩速度最慢但压缩比例最大。 0：不压缩。
to_format	String	否	对压缩后的数据进行编码的格式，目前只支持进行Base64编码。
encoding	String	否	原始未压缩数据的编码格式，默认为utf-8。

- 返回结果
返回编码后的对象。

- **函数示例**

- 测试数据

```
{
  "content": "I always look forward to my holidays whether I travel or stay at home."
}
```
- 加工规则

```
e_set("zlib_compress", zlib_compress(v("content"), to_format="base64"))
```
- 加工结果（加密结果仅供参考。）

```
zlib_compress: eJwVytEJgDAMBcBV3gTu0TECjURMeZAGQ7ZXv
+8GxEt6w8kbJ6MkJpJYDaNf87cyTdPAQIY86mBgpzQkv7T0eAGNshln
content: I always look forward to my holidays whether I travel or stay at home.
```

zlib_decompress

- **函数格式**

```
zlib_decompress(data, from_format="base64", encoding="utf-8")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
data	String	是	输入需要解压的数据。
from_format	String	否	解压数据的编码格式，目前只支持Base64编码。
encoding	String	否	编码格式，默认utf-8。

- **返回结果**

返回解压后的对象。

- **函数示例**

- 测试数据

```
{
  "content": "eJwVytEJgDAMBcBV3gTu0TECjURMeZAGQ7ZXv
+8GxEt6w8kbJ6MkJpJYDaNf87cyTdPAQIY86mBgpzQkv7T0eAGNshln"
}
```
- 加工规则

```
e_set("zlib_decompress", zlib_decompress(v("content"), from_format="base64"))
```
- 加工结果

```
content: eJwVytEJgDAMBcBV3gTu0TECjURMeZAGQ7ZXv
+8GxEt6w8kbJ6MkJpJYDaNf87cyTdPAQIY86mBgpzQkv7T0eAGNshln
zlib_decompress: I always look forward to my holidays whether I travel or stay at home.
```

md5_encoding

对数据进行MD5编码。

- **函数格式**

```
md5_encoding(value, format="hex")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	被编码的值。

format	String	否	默认值为hex，可选值：binary、hex。

- **返回结果**

返回编码后的字符串。

- **函数示例**

a. 示例1

- **测试数据**

```
{
  "str": "GeeksforGeeks"
}
```

- **加工规则**

```
e_set("str_md5_en",md5_encoding(v("str")))
```

- **加工结果**

```
str : GeeksforGeeks
str_md5_en : f1e069787ece74531d112559945c6871
```

b. 示例2

- **测试数据**

```
{
  "str": "GeeksforGeeks"
}
```

- **加工规则**

```
e_set("str_md5_en",base64_encoding(md5_encoding(v("str")), format="binary"))
```

- **加工结果**

```
str : GeeksforGeeks
str_md5_en : 8eBpeH7OdFMdESVZIFxocQ==
```

sha1_encoding

对数据进行SHA-1编码。

- **函数格式**

```
sha1_encoding(value, format=None)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	被编码的值。
format	String	否	编码类型，默认为SHA1类型，可选SHA256、SHA384、SHA224、SHA512类型。

- **返回结果**

返回编码后的字符串。

- **函数示例**

- **测试数据**

```
{
  "str": "GeeksforGeeks"
}
```

– 加工规则

```
e_set("str_sha1",sha1_encoding(v("str")))
e_set("str_sha512",sha1_encoding(v("str"),format='SHA512'))
e_set("str_sha224",sha1_encoding(v("str"),format='SHA224'))
e_set("str_sha384",sha1_encoding(v("str"),format=))
e_set("str_sha256",sha1_encoding(v("str"),format='SHA256'))
```

– 加工结果

```
str : GeeksforGeeks
str_sha1 : 4175a37afd561152fb60c305d4fa6026b7e79856
str_sha512 :
0d8fb9370a5bf7b892be4865cdf8b658a82209624e33ed71cae353b0df254a75db63d1baa35ad99f2
6f1b399c31f3c666a7fc67ecef3bdcdb7d60e8ada90b722 str_sha224 :
173994f309f727ca939bb185086cd7b36e66141c9e52ba0bdcfd145d
str_sha384 :
d1e67b8819b009ec7929933b6fc1928dd64b5df31bcde6381b9d3f90488d253240490460c0a5a1a8
73da8236c12ef9b3
str_sha256 : f6071725e7ddeb434fb6b32b8ec4a2b14dd7db0d785347b2fb48f9975126178f
```

crc32_encoding

计算数据的循环冗余校验码。

- 函数格式

```
crc32_encoding(data, input_format="raw", input_encoding="utf-8")
```

- 参数说明

参数名称	参数类型	是否必填	说明
data	String	是	计算校验码的值。
input_format	String	否	输入值的字符格式。默认为raw，可选值包括： - raw：Bytes格式 - hex：十六进制格式 - base64：Base64编码格式
input_encoding	String	否	仅当input_format取值为raw时需要配置。配置字符编码格式。默认为utf-8。

- 返回结果

返回传入数据的循环冗余校验码。

- 函数示例

a. 示例1：计算字段test的循环冗余校验码。

- 测试数据

```
{
  "test": "aatest"
}
```

- 加工规则

```
e_set("str_crc32", crc32_encoding(v("test")))
```

- 加工结果

```
str_crc32:1434103726
test:aatest
```
- b. 示例2：将字段test1和test2连接后进行MD5编码，最后计算其循环冗余校验码。
 - 测试数据

```
{
  "test1": "test1",
  "test2": "test2"
}
```
 - 加工规则

```
e_set(
  "str_crc32",
  crc32_encoding(
    md5_encoding(str_join("+", v("test1"), v("test2")), format="binary")
  ),
)
```
 - 加工结果

```
str_crc32:369733261
test1:test1
test2:test2
```
- c. 示例3：计算字段test的循环冗余校验码，其值为Base64编码格式。
 - 测试数据

```
{
  "test": "Taloz+e+PzP3NltrEXiCig=="
}
```
 - 加工规则

```
e_set("str_crc32", crc32_encoding(v("test"), input_format="base64"))
```
 - 加工结果

```
str_crc32:1093789404
test:Taloz+e+PzP3NltrEXiCig==
```

10.2.1.6 IP 解析函数

本文介绍IP解析函数的语法规则，包括参数解释、函数示例等。

函数	说明
geo_parse	根据IP地址解析出所属国家、省份和市信息
ip_cidrmatch	判断IP地址是否属于CIDR地址块
ip_version	判断IP地址为IPv4还是IPv6
ip_type	判断IP地址为私有地址还是公有地址
ip_makenet	将单个IP地址转换为CIDR地址块
ip_makenet	将输入的CIDR地址块按照Prefixlen或者Netmask格式输出
ip_overlaps	判断两个网段是否存在重叠
ip2long	将字符串格式的IP地址转换成长整型数据

函数	说明
<code>long2ip</code>	将长整型数据转换成字符串格式的IP地址

geo_parse

根据IP地址解析出所属国家、省份和城市等信息。

- 函数格式
`geo_parse(ip, keep_fields=None, ip_sep=None)`

- 参数说明

参数名称	参数类型	是否必填	说明
ip	String	是	IP地址，表示解析该IP地址所属国家、省分和市信息。如果包含多个IP地址，可通过ip_sep参数指定分割符。
keep_fields	Tuple	否	返回结果中包含的key信息。支持的key如下列表： - city：城市名称。 - province：省份名称。 - country：国家名称。 - isp：所属网络运营商名称。 - lat：IP地址所在位置的纬度。 - lon：IP地址所在位置的经度。 例如keep_fields=("city","country")表示仅输出city和country字段信息。 此外keep_fields也支持重命名。例如(("city","cty"),("country","state"))表示以cty和state形式输出。 注意：keep_fields参数为tuple类型，如果只有一个参数，需要添加逗号，例如("city",)或者(("city","cty"),)
ip_sep	String	否	IP地址分隔符，用于将包含多个IP地址的字符串分割为多个IP地址，解析结果通过JSON格式返回。默认值为None，表示不进行分隔。

- 返回结果
返回字典形式数据。
- 函数示例
 - 单个IP查询。
 - 测试数据

```
{
  "ip": "192.168.0.1"
}
```

- 加工规则
`e_set("geo_parse", geo_parse("192.168.0.1"))`
- 加工结果（私网地址结果为空，使用公网地址可解析出内容）
ip: 192.168.0.1
geo_parse: {"city": "", "province": "", "country": "", "isp": "", "lat": "", "lon": ""}

b. 多个IP查询。

- 测试数据

```
{
  "ip": "192.168.0.1"
}
```
- 加工规则
`e_set("geo_parse", geo_parse("192.168.0.1,192.168.0.2", ip_sep=","))`
- 加工结果（私网地址结果为空，使用公网地址可解析出内容）
geo_parse: {"192.168.0.1": {"province": "", "country": "", "lat": "", "lon": ""}, "192.168.0.2": {"city": "", "province": "", "country": "", "isp": "", "lat": "", "lon": ""}}
ip: 192.168.0.1

ip_cidrmatch

IP地址是否属于CIDR地址块。

- 函数格式
`ip_cidrmatch(cidr_subnet, ip, default="")`
- 参数说明

参数名称	参数类型	是否必填	说明
cidr_subnet	String	是	CIDR地址块
ip	String	是	IP地址。
default	String	否	如果IP地址与CIDR地址块无法匹配时，返回该值。

- 返回结果
当IP地址属于CIDR地址块时，返回true，否则返回false。
- 函数示例

a. ipv4地址与CIDR地址块匹配。

- 测试数据

```
{
  "subnet": "192.168.1.0/24"
}
```
- 加工规则
`e_set("result", ip_cidrmatch(v("subnet"), "192.168.1.11"))`
- 加工结果
subnet: 192.168.1.0/24
result: true

b. ip地址与CIDR地址块无法匹配。

■ 测试数据

```
{
  "subnet": "192.168.1.0/24"
}
```

■ 加工规则

```
e_set("result",ip_cidrmatch(v("subnet"), "192.168.100.10", default="error"))
```

■ 加工结果

```
subnet: 192.168.1.0/24
result: error
```

ip_version

判断IP地址为IPv4还是IPv6。

- 函数格式

```
ip_version(ip, default="")
```

- 参数说明

参数名称	参数类型	是否必填	说明
ip	String	是	输入IP地址。
default	String	否	无法判断IP地址版本时，返回该值。

- 返回结果

IPv4或IPv6。

- 函数示例

a. Ipv4地址。

■ 测试数据

```
{
  "ip": "10.21.115.10"
}
```

■ 加工规则

```
e_set("version",ip_version(v("ip")))
```

■ 加工结果

```
ip: 10.21.115.10
version: IPv4
```

b. Ipv6地址。

■ 测试数据

```
{
  "ip": "2001:0db8:85a3:0000:0000:8a2e:0370:7334"
}
```

■ 加工规则

```
e_set("version",ip_version(v("ip")))
```

■ 加工结果

```
ip: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
version: IPv6
```

ip_type

判断IP地址是私有地址还是公有地址。

- 函数格式

```
ip_type(ip, default="")
```

- 参数说明

参数名称	参数类型	是否必填	说明
ip	String	是	IP地址。
default	String	否	无法判断IP地址类型时，返回该值。

- 返回结果

IP类型private、reserved、loopback、public和allocated ripe ncc。

- 函数示例

a. 私有地址。

- 测试数据

```
{
  "ip": "10.1.2.3"
}
```

- 加工规则

```
e_set("result", ip_type(v("ip")))
```

- 加工结果

```
ip: 10.1.2.3
result: private
```

b. Ipv6地址。

- 测试数据

```
{
  "ip": "127.0.0.1"
}
```

- 加工规则

```
e_set("result", ip_type(v("ip")))
```

- 加工结果

```
ip: 127.0.0.1
result: loopback
```

ip_makenet

将单个IP地址转换为CIDR地址块。

- 函数格式

```
ip_makenet(ip, subnet_mask=None, default="")
```

- 参数说明

参数名称	参数类型	是否必填	说明
ip	String	是	IP地址

subnet_mask	String	是	子网掩码， 如果ip中输入的是IP网段， 则子网掩码可以为空。
default	String	否	无法将IP地址转成CIDR地址块时， 返回该值。

- **返回结果**
CIDR地址块。
- **函数示例**
 - a. ip地址范围转为CIDR地址块。

▪ 测试数据

```
{
  "ip": "192.168.10.0-192.168.10.255"
}
```

▪ 加工规则

```
e_set("result",ip_makenet(v("ip")))
```

▪ 加工结果

```
ip: 192.168.10.0-192.168.10.255
result: 192.168.10.0/24
```

- b. ip地址转为CIDR地址块。

▪ 测试数据

```
{
  "ip": "192.168.10.0"
}
```

▪ 加工规则

```
e_set("result",ip_makenet(v("ip"), "255.255.255.0" ))
```

▪ 加工结果

```
ip: 192.168.10.0
result: 192.168.10.0/24
```

ip_to_format

将输入的CIDR地址块按照Prefixlen或者Netmask格式输出。

- **函数格式**
ip_to_format(cidr_subnet, want_prefix_len=0, default="")
- **参数说明**

参数名称	参数类型	是否必填	说明
cidr_subnet	String	是	输入CIDR地址块， 例如： 192.168.10.0/24
want_prefix_len	Int	否	设置返回格式， 默认为0。 • 0：无格式返回。 • 1： prefix格式返回。 • 2： netmask格式返回。 • 3： IP网段格式返回。

default	String	否	无法将输入的CIDR地址块按照格式输出时，返回该值。

- **返回结果**

特定格式的IP地址。

- **函数示例**

a. 按照IP地址网段格式输出。

- **测试数据**

```
{
  "ip": "192.168.11.0/24"
}
```

- **加工规则**

```
e_set("result",ip_to_format(v("ip"),3))
```

- **加工结果**

```
ip: 192.168.11.0/24
result: 192.168.11.0-192.168.11.255
```

b. 按照netmask格式输出。

- **测试数据**

```
{
  "ip": "192.168.11.0/24"
}
```

- **加工规则**

```
e_set("result",ip_to_format(v("ip"),2))
```

- **加工结果**

```
ip: 192.168.11.0/24
result: 192.168.11.0/255.255.255.0
```

c. 按照prefix格式输出。

- **测试数据**

```
{
  "ip": "192.168.11.0/24"
}
```

- **加工规则**

```
e_set("result",ip_to_format(v("ip"),1))
```

- **加工结果**

```
ip: 192.168.11.0/24
result: 192.168.11.0/24
```

ip_overlaps

两个网段是否存在重叠。

- **函数格式**

```
ip_overlaps(cidr_subnet, cidr_subnet2, default="")
```

- **参数说明**

参数名称	参数类型	是否必填	说明

cidr_subnet	String	是	输入CIDR地址块1。
cidr_subnet2	String	是	输入CIDR地址块2。
default	String	否	无法判断两个CIDR地址块是否重叠时，返回该值。

• **返回结果**

- 0：两个CIDR地址块不重叠
- 1：两个CIDR地址块重叠在结束位置
- 1：两个CIDR地址块重叠在开始位置

• **函数示例**

a. 示例1：两个CIDR地址块不重叠。

■ 测试数据

```
{
  "a": "192.168.0.0/24",
  "b": "192.168.1.0/24"
}
```

■ 加工规则

```
e_set("result",ip_overlaps(v("a"),v("b")))
```

■ 加工结果

```
a: 192.168.0.0/24
b: 192.168.1.0/24
result: 0
```

b. 示例2：两个CIDR地址在开始位置重叠。

■ 测试数据

```
{
  "a": "192.168.1.0/24",
  "b": "192.168.0.0/23"
}
```

■ 加工规则

```
e_set("result",ip_overlaps(v("a"),v("b")))
```

■ 加工结果

```
a: 192.168.1.0/24
b: 192.168.0.0/23
result: 1
```

c. 示例3：两个CIDR地址在结尾位置重叠。

■ 测试数据

```
{
  "a": "192.168.0.0/23",
  "b": "192.168.1.0/24"
}
```

■ 加工规则

```
e_set("result",ip_overlaps(v("a"),v("b")))
```

■ 加工结果

```
a: 192.168.0.0/23
b: 192.168.1.0/24
result: 1
```

ip2long

将字符串格式的IP地址转换成长整型数据。

- 函数格式

```
ip2long(value,default=0)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	String	是	需要被转换的值。
default	String	否	某个不合法的IP地址被转换成的值。

- 返回结果

转换后的长整型数据。

- 函数示例

a. 正常解析。

- 测试数据

```
{
  "ip": "116.209.192.0"
}
```

- 加工规则

```
e_set("result",ip2long(v("ip")))
```

- 加工结果

```
result: 1959903232
ip: 116.209.192.0
```

b. 非法解析。

- 测试数据

```
{
  "ip": "116.209.abc.xxx"
}
```

- 加工规则

```
e_set("result",ip2long(v("ip"), "error"))
```

- 加工结果

```
result: error
ip: 116.209.abc.xxx
```

long2ip

将长整型数据转换成字符串格式的IP地址。

- 函数格式

```
long2ip(value,default="")
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	String	是	需要被转换的值。

default	String	否	表示将不合法的长整型数据转成空字符串。

- **返回结果**
长整型转换成功后的IP地址。
- **函数示例**
 - a. 正确的转换。
 - 测试数据

```
{
  "data": "1959903232"
}
```
 - 加工规则

```
e_set("ip",long2ip(v("data")))
```
 - 加工结果

```
data: 1959903232
ip: 116.209.192.0
```
 - b. 转换失败后自定义的错误处理。
 - 测试数据

```
{
  "data": "4294967296"
}
```
 - 加工规则

```
e_set("ip",long2ip(v("data")), default="error")
```
 - 加工结果

```
data: 4294967296
ip: error
```

10.2.1.7 特定结构化数据函数

本文介绍特定结构化数据函数的语法规则，包括参数解释、函数示例等。

类型	函数	说明
JSON	json_select	根据JMES语法提取或计算JSON表达式中特定的值。
	json_parse	将值解析为JSON对象。
XML	xml_to_json	将xml数据转成JSON数据。

json_select

根据JMES语法提取或计算JSON表达式中特定的值。

- **函数格式**
`json_select(value, jmes, default=None, restrict=false)`
- **参数说明**

参数	类型	是否必填	说明
value	String、JSON	是	传入待提取字段的JSON表达式或字段。
jmes	String	是	JMES表达式，表示提取的字段。
default	String	否	如果提取字段不存在，则返回此处设置的值。默认为None，表示不返回字段。
restrict	Bool	否	提取字段的值不是合法的JSON格式时，是否严格限制加工。默认值false。 - false：忽略报错，数据加工继续处理，返回default定义的值。 - true：直接报错，数据加工不再继续处理，直接丢弃该条日志。

- **返回结果**

返回提取到的值。

- **函数示例**

a. 示例1：从content字段提取元素name的值。

- **测试数据**

```
{
  "content": {"name": "xiaoming", "age": 10}
}
```

- **加工规则**

```
e_set("json_filter", json_select(v("content"), "name"))
```

- **加工结果**

```
content: {"name": "xiaoming", "age": 10}
json_filter: xiaoming
```

b. 示例2：从content字段提取元素name包含的所有值。

- **测试数据**

```
{
  "content": {"name": ["xiaoming", "xiaowang", "xiaoli"], "age": 10}
}
```

- **加工规则**

```
e_set("json_filter", json_select(v("content"), "name[*]"))
```

- **加工结果**

```
content: {"name": ["xiaoming", "xiaowang", "xiaoli"], "age": 10}
json_filter: ["xiaoming", "xiaowang", "xiaoli"]
```

c. 示例3：从content字段提取元素name3的值，若字段不存在，则返回default的值。

- **测试数据**

```
{
  "content": {"name": "xiaoming", "age": 10}
}
```

- **加工规则**

```
e_set("json_filter", json_select(v("content"), "name3", default="None"))
```


- 加工结果

content: {"name": "xiaoming", "age": 10}
json_filter: None
- d. 示例4: 从content字段提取携带短划线 (-) 的元素name-test的值。

▪ 测试数据

{
 "content": {"name": {"name-test": "xiaoming"}, "age": 10}
}

▪ 加工规则

e_set("json_filter", json_select(v("content"), 'name."name-test"', default=None))

▪ 加工结果

content: {"name": {"name-test": "xiaoming"}, "age": 10}
json_filter: xiaoming
- e. 示例5: 从content字段提取携带短划线 (-) 的元素name-test的值, 若元素不存在, 则不返回字段。

▪ 测试数据

{
 "content": {"name": {"name.test": "xiaoming"}, "age": 10}
}

▪ 加工规则

e_set("json_filter", json_select(v("content"), 'name."name-test"', default=None))

▪ 加工结果

content: {"name": {"name-test": "xiaoming"}, "age": 10}

json_parse

将值解析为JSON对象。

- 函数格式

json_parse(value, default=None, restrict=false)
- 参数说明

参数	类型	是否必填	说明
value	String	是	传入需要被解析的字段。
default	String	否	如果解析字段不存在, 则返回此处设置的值。默认为None, 表示不返回字段。
restrict	Bool	否	解析字段的值不是合法的JSON格式时, 是否严格限制加工。默认值false。 • false: 忽略报错, 数据加工继续处理, 返回default定义的值。 • true: 直接报错, 数据加工不再继续处理, 直接丢弃该条日志。
- 返回结果

返回转换后的JSON对象。

• 函数示例

- a. 示例1：提取content字段的JSON值。
- 测试数据

```
{
  "content": {"abc": 123, "xyz": "test" }
}
```
 - 加工规则

```
e_set("json", json_parse(v("content")))
```
 - 加工结果

```
content: {"abc": 123, "xyz": "test" }
json: {"abc": 123, "xyz": "test"}
```
- b. 示例2：提取content字段的值，如果不是JSON格式，则返回default的值。
- 测试数据

```
{
  "content": "this is not json"
}
```
 - 加工规则

```
e_set("json", json_parse(v("content"), default="FFF", restrict=false))
```
 - 加工结果

```
content: this is not json
json: FFF
```

xml_to_json

将xml数据转成JSON数据。

- 函数格式

```
xml_to_json(source)
```

- 参数说明

参数	类型	是否必填	说明
source	String	是	传入需要被转换的字段。

- 返回结果

返回转换后的JSON数据。

- 函数示例

- 测试数据

```
{
  "str": "<data><country name='Liechtenstein'><rank>1</rank><year>2008</year><gdppc>141100</gdppc><neighbor name='Austria' direction='E'><neighbor name='Switzerland' direction='W'></country><country name='Singapore'><rank>4</rank><year>2011</year><gdppc>59900</gdppc><neighbor name='Malaysia' direction='N'></country><country name='Panama'><rank>68</rank><year>2011</year><gdppc>13600</gdppc><neighbor name='Costa Rica' direction='W'><neighbor name='Colombia' direction='E'></country></data>"
}
```

- 加工规则

```
e_set("str_json", xml_to_json(v("str")))
```

- 加工结果

```
str: <data><country name="Liechtenstein"><rank>1</rank><year>2008</year><gdppc>141100</gdppc><neighbor name="Austria" direction="E"><neighbor name="Switzerland" direction="W"></country><country name="Singapore"><rank>4</rank><year>2011</year><gdppc>59900</gdppc><neighbor name="Malaysia" direction="N"></country><country name="Panama"><rank>68</rank><year>2011</year><gdppc>13600</gdppc><neighbor name="Costa Rica" direction="W"><neighbor name="Colombia" direction="E"></country></data>
```

```
year<gdppc>59900</gdppc><neighbor name="Malaysia" direction="N"/></country><country
name="Panama"><rank>68</rank><year>2011</year><gdppc>13600</gdppc><neighbor
name="Costa Rica" direction="W"/><neighbor name="Colombia" direction="E"/></country></
data> str_json:{"data": {"country": [{"@name": "Liechtenstein", "rank": "1", "year": "2008",
"gdppc": "141100", "neighbor": [{"@name": "Austria", "@direction": "E"}, {"@name":
"Switzerland", "@direction": "W"}]}, {"@name": "Singapore", "rank": "4", "year": "2011",
"gdppc": "59900", "neighbor": {"@name": "Malaysia", "@direction": "N"}}, {"@name":
"Panama", "rank": "68", "year": "2011", "gdppc": "13600", "neighbor": [{"@name": "Costa Rica",
"@direction": "W"}, {"@name": "Colombia", "@direction": "E"}]}}]}
```

10.2.1.8 正则表达式函数

本文介绍正则表达式函数的语法规则，包括参数解释、函数示例等。

类型	函数	说明
值提取函数	regex_select	根据正则表达式提取符合条件的值。
	regex_findall	根据正则表达式获得符合条件的所有值列表。
匹配判断	regex_match	判断是否匹配正则表达式。
替换	regex_replace	根据正则表达式替换字符串中的指定字符。
切分	regex_split	将一个字符串分割成字符串数组。

regex_select

- 函数格式
regex_select(value, r"regular expression", mi=None, gi=None)
- 参数说明

参数名称	参数类型	是否必填	说明
value	任意	是	填入要匹配的值。
regular expression	String	是	匹配的正则表达式。
mi	int	否	表示匹配到的第几个表达式，默认None，表示第一个。
gi	int	否	表示匹配到的第几个分组，默认None，表示第一个。

- 返回结果
返回提取的值。
- 函数示例
 - 示例1：提取字段str中符合正则表达式的第一个值。
 - 测试数据

```
{
  "str": "iZbp1a65x3r1vhpe94fi2qZ"
}
```

■ 加工规则

```
e_set("regex", regex_select(v("str"), r"\d+"))
e_set("regex2", regex_select(v("str"), r"\d+", mi=None))
e_set("regex3", regex_select(v("str"), r"\d+", mi=0))
```

■ 加工结果

```
regex:1
regex2:1
regex3:1
str:iZbp1a65x3r1vhpe94fi2qZ
```

b. 示例2：提取字段str中符合正则表达式的第一个和第二个的值。

■ 测试数据

```
{
  "str": "abc123 xyz456"
}
```

■ 加工规则

```
# 提取字段str中符合正则表达式的第一个值。
e_set("regex", regex_select(v("str"), r"\d+"))
# 提取字段str中符合正则表达式的第二个值。
e_set("regex2", regex_select(v("str"), r"\d+", mi=1))
```

■ 加工结果

```
regex: 123
regex2: 456
str: abc123 xyz456
```

c. 示例3：提取字段str中不同表达式的不同分组的值。

■ 测试数据

```
{
  "str": "abc123 xyz456"
}
```

■ 加工规则

```
# 提取字段str中符合正则表达式的第一个表达式的第一个分组的值。
e_set("regex", regex_select(v("str"), r"[a-z]+(\d+)", gi=0))
# 提取字段str中符合正则表达式的第二个表达式的第一个分组的值。
e_set("regex2", regex_select(v("str"), r"[a-z]+(\d+)", mi=1, gi=0))
# 提取字段str中符合正则表达式的第一个表达式的第一个分组的值。
e_set("regex3", regex_select(v("str"), r"([a-z]+)(\d+)", gi=0))
# 提取字段str中符合正则表达式的第一个表达式的第二个分组的值。
e_set("regex4", regex_select(v("str"), r"([a-z]+)(\d+)", gi=1))
```

■ 加工结果

```
str: abc123 xyz456
regex: 123
regex2: 456
regex3: abc
regex4: 123
```

regex_findall

根据正则表达式获得符合条件的所有值的一个列表。

● 函数格式

```
regex_findall(value, r"regular expression")
```

● 参数说明

参数名称	参数类型	是否必填	说明

value	任意	是	要匹配的值。
regular expression	String	是	正则表达式。

- **返回结果**

返回获得符合条件的列表。

- **函数示例**

获取字段str所有的数字。

- 测试数据

```
{
  "str": "iZbp1a65x3r1vhpe94fi2qZ"
}
```

- 加工规则

```
e_set("regex_findall", regex_findall(v("str"),r"\d+"))
```

- 加工结果

```
str: iZbp1a65x3r1vhpe94fi2qZ
regex_findall: ["1", "65", "3", "1", "94", "2"]
```

regex_match

判断是否匹配正则表达式。

- **函数格式**

```
regex_match(value, r"regular expression", full=false)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意	是	要匹配的值。
regular expression	String	是	正则表达式。
full	Bool	否	是否完全匹配，默认为false。

- **返回结果**

返回true或者false。

- **函数示例**

判断字段str是否包含数字。

- 测试数据

```
{
  "str": "iZbp1a65x3r1vhpe94fi2qZ"
}
```

- 加工规则

```
# 判断字段str是否包含数字。
e_set("regex_match", regex_match(v("str"),r"\d+"))
# 判断字段str是否全部为数字。
e_set("regex_match2", regex_match(v("str"),r"\d+",full=true))
```

- 加工结果

```
str: iZbp1a65x3r1vhpe94fi2qZ
regex_match: true
regex_match2: false
```

regex_replace

根据正则表达式替换字符串中的指定字符。

- 函数格式

```
regex_replace(value, r"regular expression", replace="", count=0)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意	是	要被替换的值。
regular expression	String	是	正则表达式。
replace	String	否	替换后的新字符。默认为空，表示删除字符。 支持正则表达式，例如r"\1****\2"，表示替换后的字符串要满足该正则表达式。 \1代表第一个分组。 \2代表第二个分组。
count	Number	否	替换次数。默认为0，表示替换所有。

- 返回结果

返回替换后的新字符串。

- 函数示例

a. 示例1：把str中的所有数字用13替换。

- 测试数据

```
{
  "str": "iZbp1a65x3r1vhpe94fi2qZ ",
  "replace": "13"
}
```

- 加工规则

```
e_set("regex_replace", regex_replace(v("str"),r"\d+",v("replace")))
```

- 加工结果

```
str: iZbp1a65x3r1vhpe94fi2qZ
replace: 13
regex_replace: iZbp13a13x13r13vhpe13fi13qZ
```

b. 示例2：对手机号中间4位数字进行脱敏处理。

- 测试数据

```
{
  "iphone": "13900001234"
}
```

- 加工规则

- replace=r"\1****\2"表示替换后的字符串要满足正则表达式r"\1****\2"。

- \1代表第一个分组，即(\d{0,3})。
- \2代表第二个分组，即(\d{4})。

```
e_set(
  "sec_iphone",
  regex_replace(v("iphone"), r"(\d{0,3})\d{4}(\d{4})", replace=r"\1****\2"),
)
```

- 加工结果

```
iphone: 13900001234
sec_iphone: 139****1234
```

regex_split

将一个字符串分割成字符串数组。

- 函数格式

```
regex_split(value, r"regular expression", maxsplit=0)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意	是	要分裂的值。
regular expression	String	是	正则表达式。
maxsplit	int	否	最大分裂匹配次数。默认为0表示全部匹配分裂，如果为1，表示匹配中一个就分裂，剩余不再进行匹配。

- 返回结果

返回分割后的数组列表。

- 函数示例

将字段str按照数字进行分裂。

- 测试数据

```
{
  "str": "iZbp1a65x3r1vhpe94fi2qZ"
}
```

- 加工规则

```
e_set("regex_split", regex_split(v("str"), r"\d+"))
```

- 加工结果

```
str: iZbp1a65x3r1vhpe94fi2qZ
regex_split: ["iZbp", "a", "x", "r", "vhpe", "fi", "qZ"]
```

10.2.1.9 日期时间函数

本文介绍日期时间函数的语法规则，包括参数解释、函数示例等。

在DSL加工逻辑中，日志中的所有值都以字符串的形式存储，需要根据场景对数据类型进行转换。

日志中时间主要有以下三种数据类型，您可以根据本文提供的日期时间函数进行日期时间格式转换。

- 字符串，例如2022/07/03 02-41-26。
 - Unix时间戳，例如1559500886。
 - 日期时间对象，例如2022-07-01 10:10:10+08:00或者2022-07-01 10:10:10。
- Unix时间戳本质上也是字符串。本文的日期时间函数中，只有dt_parse、dt_str和dt_parsetimestamp函数支持以上三种数据类型作为参数。

函数列表

类型	函数	说明
通用日期时间转换	dt_parse	将值或时间表达式的值转换为日期时间对象。
	dt_str	将值或时间表达式的值转换为字符串。
	dt_parsetimestamp	将值或时间表达式的值转换为Unix时间戳。
	dt_prop	获取值或时间表达式值的特定属性，包括所属day、year等。
获取日期时间	dt_now	获取当前日期时间对象。
	dt_today	获取当前日期，不含时间。
	dt_utcnow	获取当前时区的当前日期时间对象。
	dt_fromtimestamp	将Unix时间戳转换为日期时间对象。
	dt_utcfromtimestamp	将Unix时间戳转换为当前时区的日期时间对象。
	dt_strptime	将时间字符串解析为日期时间对象。
获取Unix时间戳	dt_currentstamp	获取当前Unix时间戳。
	dt_totimestamp	将日期时间对象转换为Unix时间戳。
获取日期时间字符串	dt_strftime	将日期时间对象按照指定格式转换为字符串。
	dt_strftimestamp	将Unix时间戳按照指定格式转换为字符串。
修改日期时间	dt_truncate	从值或时间表达式中截取指定的时间粒度。
	dt_add	根据指定的时间粒度修改值或时间表达式的值。
	dt_MO	dt_add函数中传递给weekday参数的值，用于表示特定星期一的偏移量。
	dt_TU	dt_add函数中传递给weekday参数的值，用于表示特定星期二的偏移量。

	dt_WE	dt_add函数中传递给weekday参数的值，用于表示特定星期三的偏移量。
	dt_TH	dt_add函数中传递给weekday参数的值，用于表示特定星期四的偏移量。
	dt_FR	dt_add函数中传递给weekday参数的值，用于表示特定星期五的偏移量。
	dt_SA	dt_add函数中传递给weekday参数的值，用于表示特定星期六的偏移量。
	dt_SU	dt_add函数中传递给weekday参数的值，用于表示特定星期日的偏移量。
修改日期时区	dt_astimezone	将值或时间表达式的值转换为特定时区的日期时间对象。
获取差异	dt_diff	按照特定粒度获取两个值或时间表达式值的差异值。

dt_parse

将值或时间表达式的值转换为日期时间对象。

- 函数格式

dt_parse(value, tz=None)

- 参数说明

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
tz	String	否	表示时区，默认为None。

- 返回结果

返回转换后的日期时间对象。

- 函数示例

a. 示例1：将time字段的值转化成日期时间。

- 测试数据

```
{
  "time": "1559500886"
}
```

- 加工规则

```
e_set("test_time", dt_parse(v("time")))
```

- 加工结果

```
time: 1559500886
test_time: 2019-06-02 18:41:26
```

b. 示例2：将time字段的值转化成日期时间，时区是上海。

■ 测试数据

```
{
  "time": "2019-06-01 10:10:10"
  "tz": "Asia/Shanghai"
}
```

■ 加工规则

```
e_set("test_time", dt_parse(v("time"),tz=v("tz")))
```

■ 加工结果

```
time: 2019-06-01 10:10:10
tz: Asia/Shanghai
test_time: 2019-06-01 10:10:10+08:00
```

dt_str

将值或时间表达式的值转换为字符串。

- 函数格式

```
dt_str(value, fmt="format_string", tz=None)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
fmt	String	否	格式化字符串。
tz	String	否	表示时区。

- 返回结果

返回转换后的时间字符串。

- 函数示例

a. 示例1：把time字段的值转换成fmt形式，时区东京。

■ 测试数据

```
{
  "time": "2019-06-03 02:41:26",
  "fmt": "%Y/%m/%d %H-%M-%S",
  "tz": "Asia/Tokyo"
}
```

■ 加工规则

```
e_set("dt_str", dt_str(v("time"),fmt=v("fmt"),tz=v("tz")))
```

■ 加工结果

```
{
  "time": "2019-06-03 02:41:26"
  "fmt": "%Y/%m/%d %H-%M-%S"
  "tz": "Asia/Tokyo"
  "dt_str": "2019/06/03 02-41-26"
}
```

b. 示例2：把time字段的值（Unix时间戳）转换成fmt形式。

■ 测试数据

```
{
  "time": "1559500886",
  "fmt": "%Y/%m/%d %H-%M-%S"
}
```

- 加工规则
e_set("dt_str", dt_str(v("time"),fmt=v("fmt")))
- 加工结果
time: 1559500886
fmt: %Y/%m/%d %H-%M-%S
dt_str: 2019/06/02 18-41-26

c. 示例3：把time字段的值转换成默认形式。

- 测试数据
{
 "time": "2019-06-03 02:41:26"
}
- 加工规则
e_set("dt_str", dt_str(v("time")))
- 加工结果
time: 2019-06-03 02:41:26
dt_str: 2019-06-03 02:41:26

dt_parsetimestamp

将值或时间表达式的值转换为Unix时间戳。

- 函数格式
dt_parsetimestamp(value, tz=None)
- 参数说明

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
tz	String	否	表示时区，默认为None。

- 返回结果
返回转换后的Unix时间戳。
 - 函数示例
- a. 示例1：把time字段的值转换成时间戳，时区是北京。

- 测试数据
{
 "time": "2019-06-03 2:41:26",
 "tz": "Asia/Tokyo"
}
- 加工规则
e_set("dt_parsetimestamp", dt_parsetimestamp(v("time"),v("tz")))

- 加工结果

time: 2019-06-03 2:41:26
tz: Asia/Tokyo
dt_parsetimestamp: 1559497286
- b. 示例2：把time字段的值转换成Unix时间戳。

▪ 测试数据

{
 "time": "2019-06-03 2:41:26"
}

▪ 加工规则

e_set("dt_parsetimestamp",dt_parsetimestamp(v("time")))

▪ 加工结果

time: 2019-06-03 2:41:26
dt_parsetimestamp: 1559529686
- c. 示例3：把time字段的值转换成Unix时间戳。

▪ 测试数据

{
 "time": "2019-06-03 2:41:26"
}

▪ 加工规则

e_set("dt_parsetimestamp",dt_parsetimestamp(v("time")))

▪ 加工结果

time: 2019-06-03 02:41:26+8:00
dt_parsetimestamp: 1559500886

dt_prop

获取值或时间表达式值的特定属性，包括所属day、year等。

- 函数格式

dt_prop(value, props)
- 参数说明

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
props	String	是	需要获取的时间属性。例如属性名为year，则只会输出年份。参数值可以为day、year、month、hour、second、minute、microsecond、weekday、weekdayname、weekdayshortname、monthname、monthshortname、dayofyear、dayofweek、weekofyear、weekofyear_m、tzname、weekofmonth。

- **返回结果**

返回提取的属性。

- **函数示例**

a. 示例1：提取time字段的值的day属性值。

- **测试数据**

```
{  
  "time": "2018-10-2 09:11:40"  
}
```

- **加工规则**

```
e_set("dt_parsetimestamp",dt_prop(dt_parse(v("time")), "day"))
```

- **加工结果**

```
time: 2018-10-2 09:11:40  
dt_parsetimestamp: 2
```

b. 示例2：提取time字段的值的year属性值。

- **测试数据**

```
{  
  "time": "2018-10-2 09:11:40"  
}
```

- **加工规则**

```
e_set("dt_parsetimestamp",dt_prop(dt_parse(v("time")), "year"))
```

- **加工结果**

```
time: 2018-10-2 09:11:40  
dt_parsetimestamp: 2018
```

c. 示例3：提取time字段的值的weekdayname属性值。

- **测试数据**

```
{  
  "time": "2018-10-2 09:11:40"  
}
```

- **加工规则**

```
e_set("dt_prop",dt_prop(dt_parse(v("time")), "weekdayname"))
```

- **加工结果**

```
time: 2018-10-2 09:11:40  
dt_prop: Tuesday
```

d. 示例4：提取time字段的值的weekofyear属性值。

- **测试数据**

```
{  
  "time": "2018-10-2 09:11:40"  
}
```

- **加工规则**

```
e_set("dt_prop",dt_prop(dt_parse(v("time")), "weekofyear"))
```

- **加工结果**

```
time: 2018-10-2 09:11:40  
dt_prop: 39
```

dt_now

获取当前日期时间对象。

- **函数格式**

`dt_now(tz=None)`

- **参数说明**

参数名称	参数类型	是否必填	说明
tz	String	否	表示时区，默认为None。

- **返回结果**

返回指定时区的时间对象。

- **函数示例**

获取当前时间，时区是上海。

- 测试数据

```
{
  "tz": "Asia/Shanghai"
}
```

- 加工规则

```
e_set("dt_now", dt_now(tz=v("tz")))
```

- 加工结果

```
tz: Asia/Shanghai
dt_now: 2022-06-30 11:21:25.111836+08:00
```

dt_today

获取当前日期，不含时间。

- **函数格式**

`dt_today(tz=None)`

- **参数说明**

参数名称	参数类型	是否必填	说明
tz	String	否	表示时区，默认为None。

- **返回结果**

返回指定时区的日期对象。

- **函数示例**

获取当前日期，不含时间。

- 测试数据

无

- 加工规则

```
e_set("dt_today", dt_today())
```

- 加工结果

```
dt_today: 2022-06-30 00:00:00
```

dt_utcnow

获取当前UTC时间。

- **函数格式**

```
dt_UTCnow()
```

- **参数说明**

无。

- **返回结果**

返回当前UTC时间。

- **函数示例**

获取当前UTC时间。

- 测试数据

无

- 加工规则

```
e_set("dt_UTCnow",dt_UTCnow())
```

- 加工结果

```
dt_UTCnow:2022-06-30 03:33:56.614005
```

dt_fromtimestamp

将Unix时间戳转换为日期时间对象。

- **函数格式**

```
dt_fromtimestamp(value, tz=None)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	值或时间表达式。
tz	String	否	表示时区，默认为None。

- **返回结果**

返回转换后的日期时间。

- **函数示例**

a. 示例1：把time字段的值转化成日期时间对象。

- 测试数据

```
{  
  "time": "1559500886"  
}
```

- 加工规则

```
e_set("dt_fromtimestamp",dt_fromtimestamp(v("time")))
```

- 加工结果

```
time: 1559500886  
dt_fromtimestamp: 2019-06-02 18:41:26
```

b. 示例2：把time字段的值转化成日期时间对象，时区上海。

- 测试数据

```
{  
  "time": "1559500886"  
  "tz": "Asia/Shanghai"  
}
```

tz: Asia/Shanghai

- 加工规则
`e_set("dt_fromtimestamp",dt_fromtimestamp(v("time"),tz=v("tz")))`
- 加工结果
time: 1559500886
tz: Asia/Shanghai
dt_fromtimestamp: 2019-06-03 02:41:26

dt_utcfromtimestamp

将Unix时间戳转换为当前时区的日期时间对象。

- 函数格式
`dt_utcfromtimestamp(value)`

- 参数说明

参数名称	参数类型	是否必填	说明
value	String	是	值或时间表达式。

- 返回结果
返回转换后的日期时间对象。

- 函数示例

- 测试数据

```
{
  "time": "1559500886"
}
```
- 加工规则
`e_set("dt_utcfromtimestamp",dt_utcfromtimestamp(v("time")))`
- 加工结果
time: 1559500886
dt_utcfromtimestamp: 2019-06-02 18:41:26

dt_strptime

将时间字符串解析为日期时间对象。

- 函数格式
`dt_strptime(value, "format_string")`

- 参数说明

参数名称	参数类型	是否必填	说明
value	String	是	值或时间表达式。
fmt	String	否	格式化字符串。

- 返回结果
返回解析后的日期时间对象。

- 函数示例

- 测试数据


```
{
  "time": "2019/06/03 02-41-26",
  "fmt": "%Y/%m/%d %H-%M-%S"
}
```

- 加工规则
e_set("dt_strptime",dt_strptime(v("time"),v("fmt")))
- 加工结果
time: 2019/06/03 02-41-26
fmt: %Y/%m/%d %H-%M-%S
dt_strptime: 2019-06-03 02:41:26

dt_currentstamp

获取当前Unix时间戳。

- **函数格式**
dt_currentstamp(value, normalize='floor')
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	值或时间表达式。
normalize	String	否	计算结果数字格式。参数取值为： • floor（默认值）：向下取整。 • int：取整。 • round：四舍五入。 • ceil：向上取整。

- **返回结果**
返回当前Unix时间戳。
- **函数示例**
 - 测试数据
无
 - 加工规则
e_set("dt_currentstamp",dt_currentstamp())
 - 加工结果
dt_currentstamp: 1656560437

dt_totimestamp

将日期时间对象转换为Unix时间戳。

- **函数格式**
dt_totimestamp(timeexpression)
- **参数说明**

参数名称	参数类型	是否必填	说明
timeexpression	日期时间对象	是	需要被转换的日期时间对象。

- **返回结果**

返回转换后的Unix时间戳。

- **函数示例**

- 测试数据

```
{
  "time": "2019-06-03 2:41:26"
}
```

- 加工规则

```
e_set("dt_totimestamp",dt_totimestamp(dt_parse(v("time"))))
```

- 加工结果

```
time: 2019-06-03 2:41:26
dt_totimestamp: 1559529686
```

dt_strftime

将日期时间对象按照指定格式转换为字符串。

- **函数格式**

```
dt_strftime(timeexpression, "format_string")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
timeexpression	日期时间对象	是	需要被转换的日期时间对象。
format_string	String	是	格式化字符串。

- **返回结果**

返回格式化后的字符串。

- **函数示例**

将time字段的值按照fmt格式转换。

- 测试数据

```
{
  "time": "2019-06-03 2:41:26",
  "fmt": "%Y/%m/%d %H-%M-%S"
}
```

- 加工规则

```
e_set("dt_strftime",dt_strftime(dt_parse(v("time")),v("fmt")))
```

- 加工结果

```
time: 2019-06-03 2:41:26
fmt: %Y/%m/%d %H-%M-%S
dt_strftime: 2019/06/03 02-41-26
```

dt_strtimestamp

将Unix时间戳按照指定格式转换为字符串。

- **函数格式**

```
dt_strtimestamp(value, fmt="format_string", tz=None)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	需要被转换的Unix时间戳。
fmt	String	是	格式化字符串。
tz	String	否	表示时区，默认为None。

- 返回结果

返回格式化后的字符串。

- 函数示例

a. 示例1

- 测试数据

```
{
  "time": "1559500886",
  "fmt": "%Y/%m/%d %H-%M-%S"
}
```

- 加工规则

```
e_set("dt_strftimestamp",dt_strftimestamp(v("time"),v("fmt")))
```

- 加工结果

```
time: 1559500886
fmt: %Y/%m/%d %H-%M-%S
dt_strftimestamp: 2019/06/02 18-41-26
```

b. 示例2

- 测试数据

```
{
  "time": "1559500886",
  "fmt": "%Y/%m/%d %H-%M-%S",
  "tz": "Asia/shanghai"
}
```

- 加工规则

```
e_set("dt_strftimestamp",dt_strftimestamp(v("time"),v("fmt"),v("tz")))
```

- 加工结果

```
dt_strftimestamp:2019/06/03 03-41-26
fmt:%Y/%m/%d %H-%M-%S
time:1559500886
tz:Asia/shanghai
```

dt_truncate

从值或时间表达式中截取指定的时间粒度。

- 函数格式

```
dt_truncate(value, unit='day')
```

- 参数说明

参数名称	参数类型	是否必填	说明

value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
unit	String	是	需要获取的时间属性粒度。默认为day。可选second、minute、<num>_minute（例如：5_minute、19_minute、2_minute等）、hour、day、week、month、quarter、half_year、year。

- **返回结果**

返回提取的特定时间粒度。

- **函数示例**

- a. 示例1

- 测试数据

```
{
  "time": "2019-06-03 2:41:26",
  "unit": "year"
}
```

- 加工规则

```
e_set("dt_truncate",dt_truncate(v("time"),v("unit")))
```

- 加工结果

```
time: 2019-06-03 2:41:26
unit: year
dt_truncate: 2019-01-01 00:00:00
```

- b. 示例2

- 测试数据

```
{
  "time": "2019-06-03 2:41:26",
  "unit": "hour"
}
```

- 加工规则

```
e_set("dt_truncate",dt_truncate(v("time"),v("unit")))
```

- 加工结果

```
time: 2019-06-03 2:41:26
unit: hour
dt_truncate: 2019-06-03 02:00:00
```

dt_add

根据指定的时间粒度修改值或时间表达式的值。

- **函数格式**

```
dt_add(value, dt1=None, dt2=None, year(s)=None, month(s)=None, day(s)=None, hour(s)=None, minute(s)=None, second(s)=None, microsecond(s)=None, week(s)=None, weekday=None)
```

- **参数说明**

参数名称	参数类型	是否必填	说明

value	字符串、Unix时间戳或日期时间对象	是	日期时间表达式。
dt1	字符串、Unix时间戳或日期时间对象	否	日期时间表达式，默认为None。
dt2	字符串、Unix时间戳或日期时间对象	否	日期时间表达式，默认为None。
year/years	Number	否	• year: 表示需要替换的年份，例如year=2020，默认为None。 • years: 表示需要增加年份的数量，如years=1表示在原来year的基础上再增加一年。
day/days	Number	否	• day: 表示需要替换的天，例如day=1，默认为None。 • days: 表示需要增加天的数量，如days=1表示在原来day的基础上加一天。
hour/hours	Number	否	• hour: 表示需要替换的小时，例如hour=1，默认为None。 • hours: 表示需要增加小时的数量，如hours=1表示在原来hour的基础上加一小时。
minute/minutes	Number	否	• minute: 表示需要替换的分钟，例如minute=1，默认为None。 • minutes: 表示需要增加分钟的数量，如minutes=1表示在原来minute的基础上加一分钟。
second/seconds	Number	否	• second: 表示需要替换的秒数，例如second=1，默认为None。 • seconds: 表示需要增加秒的数量，如seconds=1表示在原来second的基础上加一秒钟。

microsecond / microsecond s	Number	否	- microsecond：表示需要替换的毫秒数，例如microsecond=1，默认为None。 - microseconds：表示需要增加毫秒的数量，microseconds=1表示在原来microsecond的基础上加一毫秒。
week/weeks	Number	否	- week：表示需要偏移的周数，例如week=1，默认为None。 - weeks：表示需要增加周的数量，weeks=1表示在原来week的基础上加一周。
weekday	Number	否	表示需要偏移的工作日，例如weekday=dt_MO(1)，默认为None。

- 返回结果

返回修改后的时间表达式。

- 函数示例

a. 示例1

- 测试数据

```
{
  "dt": "2018-10-10 1:2:3",
  "dt1": "2018-11-3 11:12:13",
  "dt2": "2018-10-1 10:10:10"
}
```

- 加工规则

```
e_set("dt_add",dt_add(dt_parse(v("dt")), dt1=dt_parse(v("dt1")), dt2=dt_parse(v("dt2"))))
```

- 加工结果

```
dt:2018-10-10 1:2:3
dt1:2018-11-3 11:12:13
dt2:2018-10-1 10:10:10
dt_add:2018-11-12 02:04:06
```

b. 示例2

- 测试数据

```
{
  "dt": "2018-10-11 02:03:04",
  "year": "2019"
}
```

year: 2019

- 加工规则

```
e_set("dt_add", dt_add(dt_parse(v("dt")), year=ct_int(v("year"))))
```

- 加工结果

```
dt:2018-10-11 02:03:04
dt_add:2019-10-11 02:03:04
year:2019
```

dt_MO

dt_add函数中传递给weekday参数的值，用于表示特定星期一的偏移量。

- **函数格式**

```
dt_MO(Integer_or_negative)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive)。例如-1用op_neg(1)表示。

- **返回结果**

返回偏移后的时间。

- **函数示例**

- 测试数据

```
{
  "time": "2019-08-13 02:03:04"
}
```

- 加工规则

```
e_set("dt_MO",dt_add(v("time"),weekday=dt_MO(1)))
```

- 加工结果

```
time: 2019-08-13 02:03:04
dt_MO: 2019-08-19 02:03:04
```

dt_TU

dt_add函数中传递给weekday参数的值，用于表示特定星期二的偏移量。

- **函数格式**

```
dt_TU(Integer_or_negative)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive)。例如-1用op_neg(1)表示。

- **返回结果**

返回偏移后的时间。

- **函数示例**

- 测试数据

```
{
  "time": "2023-06-17 02:03:04"
}
```

- 加工规则
e_set("dt_TU",dt_add(v("time"),weekday=dt_TU(1)))
- 加工结果
time: 2023-06-17 02:03:04
dt_TU: 2023-06-20 02:03:04

dt_WE

dt_add函数中传递给weekday参数的值，用于表示特定星期三的偏移量。

- 函数格式
dt_WE(Integer_or_negative)
- 参数说明

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。
- 返回结果
返回偏移后的时间。
- 函数示例
 - 测试数据

```
{
  "time": "2023-06-17 02:03:04"
}
```
 - 加工规则
e_set("dt_WE",dt_add(v("time"),weekday=dt_WE(1)))
 - 加工结果
time: 2023-06-17 02:03:04
dt_WE: 2023-06-21 02:03:04

dt_TH

dt_add函数中传递给weekday参数的值，用于表示特定星期四的偏移量。

- 函数格式
dt_TH(Integer_or_negative)
- 参数说明

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。
- 返回结果

返回偏移后的时间。

- 函数示例**
 - 测试数据

```
{
  "time": "2023-06-17 02:03:04"
}
```
 - 加工规则

```
e_set("dt_TH",dt_add(v("time"),weekday=dt_TH(1)))
```
 - 加工结果

```
time: 2023-06-17 02:03:04
dt_TH: 2023-06-22 02:03:04
```

dt_FR

dt_add函数中传递给weekday参数的值，用于表示特定星期五的偏移量。

- 函数格式**

```
dt_FR(Integer_or_negative)
```
- 参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive)。例如-1用op_neg(1)表示。

- 返回结果**

返回偏移后的时间。

- 函数示例**
 - 测试数据

```
{
  "time": "2023-06-17 02:03:04"
}
```
 - 加工规则

```
e_set("dt_FR",dt_add(v("time"),weekday=dt_FR(1)))
```
 - 加工结果

```
time: 2023-06-17 02:03:04
dt_FR: 2023-06-23 02:03:04
```

dt_SA

dt_add函数中传递给weekday参数的值，用于表示特定星期六的偏移量。

- 函数格式**

```
dt_SA(Integer_or_negative)
```
- 参数说明**

参数名称	参数类型	是否必填	说明

Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。

- **返回结果**

返回偏移后的时间。

- **函数示例**

- 测试数据

```
{
  "time": "2023-06-17 02:03:04"
}
```

- 加工规则

```
e_set("dt_SA",dt_add(v("time"),weekday=dt_SA(1)))
```

- 加工结果

```
dt_SA: 2023-06-17 02:03:04
time: 2023-06-17 02:03:04
```

dt_SU

dt_add函数中传递给weekday参数的值，用于表示特定星期日的偏移量。

- **函数格式**

```
dt_SU(Integer_or_negative)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。

- **返回结果**

返回偏移后的时间。

- **函数示例**

- 测试数据

```
{
  "time": "2023-06-17 02:03:04"
}
```

- 加工规则

```
e_set("dt_SU",dt_add(v("time"),weekday=dt_SU(1)))
```

- 加工结果

```
dt_SU: 2023-06-18 02:03:04
time: 2023-06-17 02:03:04
```

dt_astimezone

将值或时间表达式的值转换为特定时区的日期时间对象。

- **函数格式**

```
dt_astimezone(value, tz, reset=false)
```

• 参数说明

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
tz	String	否	表示时区，默认为None。
reset	Bool	否	表示是否对时区重新设置，默认为false表示不设置。如果为true则返回原时间加上设置的时区。

• 返回结果

返回特定时区的日期时间对象。

• 函数示例

a. 示例1

■ 测试数据

```
{
  "time": "2019-06-03 2:41:26",
  "tz": "UTC"
}
```

■ 加工规则

```
e_set("dt_astimezone",dt_astimezone(dt_parse(v("time")), v("tz")))
```

■ 加工结果

```
time: 2019-06-03 2:41:26
tz: UTC
dt_astimezone: 2019-06-03 02:41:26+00:00
```

b. 示例2

■ 测试数据

```
{
  "time": "2019-06-01 10:10:10+10:00",
  "tz": "Asia/shanghai"
}
```

■ 加工规则

```
e_set("dt_astimezone",dt_astimezone(v("time"), v("tz"),reset=true))
```

■ 加工结果

```
time: 2019-06-01 10:10:10+10:00
tz: Asia/shanghai
dt_astimezone: 2019-06-01 10:10:10+08:00
```

c. 示例3

■ 测试数据

```
{
  "time": "2019-06-01 10:10:10+08:00",
  "tz": "Asia/shanghai"
}
```

■ 加工规则

```
e_set("dt_astimezone",dt_astimezone(v("time"), v("tz"),reset=false))
e_set("dt_astimezone_true",dt_astimezone(v("time"), v("tz"),reset=true))
```

■ 加工结果

```
dt_astimezone: 2019-06-01 10:10:10+08:00
dt_astimezone_true: 2019-06-01 10:10:10+08:00
time:2019-06-01 10:10:10+08:00
tz:Asia/shanghai
```

dt_diff

按照特定粒度获取两个值或时间表达式值的差异值。

● 函数格式

```
dt_diff(value1, value2, unit='second', normalize='floor')
```

● 参数说明

参数名称	参数类型	是否必填	说明
value1	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
value2	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
unit	String	否	按照输入时间属性返回，默认second。也可以是microsecond、millisecond、minutes、hours、day等。
normalize	String	否	计算结果数字格式。取值为： • floor（默认值）：向下取整。 • int：取整。 • round：保留N位小数。 • ceil：向上取整。

● 返回结果

返回按照特定粒度获取的两个值的差异值。

● 函数示例

a. 示例1：对time1和time2字段的值，按照秒计算差异值。

■ 测试数据

```
{
  "time1": "2018-10-1 10:10:10",
  "time2": "2018-10-1 10:10:10"
}
```

■ 加工规则

```
e_set("diff",dt_diff(v("time1"), v("time2")))
```

■ 加工结果

```
time1: 2018-10-1 10:10:10
time2: 2018-10-1 10:10:10
diff: 0
```

b. 示例2：对time1和time2字段的值，按照秒计算差异值。

■ 测试数据

```
{
  "time1": "2018-10-1 11:10:10",
  "time2": "2018-10-1 10:10:10"
}
```

■ 加工规则

```
e_set("diff",dt_diff(v("time1"), v("time2")))
```

■ 加工结果

```
time1: 2018-10-1 11:10:10
time2: 2018-10-1 10:10:10
diff: 3600
```

c. 示例3：对time1和time2字段的值，按照微秒计算差异值。

■ 测试数据

```
{
  "time1": "2018-10-1 11:10:11",
  "time2": "2018-10-1 10:10:10",
  "unit": "microsecond"
}
```

■ 加工规则

```
e_set("diff",dt_diff(v("time1"), v("time2"),v("unit")))
```

■ 加工结果

```
diff:3601000000
time1:2018-10-1 11:10:11
time2:2018-10-1 10:10:10
unit:microsecond
```

d. 示例4：对time1和time2字段的值，按照分钟计算差异值，向下取整。

■ 测试数据

```
{
  "time1": "2018-10-1 11:11:59",
  "time2": "2018-10-1 10:10:00",
  "unit": "minute ",
  "normalize": "floor"
}
```

■ 加工规则

```
e_set("diff", dt_diff(v("time1"), v("time2"), v("unit"), v("normalize")))
```

■ 加工结果

```
diff:61
normalize:floor
time1:2018-10-1 11:11:59
time2:2018-10-1 10:10:00
unit:minute
```

e. 示例5：对time1和time2字段的值，按照秒计算差异值，向下取整。

■ 测试数据

```
{
  "time1": "10:00:00",
  "time2": "11:00:00",
  "unit": "second",
  "normalize": "floor"
}
```

- 加工规则
- e_set("diff", dt_diff(v("time1"), v("time2"), v("unit"), v("normalize")))
- 加工结果
- diff:-3600
normalize:floor
time1:10:00:00
time2:11:00:00
unit:second

10.2.1.10 字符串函数

10.2.1.10.1 概述

本文介绍字符串函数的语法规则，包括参数解释、函数示例等。

函数列表

类型	函数	说明
多字符串操作	str_format	按照指定格式对字符串进行格式化。
	str_join	通过连接符将输入的字符串连接生成一个新的字符串。
	str_zip	将两个值或表达式的字符串进行并发分裂然后再合并成一个字符串。
排序、反转、替换	str_sort	字符串排序。
	str_reverse	将一个字符串进行反转。
	str_replace	根据规则将旧字符串替换成新字符串。
	str_translate	将字符串中的指定字符按照对应关系进行替换。
常见操作	str_strip	删除字符串中指定的字符。
	str_lstrip	删除字符串开头的指定字符。
	str_rstrip	删除字符串结尾的指定字符。
	str_lower	将字符串中所有大写字符转换为小写字符。
	str_upper	将字符串中所有小写字符转换为大写字符。
	str_title	将所有单词的第一个字母转化为大写，其余字母均为小写。
	str_capitalize	将字符串的第一个字母转化为大写，其他字母转化为小写。
	str_swapcase	对字符串的大小写字母进行转换。
查找判断	str_count	统计字符串里某个字符出现的次数。

类型	函数	说明
	str_find	判断原字符串中是否包含指定的子字符串。
	str_rfind	查找字符串中指定字符或者字符串最后一次出现的位置。
	str_endswith	判断字符串是否以指定后缀结尾。
	str_startswith	判断字符串是否以指定字符串开头。
切分	str_split	通过指定分隔符对字符串进行分割。
	str_splitlines	通过换行符对字符串进行分割。
	str_partition	根据指定的分隔符将字符串从左往右分割为三部分。
	str_rpartition	根据指定的分隔符将字符串从右往左分割为三部分。
格式化	str_center	用指定字符将字符串填充到指定长度。
	str_ljust	用指定字符将字符串从结尾填充至指定长度。
	str_rjust	用指定字符将原字符串从开头填充至指定长度。
	str_zfill	用字符0从开头将字符串填充至指定长度。
	str_expandtabs	将字符串中的\t转为空格。
字符集判断	str_isalnum	判断字符串是仅由字母和数字组成。
	str_isalpha	判断字符串是否仅由字母组成。
	str_iscii	判断字符串由ASCII组成。
	str_isdecimal	判断字符串是否仅包含十进制字符。
	str_isdigit	判断字符串是否仅由数字组成。
	str_isidentifier	判断字符串是否是有效的Python标识符
	str_islower	判断字符串是否由小写字母组成。
	str_isnumeric	判断字符串是否由数字组成。
	str_isprintable	判断字符串中是否所有字符都是可打印字符。
	str_isspace	判断字符串是否仅由空格字符组成。
	str_istitle	判断字符串中所有单词的拼写首字母是否为大写，且其他字母为小写。
	str_isupper	判断字符串中所有的字母是否都为大写。

类型	函数	说明
	str_uuid	随机生成UUID。

10.2.1.10.2 多字符串和排序、反转、替换

str_format

- **函数格式**
`str_format(format, str1, str2, ...)`
- **参数说明**

参数名称	参数类型	是否必填	说明
format	string	是	转换后的格式
str1	任意	是	待格式化的值1
str2	任意	是	待格式化的值2
- **返回结果**
格式化后的字符串。
- **函数示例**
 - 测试数据：无
 - 加工规则
`e_set("result", str_format("{}={}", "lts", 8))`
 - 加工结果
result: lts=8

str_join

- **函数格式**
`str_join(connector, str1, str2, ...)`
- **参数说明**

参数名称	参数类型	是否必填	说明
connector	任意（自动转为string）	是	连接符，比如#,\$,%等
str1	任意（自动转为string）	是	待连接的值1
str2	任意（自动转为string）	是	待连接的值2
- **返回结果**
连接后的字符串。
- **函数示例**

- 测试数据：无
- 加工规则
e_set("email", str_join("@", "lts", "aa", "com"))
- 加工结果
email: lts@aa.com

str_zip

- **函数格式**

str_zip(value1,value2,combine_sep=None,sep=None,quote=None,lparse=None,rparse=None)

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意（自动转为String）	是	需要被合并的值。
value2	任意（自动转为String）	是	需要被合并的值。
combine_sep	任意（自动转为String）	否	合并时元素之间的合并标识，默认为#
sep	任意（自动转为String）	否	合并后元素之间的分隔符，仅支持单个字符，默认为,
quote	任意（自动转为String）	否	将合并后的元素括起来的字符，当值包含分隔符时需要使用，默认为"
lparse	任意（自动转为String）	否	指定value1中元素之间的分隔符和引用符，默认分隔符为,，默认引用符为"。格式为lparse=(';', '"')。 说明 引用符优先级高于分隔符。
rparse	任意（自动转为String）	否	指定value2中元素之间的分隔符和引用符，默认分隔符为,，默认引用符为"。格式为rparse=(';', '"')。 说明 引用符优先级高于分隔符。

- **返回结果**

合并后的字符串。

- **函数示例**

- 示例1：sep的使用。
 - 测试数据

```
{
  "key1": "value1,value11",
  "key2": "value2,value21"
}
```

- 加工规则

```
e_set("combine", str_zip(v("key1"), v("key2"), sep="|"))
```

- 加工结果

```
key1: value1,value11
key2: value2,value21
combine: value1#value2|value11#value21
```

b. 示例2: quote的使用。

- 测试数据

```
{
  "key1": "\"value1, value2\", value3, \"value4,value5\"",
  "key2": "value11,\"value12,value13\",value14"
}
```

- 加工规则

```
e_set("combine", str_zip(v("key1"), v("key2"), quote='|'))
```

- 加工结果

```
key1: "value1, value2", value3, "value4,value5"
key2: value11,"value12,value13",value14
combine: |value1,value2#value11|,|value3#value12,value13|,|value4,value5#value14|
```

c. 示例3: 不同长度的值。

- 测试数据

```
{
  "key1": "value1,value2",
  "key2": "value11,value12,value13"
}
```

- 加工规则

```
e_set("combine", str_zip(v("key1"), v("key2")))
```

- 加工结果

```
key1: value1,value2
key2: value11,value12,value13
combine: value1#value11,value2#value12
```

d. 示例4: lparse和rparse的使用。

- 测试数据

```
{
  "key1": "|value1, value1|, value2, |value3,value3|",
  "key2": "value11, #value12,value12#, value13"
}
```

- 加工规则

```
e_set("combine", str_zip(v("key1"), v("key2"), lparse="(", "|", rparse="(", "#")))
```

- 加工结果

```
key1: |value1, value1|, value2, |value3,value3|
key2: value11, #value12,value12#, value13
combine: "value1,value1#value11","value2#value12,value12","value3,value3#value13"
```

str_sort

- 函数格式

```
str_sort(value, reverse=false)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被排序的原字符串。
reverse	Boolean	否	默认为false，表示升序排列。

- 返回结果

排序后的字符串。

- 函数示例

- 测试数据

```
{
  "key1": "value"
}
```

- 加工规则

```
e_set("str_sort", str_sort(v("key1")))
```

- 加工结果

```
key1: value
str_sort: aeluv
```

str_reverse

- 函数格式

str_reverse(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被反转的值

- 返回结果

反转后的字符串。

- 函数示例

- 测试数据

```
{
  "data": "switch"
}
```

- 加工规则

```
e_set("reverse_data", str_reverse(v("data")))
```

- 加工结果

```
data: switch
reverse_data: hctiws
```

str_replace

- 函数格式

str_replace(value, old, new, count)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被替换的值。
old	任意（自动转为String）	是	需要被替换的字符串。
new	任意（自动转为String）	是	替换后新的字符串。
count	Number	否	替换次数，可选项。如果不设置count，则表示替换所有。

- **返回结果**

替换后的新字符串。

- **函数示例**

- 测试数据：无

- 加工规则

```
e_set("str_replace", str_replace("this is string example", "is", "was"))
```

- 加工结果

```
str_replace: thwas was string example
```

str_translate

- **函数格式**

```
str_translate(value, replace_string, mapping_string)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被替换的原字符串。
replace_string	任意（自动转为String）	是	需要替换的字符集合。
mapping_string	任意（自动转为String）	是	替换后的字符集合。

- **返回结果**

处理后的字符串。

- **函数示例**

- 测试数据：无

- 加工规则

```
e_set("str_translate", str_translate("lts", "ts", "34"))
```

- 加工结果

```
str_translate: l34
```

10.2.1.10.3 常见操作

str_strip

删除字符串中指定的字符。

- 函数格式

```
str_strip(value, chars)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
chars	任意（自动转为String）	否	字符串开头和结尾需要删除的字符集，默认为\t\r\n。

- 返回结果

修改后的字符串。

- 函数示例

a. 示例1：删除空格。

- 测试数据

```
{
  "source": "   lts"
}
```

- 加工规则

```
e_set("str_strip", str_strip(v("source")))
```

- 加工结果

```
source:   lts
str_strip: lts
```

b. 示例2：删除开头和结尾是#的字符。

- 测试数据

```
{
  "source": "##lts#"
}
```

- 加工规则

```
e_set("str_strip", str_strip(v("source"), "#"))
```

- 加工结果

```
source: ##lts#
str_strip: lts
```

str_lstrip

删除字符串开头指定的字符。

- 函数格式

```
str_lstrip(value, chars)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
chars	任意（自动转为String）	否	字符串开头需要删除的字符集，默认为空格。

- **返回结果**

修改后的字符串。

- **函数示例**

- 测试数据：无
- 加工规则
e_set("str_strip", str_lstrip("***123**", "**"))
- 加工结果
str_strip: 123**

str_rstrip

删除字符串结尾指定的字符。

- **函数格式**

str_rstrip(value, chars)

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
chars	任意（自动转为String）	否	字符串结尾需要删除的字符集，默认为空格。

- **返回结果**

修改后的字符串。

- **函数示例**

- 测试数据：无
- 加工规则
e_set("str_strip", str_rstrip("***123**", "**"))
- 加工结果
str_strip: 123**

str_lower

将字符串中所有大写字符转换为小写字符。

- **函数格式**

str_lower(value)

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

转换后的字符串。

- **函数示例**

- 测试数据

```
{
  "name": "LTs"
}
```

- 加工规则

```
e_set("str_lower", str_lower(v("name")))
```

- 加工结果

```
name: LTs
str_lower: lts
```

str_upper

将字符串中所有小写字符转换为大写字符。

- **函数格式**

```
str_upper(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

转换后的字符串。

- **函数示例**

- 测试数据

```
{
  "name": "LTs"
}
```

- 加工规则

```
e_set("str_upper", str_upper(v("name")))
```

- 加工结果

```
name: LTs
str_upper: LTS
```

str_title

将所有单词的第一个字母转化为大写，其余字母均为小写。

- **函数格式**

```
str_title(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

转换后的字符串。

- **函数示例**

- 测试数据

```
{
  "name": "for example"
}
```

- 加工规则

```
e_set("str_title", str_title(v("name")))
```

- 加工结果

```
name:for example
example str_title: For Exampl
```

str_capitalize

将字符串的第一个字母转化为大写，其他字母转化为小写。

- **函数格式**

```
str_capitalize(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

转换后的字符串。

- **函数示例**

- 测试数据

```
{
  "value": "welcome to xian"
}
```

- 加工规则

```
e_set("str_capitalize", str_capitalize(v("value")))
```

- 加工结果

```
value: welcome to xian
str_capitalize: Welcome to xian
```

str_swapcase

对字符串的大小写字母进行转换。

- **函数格式**

```
str_swapcase(value)
```


- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

转换后的字符串。

- **函数示例**

- 测试数据

```
{
  "name": "this is lts"
}
```

- 加工规则

```
e_set("str_swapcase", str_swapcase(v("name")))
```

- 加工结果

```
name: this is lts
str_swapcase: THIS IS LTS
```

10.2.1.10.4 查找判断、切分和格式化

str_count

统计字符串里某个字符出现的次数。

- **函数格式**

```
str_count(value, sub, start, end)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	被统计的原字符串。
sub	任意（自动转为String）	是	需要统计个数的字符。
start	Number	否	字符串开始搜索的位置。 • 0（默认值）：第一个字符。 • -1：最后一个字符。
end	Number	否	字符串中结束搜索的位置。 • 0：第一个字符。 • -1（默认值）：最后一个字符。

- **返回结果**

返回指定字符出现的次数。

- **函数示例**

- 测试数据

```
{
  "name": "Its is a log service"
}
```

- 加工规则

```
e_set("str_count", str_count(v("name"), "l"))
```

- 加工结果

```
name: Its is a log service
str_count: 2
```

str_find

判断原字符串中是否包含指定的子字符串。

- **函数格式**

```
str_find(value, str, begin, end)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被查找的原字符串。
str	任意（自动转为String）	是	指定查找的子字符串。
begin	Number	否	开始索引的位置。默认为0表示第一个字符，-1表示倒数第一个字符。
end	Number	否	结束索引的位置。默认为字符串的长度。0表示第一个字符，-1表示倒数第一个字符。

- **返回结果**

指定子字符串在原字符串中的位置。如果指定的子字符串在原字符串中出现多次，只返回第一次出现的子字符串的位置。

- **函数示例**

- 测试数据

```
{
  "name": "Its is a log service"
}
```

- 加工规则

```
e_set("str_find", str_find(v("name"), "l"))
```

- 加工结果

```
name: Its is a log service
str_find: 0
```

str_rfind

判断原字符串中是否包含指定的子字符串。

- 函数格式**
str_rfind(value, substr, beg, end)
- 参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	被查找的原字符串。
substr	任意（自动转为String）	是	需要查找的字符。
beg	Number	否	开始查找的位置，默认为0。
end	Number	否	结束查找的位置，默认为字符串的长度。
- 返回结果**
返回字符或字符串最后一次出现的位置。
- 函数示例**
 - 测试数据

```
{
  "name": "Its is a log service"
}
```
 - 加工规则

```
e_set("str_rfind", str_rfind(v("name"), "i"))
```
 - 加工结果

```
name: Its is a log service
str_rfind: 17
```

str_endswith

判断字符串是否以指定后缀结尾。

- 函数格式**
str_endswith(value, suffix, start, end)
- 参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被判断的原字符串。
suffix	任意（自动转为String）	是	后缀结尾规则。该参数可以是一个字符串或者是一个元素。

start	Number	否	字符串检测的起始位置。0表示第一个字符，-1表示倒数第一个字符。
end	Number	否	字符串检测的结束位置。0表示第一个字符，-1表示倒数第一个字符。

- **返回结果**
如果字符串以指定后缀结尾则返回true，否则返回false。
- **函数示例**
 - 测试数据

```
{
  "name": "Its is a log service"
}
```
 - 加工规则

```
e_set("str_endswith",str_endswith(v("name"), "service"))
```
 - 加工结果

```
name: Its is a log service
str_endswith: true
```

str_startswith

判断字符串是否以指定字符串开头。

- **函数格式**
str_startswith(value, prefix, start, end)
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被判断的原字符串。
prefix	任意（自动转为String）	是	前缀规则。该参数可以是一个字符串或者是一个元素。
start	Number	否	字符串检测的起始位置。0表示第一个字符，-1表示倒数第一个字符。
end	Number	否	字符串检测的结束位置。0表示第一个字符，-1表示倒数第一个字符。
- **返回结果**
如果字符串以指定前缀开头则返回true，否则返回false。
- **函数示例**
 - 测试数据

```
{
  "name": "Its is a log service"
}
```

- 加工规则
e_set("str_startswith",str_startswith(v("name"), "lts"))
- 加工结果
name: lts is a log service
str_startswith: true

str_split

通过指定分隔符对字符串进行分割。

- 函数格式
str_split(value, sep=None, maxsplit=-1)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被分割的原字符串。
sep	Number	否	分隔符，None表示空格。
maxsplit	Number	否	最大分裂数。-1表示 unlimited。

- 返回结果
处理后的字符串。
- 函数示例

- 测试数据
{
 "content": "lts,a,log,service"
}
- 加工规则
e_set("str_split", str_split(v("content"), ","))
- 加工结果
content: lts,a,log,service
str_split: ["lts","a","log","service"]

str_splitlines

通过换行符对字符串进行分割。

- 函数格式
str_splitlines(value, keepends)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被分割的原字符串。
keepends	Boolean	否	在输出结果里是否去掉换行符（\r、\r\n、\n）。默认为false，不包含换行符，如果为true，则保留换行符。

- **返回结果**

处理后的列表。

- **函数示例**

- 测试数据

```
{
  "value": "lts\nis\ra\r\n"
}
```

- 加工规则

```
e_set("str_splitlines", str_splitlines(v("value"), false))
```

- 加工结果

```
value: lts\nis\ra\r\n
str_splitlines: ['lts','is','a']
```

str_partition

根据指定的分隔符将字符串从左往右分割为三部分。

- **函数格式**

```
str_partition(value, substr)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被分割的字符串。
substr	任意（自动转为String）	否	指定的分隔符。

- **返回结果**

分隔后的列表。

- **函数示例**

- 测试数据

```
{
  "address": "big.middle.small"
}
```

- 加工规则

```
e_set("str_partition", str_partition(v("address"), "."))
```

- 加工结果

```
address: big.middle.small
str_partition: ["big",".", "middle.small"]
```

str_rpartition

根据指定的分隔符将字符串从右往左分割为三部分。

- **函数格式**

```
str_rpartition(value, substr)
```

- **参数说明**

参数名称	参数类型	是否必填	说明

value	任意（自动转为String）	是	需要被分割的原字符串。
substr	任意（自动转为String）	否	指定的分隔符。

- **返回结果**

分隔后的列表。

- **函数示例**

- 测试数据

```
{
  "address": "big.middle.small"
}
```

- 加工规则

```
e_set("str_partition", str_rpartition(v("address"), "."))
```

- 加工结果

```
address: big.middle.small
str_partition: ["big.middle", ".", "small"]
```

str_center

用指定字符将字符串填充到指定长度。

- **函数格式**

```
str_center(value, width, fillchar)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
width	Number	是	填充后字符串的总长度。
fillchar	任意（自动转为String）	否	填充字符，默认为空格。

- **返回结果**

处理后的字符串。

- **函数示例**

- 测试数据

```
{
  "value": "Its is a log service"
}
```

- 加工规则

```
e_set("str_center", str_center(v("value"), 40, "*"))
```

- 加工结果

```
center: Its is a log service
str_center: *****Its is a log service*****
```

str_ljust

用指定字符将原字符串从结尾填充至指定长度。

- **函数格式**

`str_ljust(value, width, fillchar)`

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
width	Number	是	填充后字符串的总长度。
fillchar	任意（自动转为String）	否	填充字符，默认为空格。

- **返回结果**

处理后的字符串。

- **函数示例**

- 测试数据

```
{
  "value": "Its is a log service"
}
```

- 加工规则

```
e_set("str_ljust", str_ljust(v("value"), 30, "*"))
```

- 加工结果

```
value: Its is a log service
str_ljust: Its is a log service*****
```

str_rjust

用指定字符将原字符串从开头填充至指定长度。

- **函数格式**

`str_rjust(value, width, fillchar)`

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
width	Number	是	填充后字符串的总长度。
fillchar	任意（自动转为String）	否	填充字符，默认为空格。

- **返回结果**

处理后的字符串。

- **函数示例**

- 测试数据

```
{
  "value": "Its is a log service"
}
```


- 加工规则
e_set("str_rjust", str_rjust(v("value"), 30, "**"))
- 加工结果
value: lts is a log service
str_ljust: *****lts is a log service

str_zfill

用字符0从开头将字符串填充至指定长度。

- 函数格式
str_zfill(value, width)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
width	Number	是	填充后字符串的总长度。

- 返回结果
处理后的字符串。

- 函数示例
 - 测试数据
{
 "value": "this is lts"
}
 - 加工规则
e_set("str_zfill", str_zfill(v("value"), 20))
 - 加工结果
value: this is lts
str_zfill: 000000000this is lts

str_expandtabs

将字符串中的\t转为空格。

- 函数格式
str_expandtabs(value, tabsize)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
tabsize	Number	否	指定转换后空格的字符数。

- 返回结果
处理后的字符串。
- 函数示例

a. 示例1：把key中的\t字符转为空格。

■ 测试数据

```
{
  "key": "this is\tlts"
}
```

■ 加工规则

```
e_set("str_expandtabs", str_expandtabs(v("key")))
```

■ 加工结果

```
key: this is\tlts
str_expandtabs: this is lts
```

b. 示例2：把key中的\t字符转为空格，并填充指定空格的数量。

■ 测试数据

```
{
  "key": "this is\tlts"
}
```

■ 加工规则

```
e_set("str_expandtabs", str_expandtabs(v("key"), 5))
```

■ 加工结果

```
key: this is\tlts
str_expandtabs: this is   lts
```

10.2.1.10.5 字符集判断

str_isalnum

判断字符串是否仅由字母和数字组成。

- 函数格式

```
str_isalnum(value)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "value": "ltsv5"
}
```

- 加工规则

```
e_set("str_isalnum", str_isalnum(v("value")))
```

- 加工结果

```
value: ltsv5
str_isalnum: true
```

str_isalpha

判断字符串是否仅由字母组成。

- 函数格式

str_isalpha(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "value": "ltsv5"
}
```

- 加工规则

```
e_set("str_isalpha", str_isalpha(v("value")))
```

- 加工结果

```
value: ltsv5
str_isalpha: false
```

str_isascii

判断字符串是否在ASCII中。

- 函数格式

str_isascii(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "value": "{ltsv5}#@ "
}
```

- 加工规则

```
e_set("str_isascii", str_isascii(v("value")))
```

- 加工结果

```
value: {ltsv5}#@
str_isascii: true
```

str_isdecimal

判断字符串是否仅包含十进制字符。

- 函数格式

str_isdecimal(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "value": "111"
}
```

- 加工规则

```
e_set("str_isdecimal", str_isdecimal(v("value")))
```

- 加工结果

```
value: 111
str_isdecimal: true
```

str_isdigit

判断字符串是否仅由数字组成。

- 函数格式

str_isdigit(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "value": "111"
}
```

- 加工规则

```
e_set("str_isdigit", str_isdigit(v("value")))
```

- 加工结果

```
value: 111
str_isdigit: true
```

str_isidentifier

字符串是否是有效的Python标识符，也可以用来判断变量名是否合法。

- 函数格式

str_isidentifier(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "key": "int"
}
```

- 加工规则

```
e_set("str_isidentifier", str_isidentifier(v("key")))
```

- 加工结果

```
key: int
str_isidentifier: true
```

str_islower

判断字符串是否由小写字母组成。

- 函数格式

str_islower(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "key": "lower"
}
```

- 加工规则

```
e_set("str_islower", str_islower(v("key")))
```

- 加工结果

```
key: lower
str_islower: true
```

str_isnumeric

判断字符串是否由数字组成。

- 函数格式

str_isnumeric(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "key": "123W"
}
```

- 加工规则

```
e_set("str_isnumeric", str_isnumeric(v("key")))
```

- 加工结果

```
key: 123W
str_isnumeric: false
```

str_isprintable

判断字符串中是否所有字符都是可打印字符。

- 函数格式

str_isprintable(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "key": "@#11!"
}
```

- 加工规则

```
e_set("str_isprintable", str_isprintable(v("key")))
```

- 加工结果

```
key: @#11!
str_isprintable: true
```

str_isspace

判断字符串是否仅由空格字符组成。

- 函数格式

str_isspace(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "key": "@#11!"
}
```

- 加工规则

```
e_set("str_isspace", str_isspace(v("key")))
```

- 加工结果

```
key: @#11!
str_isspace: truefalse
```

str_istitle

判断字符串中所有单词的拼写首字母是否为大写，且其他字母为小写。

- 函数格式

str_istitle(value)

- 参数说明

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- 返回结果

true/false。

- 函数示例

- 测试数据

```
{
  "key": "Lts Is A Log Service"
}
```

- 加工规则

```
e_set("str_istitle", str_istitle(v("key")))
```

- 加工结果

```
key: Lts Is A Log Service
str_istitle: true
```

str_isupper

判断字符串中所有的字母是否都为大写。

- **函数格式**

```
str_isupper(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

true/false。

- **函数示例**

- 测试数据

```
{  
  "key": "LTS"  
}
```

- 加工规则

```
e_set("str_isupper", str_istitle(v("key")))
```

- 加工结果

```
key: LTS  
str_isupper: true
```

str_uuid

随机生成UUID。

- **函数格式**

```
str_uuid(lower=true)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
lower	Boolean	否	生成的UUID中的字母是否是小写。默认为true表示小写。

- **返回结果**

true/false。

- **函数示例**

- 测试数据：无

- 加工规则

```
e_set("UUID", str_uuid())
```

- 加工结果

```
UUID: 1acf7e1d-5a6b-4267-b7c1-874c8b70745d
```

10.2.1.11 算术函数

本文介绍算术函数的语法规则，包括参数解释、函数示例等。

函数列表

如果值为负数，请使用op_neg(positive value)函数，例如：要表示-1，请使用op_neg(1)。

表 10-4 算术函数列表

类型	函数	说明
多值计算比较	op_sum	对传入值进行求和操作。
基础计算	op_abs	对传入值进行绝对值操作。
	op_div_floor	对传入值进行整除操作。
	op_div_true	对传入值进行除法操作。
	op_pow	对值进行幂值计算操作。
	op_mul	对传入值进行乘法运算。
	op_neg	计算传入值的相反数。
	op_mod	对传入值进行模计算。
	op_sub	对传入值进行减法运算。
	op_round	对传入值进行四舍五入操作。
数学计算	mat_ceil	对传入值进行向上取整操作。
	mat_exp	以常数e为底的指数函数。
	mat_fabs	计算传入值的绝对值。
	mat_floor	对传入值进行向下取整操作。
	mat_log	计算传入值的对数。
	mat_log10	计算以10为基数的传入值的对数。

op_sum

对传入值进行求和操作。

- 函数格式
op_sum(value1, value2, ...)
- 参数说明

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- 返回结果

返回多个传入值的求和结果。

- **函数示例**

计算course_price和goods_price总价。

- 测试数据

```
{
  "course_price": 12,
  "goods_price": 2
}
```

- 加工规则

```
e_set("account", op_sum(v("course_price"), v("goods_price")))
```

- 加工结果

```
course_price: 12
goods_price: 2
account: 14
```

op_abs

对传入值进行绝对值操作。

- **函数格式**

```
op_abs(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**

返回传入值的绝对值。

- **函数示例**

- 测试数据

```
{
  "course_price": -4
}
```

- 加工规则

```
e_set("op_abs", op_abs(v("course_price")))
```

- 加工结果

```
course_price: -4
op_abs: 4
```

op_div_floor

对传入值进行整除操作。

- **函数格式**

```
op_div_floor(value1, value2)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**

返回值1与值2整除的结果。

- **函数示例**

根据course_price字段和count字段计算单价。

- **测试数据**

```
{
  "course_price": 4,
  "count": 2
}
```

- **加工规则**

```
e_set("op_div_floor", op_div_floor(v("course_price"), v("count")))
```

- **加工结果**

```
course_price: 4
count: 2
op_div_floor: 2
```

op_div_true

对传入值进行除法操作。

说明 该函数支持自动转换数据类型，输入string或者int类型的数据即可。

- **函数格式**

```
op_div_true(value1, value2)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字字符串/ Number	是	待计算的值。
value2	数字字符串/ Number	是	待计算的值。

- **返回结果**

返回值1/值2的结果。

- **函数示例**

a. 示例1：根据fruit_price和count字段计算单价。

- **测试数据**

```
{
  "fruit_price": 9,
  "count": 2
}
```

- **加工规则**

```
e_set("op_div_true", op_div_true(v("fruit_price"), v("count")))
```

- **加工结果**

```
fruit_price: 9
count: 2
op_div_true: 4.5
```

b. 示例2：计算加速度（四舍五入），计算公式为 $a = (\text{one\_speed} - \text{two\_speed}) / \text{time}$ 。

■ 测试数据

```
{
  "one_speed": 9,
  "two_speed": 2,
  "time": 3
}
```

■ 加工规则

```
e_set("a", op_round(op_div_true(op_sub(v("one_speed"), v("two_speed")), v("time")), 2))
```

■ 加工结果

```
a:2.33
one_speed:9
time:3
two_speed:2
```

op_pow

对值进行幂值计算操作。

- 函数格式

```
op_pow(value1, value2)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- 返回结果

返回值1的值2次方的结果。

- 函数示例

计算course的pow次方的值。

- 测试数据

```
{
  "course": 100,
  "pow": 2
}
```

- 加工规则

```
e_set("pow_course", op_pow(v("course"), v("pow")))
```

- 加工结果

```
course: 100
pow: 2
pow_course: 10000
```

op_mul

对传入值进行乘法运算。

- 函数格式

```
op_mul(value1, value2)
```

- 参数说明

参数名称	参数类型	是否必填	说明

value1	数字、字符串、元祖、列表等	是	待计算的值。
value2	数字	是	待计算的值。

- **返回结果**

- 若值1和值2为数字，返回相乘的结果。
- 若值1和值2为字符串、元祖、列表等，返回扩大倍数的原类型。

- **函数示例**

a. 示例1：根据course和price字段计算总价。

- **测试数据**

```
{
  "course": 10,
  "price": 23
}
```

- **加工规则**

```
e_set("account", op_mul(ct_int(v("course")), ct_int(v("price"))))
```

- **加工结果**

```
course: 10
price: 23
account: 230
```

b. 示例2：将course字段扩大3倍。

- **测试数据**

```
{
  "course": "abc"
}
```

- **加工规则**

```
e_set("course", op_mul(v("course"), 3))
```

- **加工结果**

```
course: abcabcab
```

op_neg

计算传入值的相反数。

- **函数格式**

```
op_neg(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**

返回传入值的相反数。

- **函数示例**

- 测试数据

```
{
  "course": -100
}
```

- 加工规则
e_set("account", op_neg(v("course_price")))
- 加工结果
course: -100
account: 100

op_mod

对传入值进行模计算。

- **函数格式**
op_mod(value1, value2)
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**
返回值1对值2取模的结果。
- **函数示例**

- 测试数据

```
{
  "course": 4,
  "count": 3
}
```
- 加工规则
e_set("op_mod", op_mod(v("course"), v("count")))
- 加工结果
course: 4
count: 3
op_mod: 1

op_sub

对传入值进行减法运算。

- **函数格式**
op_sub(value1, value2)
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**
返回值1减去值2的结果。

- **函数示例**

计算count减count_apple的结果。

- 测试数据

```
{
  "count": 6,
  "count_apple": 3
}
```

- 加工规则

```
e_set("sub_number", op_sub(v("count"),v("count_apple")))
```

- 加工结果

```
count: 6
count_apple: 3
sub_number: 3
```

op_round

对传入值进行四舍五入操作。

- **函数格式**

op_round(value, number)

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。
number	数字	是	表示四舍五入后保留的小数点位数，默认值为0。

- **返回结果**

返回传入值的四舍五入的结果。

- **函数示例**

返回price小数点后1位数。

- 测试数据

```
{
  "price": 4.56
}
```

- 加工规则

```
e_set("round_price", op_round(v("price"),1))
```

- 加工结果

```
price: 4.56
round_price: 4.6
```

mat_ceil

对传入值进行向上取整操作。

- **函数格式**

mat_ceil(value)

- **参数说明**

参数名称	参数类型	是否必填	说明

value	数字或数字字符串	是	待计算的值。

- **返回结果**
返回大于或者等于指定值的最小整数。
- **函数示例**
 - 测试数据

```
{  
  "price": 4.1  
}
```
 - 加工规则

```
e_set("mat_ceil", mat_ceil(v("price")))
```
 - 加工结果

```
price: 4.1  
mat_ceil: 5
```

mat_exp

以常数e为底的指数函数，返回e的次幂结果。

- **函数格式**
mat_exp(value)
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**
返回e的次幂结果。
- **函数示例**
 - 测试数据

```
{  
  "number": 2  
}
```
 - 加工规则

```
e_set("e_x", mat_exp(v("number")))
```
 - 加工结果

```
number: 2  
e_x: 7.38905609893065
```

mat_fabs

计算传入值的绝对值。

- **函数格式**
mat_fabs(value)
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**

返回传入值的绝对值。

- **函数示例**

- 测试数据

```
{
  "course_price": -10
}
```

- 加工规则

```
e_set("mat_fabs", mat_fabs(v("course_price")))
```

- 加工结果

```
course_price: -10
mat_fabs: 10.0
```

mat_floor

对传入值进行向下取整操作。

- **函数格式**

```
mat_floor(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**

返回小于或者等于指定值的最大整数。

- **函数示例**

- 测试数据

```
{
  "course_price": 4.9
}
```

- 加工规则

```
e_set("mat_floor", mat_floor(v("course_price")))
```

- 加工结果

```
course_price: 4.9
mat_floor: 4
```

mat_log

计算传入值的对数。

- **函数格式**

```
mat_log(value1,value2)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**
返回传入值的对数。
- **函数示例**
 - 测试数据

```
{
  "number1": 100,
  "number2": 10
}
```
 - 加工规则

```
e_set("mat_log", mat_log(v("number1"),v("number2")))
```
 - 加工结果

```
number1: 100
number2: 10
mat_log: 2.0
```

mat_log10

计算以10为基数的传入值的对数。

- **函数格式**
mat_log10(value)
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。
- **返回结果**
返回以10为基数的传入值的对数结果。
- **函数示例**
 - 测试数据

```
{
  "number": 100
}
```
 - 加工规则

```
e_set("number2", mat_log10(v("number")))
```
 - 加工结果

```
number: 100
numbe2: 2.0
```

10.2.1.12 转换函数

本文主要介绍操作符函数的语法规则，包括参数说明、函数示例等。

函数列表

表 10-5 转换函数列表

类型	函数名称	功能描述
基础类型转换	ct_int	将字段或表达式的值转换为整数。

类型	函数名称	功能描述
	ct_float	将字段或表达式的值转换为浮点数。
	ct_str	将字段或表达式的值转换为字符串。
	ct_bool	将字段或表达式值转换为布尔值。
数字转换	ct_chr	将字段或表达式的ANSI值、Unicode值转换为对应字符。
	ct_ord	将字段或表达式的字符转换为对应ANSI值、Unicode值。
	ct_hex	将字段或表达式的数值转换为十六进制数。
	ct_oct	将字段或表达式的数值转换为八进制数。
	ct_bin	将字段或表达式的数值转换为二进制数。
进制转换	bin2oct	将二进制数转换为八进制数。
	bin2hex	将二进制数转换为十六进制字符串。

ct_int

使用ct_int函数将字段或表达式的值转换为整数。

- 函数格式

ct_int(value, base=10)
- 参数说明

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。
base	Number	否	参数值所代表的进制，默认为十进制。例如 base=8，表示将八进制要转成十进制。
- 返回结果

返回整型数值。
- 函数示例
 - 示例1：将字符串转换成整型。
 - 测试数据

```
{  
  "number": 2  
}
```

- 加工规则

```
e_set("int_number", ct_int(v("number")))
```

- 加工结果

```
number: 2  
int_number: 2
```

b. 示例2：将十六进制转换成十进制。

- 测试数据

```
{  
  "number": "AB"  
}
```

- 加工规则

```
e_set("int_number", ct_int(v("number"),base=16))
```

- 加工结果

```
number: AB  
int_number: 171
```

ct_float

使用ct_float函数将字段或表达式的值转换为浮点数。

- 函数格式

```
ct_float(value)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- 返回结果

返回浮点类型数值。

- 函数示例

- 测试数据

```
{  
  "price": 2  
}
```

- 加工规则

```
e_set("price_float", ct_float(v("price")))
```

- 加工结果

```
price: 2  
price_float: 2.0
```

ct_str

使用ct_str函数将字段或表达式的值转换为字符串。

- 函数格式

```
ct_str(value)
```

- 参数说明

参数名称	参数类型	是否必填	说明

value	任意值	是	待转换的值。

- **返回结果**

返回字符串。

- **函数示例**

- 测试数据：无

- 加工规则

```
e_set("ct_str", ct_str(b'test byte'))
```

- 加工结果

```
ct_str: test byte
```

ct_bool

使用ct_bool函数将字段或表达式值转换为布尔值。

- **函数格式**

```
ct_bool(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意值	是	待转换的值。

- **返回结果**

返回布尔值。

- **函数示例**

- 测试数据

```
{  
  "num": 2  
}
```

- 加工规则

```
e_set("ct_bool", ct_bool(v("num")))
```

- 加工结果

```
num: 2  
ct_bool: true
```

ct_chr

使用ct_chr函数将字段或表达式的ANSI值、Unicode值转换为对应字符。

- **函数格式**

```
ct_chr(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- **返回结果**

返回chr类型对应的字符。

- **函数示例**

- 测试数据

```
{
  "num": 78
}
```

- 加工规则

```
e_set("ct_chr", ct_chr(v("number")))
```

- 加工结果

```
number: 78
ct_chr: N
```

ct_ord

使用ct_ord函数将字段或表达式的字符转换为对应ANSI值、Unicode值。

- **函数格式**

```
ct_ord(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	待转换的值，长度为1。

- **返回结果**

返回对应的ANSI值或Unicode值。

- **函数示例**

- 测试数据

```
{
  "world": "a"
}
```

- 加工规则

```
e_set("ct_ord", ct_ord(v("world")))
```

- 加工结果

```
world: a
ct_ord: 97
```

ct_hex

使用ct_hex函数将字段或表达式的数值转换为十六进制数。

- **函数格式**

```
ct_hex(value)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- **返回结果**

返回十六进制的数值。

- **函数示例**

- 测试数据

```
{
  "number": 123
}
```
- 加工规则

```
e_set("ct_hex", ct_hex(v("number")))
```
- 加工结果

```
number: 123
ct_hex: 0x7b
```

ct_oct

使用ct_oct函数将字段或表达式的数值转换为八进制数。

- **函数格式**

ct_oct(value)

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- **返回结果**

返回八进制的数值。

- **函数示例**

- 测试数据

```
{
  "number": 123
}
```
- 加工规则

```
e_set("ct_oct", ct_oct(v("number")))
```
- 加工结果

```
number: 123
ct_oct: 0o173
```

ct_bin

使用ct_bin将字段或表达式的数值转换为二进制数。

- **函数格式**

ct_bin(value)

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- **返回结果**

返回二进制的数值。

- **函数示例**

- 测试数据

```
{
  "number": 123
}
```

- 加工规则
e_set("ct_bin", ct_bin(v("number")))
- 加工结果
number: 123
ct_bin: 0b1111011

bin2oct

使用bin2oct函数将二进制数转换为八进制数。

- 函数格式
bin2oct(binary)
- 参数说明

参数名称	参数类型	是否必填	说明
binary	Binary	是	Binary类型的字符串。
- 返回结果
返回八进制的字符串。
- 函数示例
 - 测试数据

```
{
  "test": "test"
}
```
 - 加工规则
e_set("new",bin2oct(base64_decoding("ARi8WnFiLAAACHcAGkADV37Xs8BXftezgAdqwF9")))
 - 加工结果
test : test
new :
214274264705421300000002073400064044000325677327547401273755366340003552600575

bin2hex

使用bin2hex函数将二进制数转换为十六进制字符串。

- 函数格式
bin2hex(binary)
- 参数说明

参数名称	参数类型	是否必填	说明
binary	Binary	是	Binary类型的字符串。
- 返回结果
返回十六进制的字符串。
- 函数示例

- 测试数据

```
{
  "test": "test"
}
```
- 加工规则

```
e_set("new",bin2hex(base64_decoding("ARi8WnFiLAAACHcAGkADV37Xs8BXftezgAdqwF9")))
```
- 加工结果

```
test : test
new :0118bc5a71622c00000877001a09000d5dfb5ecf015dfb5ece001dab017d
```

10.2.1.13 操作符函数

本文介绍操作符函数的语法规则，包括参数解释、函数示例等。

函数列表

如果值为负数，请使用op_neg(positive value)函数，例如：您要表示-1，请使用op_neg(1)。

表 10-6 操作符函数列表

类型	函数	说明
条件判断函数	op_if	根据判断条件返回不同表达式的值。
	op_ifnull	返回第一个值不为None的表达式的值。
	op_coalesce	返回第一个值不为None的表达式的值。
	op_nullif	如果表达式1等于表达式2，返回None。否则返回表达式1的值。
	op_and	使用逻辑运算and，对任意类型值进行真假判断，所有参数值为真时返回true。
	op_not	使用逻辑运算not，对任意类型值进行真假判断，返回参数值的反义布尔值。
	op_or	使用逻辑运算or，对任意类型值进行真假判断。当任意参数值为真时返回true，所有参数值为假时返回false。
比较	op_eq	按照a==b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
	op_ge	按照a>=b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
	op_gt	按照a>b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
	op_le	按照a<=b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
	op_lt	按照a<b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。

类型	函数	说明
	<code>op_ne</code>	按照 <code>a!=b</code> 条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
容器判断	<code>op_len</code>	计算文本字符串中的字符数，可用于字符串和其他返回元组、列表、字典的表达式。
	<code>op_in</code>	判断字符串、元组、列表或字典中是否包含特定元素，返回true或false。
	<code>op_not_in</code>	判断字符串、元组、列表或字典中是否不包含特定元素，返回true或false。
	<code>op_slice</code>	对指定字符串、数组、元组进行截取。
	<code>op_index</code>	根据字符串、数组、元组的下标返回其对应的元素。
一般性多值操作	<code>op_add</code>	计算多个值的和，可以是字符串或者数字等。
	<code>op_max</code>	计算多个字段或表达式表示的数值的最大值。
	<code>op_min</code>	计算多个字段或表达式表示的数值的最小值。

op_if

根据判断条件返回不同表达式的值。

- **函数格式**
`op_if(condition, expression1, expression2)`
- **参数说明**

参数名称	参数类型	是否必填	说明
condition	任意	是	判断条件。如果该条件为非布尔值，系统将其采用真假判断。
expression1	任意	是	判断结果为true时，返回该表达式的值。
expression2	任意	是	判断结果为false时，返回该表达式的值。

- **返回结果**
返回相应的表达式的值。
- **函数示例**
 - a. 示例1：如果content为true，则把表达式1的值赋给test_if。
 - 测试数据

```
{
  "content": "hello"
}
```

- 加工规则
e_set("test_if", op_if(v("content"),"still origion content","replace this"))
 - 加工结果
content: hello
test_if: still origion content
- b. 示例2：如果content为false，则把表达式2的值赋给test_if。
- 测试数据
{
 "content": 0
}
 - 加工规则
e_set("test_if", op_if(ct_int(v("content", default=0)),"still origion content","replace this"))
 - 加工结果
content: 0
test_if: replace this

op_ifnull

返回第一个值不为None的表达式的值。

- 函数格式
op_ifnull(expression1, expression2, ...)
- 参数说明

参数名称	参数类型	是否必填	说明
expression1	任意	是	表达式1。
expression2	任意	是	表达式2。

- 返回结果
返回第一个值不为None的表达式的值。
 - 函数示例
- a. 示例1：

- 测试数据
{
 "test_if": "hello",
 "escape_name": "Etl"
}
- 加工规则
e_set("test_ifnull", op_ifnull(v("escape_name"),v("test_if")))
- 加工结果
test_if: hello
escape_name: Etl
test_ifnull: Etl

- b. 示例2：
- 测试数据
{
 "test_if": "hello",

```
"escape_name": "Etl"
}
```

- 加工规则
`e_set("test_ifnull", op_ifnull(v("test_if"),v("escape_name")))`
- 加工结果
test_if: hello
escape_name: Etl
test_ifnull: hello

op_coalesce

返回第一个值不为None的表达式的值。

- 函数格式
`op_coalesce(expression1, expression2, ...)`
- 参数说明

参数名称	参数类型	是否必填	说明
expression1	任意	是	表达式1。
expression2	任意	是	表达式2。

- 返回结果
返回第一个值不为None的表达式的值。
- 函数示例

a. 示例1:

- 测试数据

```
{
  "test_if": "hello",
  "escape_name": "Etl"
}
```
- 加工规则
`e_set("test_coalesce", op_coalesce(v("escape_name"),v("test_if")))`
- 加工结果
test_if: hello
escape_name: Etl
test_coalesce: Etl

b. 示例2:

- 测试数据

```
{
  "test_if": "hello",
  "escape_name": "Etl"
}
```
- 加工规则
`e_set("test_coalesce", op_coalesce(v("test_if"),v("escape_name")))`
- 加工结果
test_if: hello
escape_name: Etl
test_coalesce: hello

op_nullif

如果表达式1等于表达式2，返回None。否则返回表达式1的值。

- 函数格式

```
op_nullif(expression1, expression2)
```

- 参数说明

参数名称	参数类型	是否必填	说明
expression1	任意	是	表达式1。
expression2	任意	是	表达式2。

- 返回结果

如果表达式1和表达式2相等返回None，否则返回表达式1的值。

- 函数示例

a. 示例1:

- 测试数据

```
{
  "test_if": "hello",
  "escape_name": "Etl"
}
```

- 加工规则

```
e_set("test_ifnull", op_nullif(v("test_if"),v("escape_name")))
```

- 加工结果

```
test_if: hello
escape_name: Etl
test_ifnull: hello
```

b. 示例2:

- 测试数据

```
{
  "test_if": "hello",
  "escape_name": "hello"
}
```

- 加工规则

```
e_set("test_ifnull", op_nullif(v("content"),v("escape_name")))
```

- 加工结果

```
#因为content与escape_name内容一样，所以没有任何内容返回给test_isnull字段。
test_if: hello
escape_name: hello
```

op_and

使用逻辑运算and，对任意类型值进行真假判断，所有参数值为真时返回true。

- 函数格式

```
op_and(value1, value2, ...)
```

- 参数说明

参数名称	参数类型	是否必填	说明

value1	任意	是	运算值1。
value2	任意	是	运算值2。

- **返回结果**

- 所有参数值为真时返回true。
- 对任意类型值进行真假判断。

- **函数示例**

- a. 示例1:

- 测试数据

```
{  
  "number1": 123,  
  "number2": 234  
}
```

- 加工规则

```
e_set("op_and", op_and(v("number1"),v("number2")))
```

- 加工结果

```
number1: 123  
number2: 234  
op_and: true
```

- b. 示例2:

- 测试数据

```
{  
  "number1": 0,  
  "number2": 234  
}
```

- 加工规则

```
e_set("op_and", op_and(v("number1"),v("number2")))
```

- 加工结果

```
number1: 0  
number2: 234  
op_and: false
```

- c. 示例3:

- 测试数据

```
{  
  "ctx1": "false",  
  "ctx2": 234  
}
```

- 加工规则

```
e_set("op_and", op_and(v("ctx1"),v("ctx2")))
```

- 加工结果

```
ctx1: false  
ctx2: 234  
op_and: true
```

- d. 示例4:

- 测试数据

```
{  
  "ctx1": "true",  
}
```

```
"ctx2": 234
}
```

- 加工规则
e_set("op_and", op_and(v("ctx1"),v("ctx2")))
- 加工结果
ctx1: true
ctx2: 234
op_and: true

op_not

使用逻辑运算not，对任意类型值进行真假判断，返回表达式值的反义布尔值。

- 函数格式

op_not(expression)

- 参数说明

参数名称	参数类型	是否必填	说明
expression	任意	是	表达式。

- 返回结果

- 返回表达式值的反义布尔值。
- 对任意类型值进行真假判断。

- 函数示例

a. 示例1:

- 测试数据
{
 "ctx1": "true"
}
- 加工规则
e_set("op_not", op_not(v("ctx1")))
- 加工结果
ctx1: true
op_not: false

b. 示例2:

- 测试数据
{
 "ctx1": 345
}
- 加工规则
e_set("op_not", op_not(v("ctx1")))
- 加工结果
ctx1: 345
op_not: false

c. 示例3:

- 测试数据

```
{  
  "ctx1": 0  
}
```

- 加工规则
e_set("op_not", op_not(ct_int(v("ctx1"))))
- 加工结果
ctx1: 0
op_not: true

d. 示例4:

- 测试数据
{
 "ctx1": "ETL"
}
- 加工规则
e_set("op_not", op_not(v("ctx1")))
- 加工结果
ctx1: ETL
op_not: false

e. 示例5:

- 测试数据
{
 "ctx1": "None"
}
- 加工规则
e_set("op_not", op_not(v("ctx1")))
- 加工结果
ctx1: None
op_not: false

op_or

使用逻辑运算or，对任意类型值进行真假判断。当任意表达式的值为真时返回true，所有表达式值为假时返回false。

- 函数格式
op_or(expression1, expression2, ...)

- 参数说明

参数名称	参数类型	是否必填	说明
expression1	任意	是	表达式1。
expression2	任意	是	表达式2。

- 返回结果
 - 任意表达式的值为真时返回true，所有表达式的值为假时返回false。
 - 对任意类型值进行真假判断。

- 函数示例

a. 示例1:

■ 测试数据

```
{
  "ctx1": 123,
  "ctx2": 234
}
```

■ 加工规则

```
e_set("op_or", op_or(v("ctx1"),v("ctx2")))
```

■ 加工结果

```
ctx1: 123
ctx2: 234
op_or: true
```

b. 示例2:

■ 测试数据

```
{
  "ctx1": 0,
  "ctx2": 234
}
```

■ 加工规则

```
e_set("op_or", op_or(v("ctx1"),v("ctx2")))
```

■ 加工结果

```
ctx1: 0
ctx2: 234
op_or: true
```

c. 示例3:

■ 测试数据

```
{
  "ctx1": "ETL",
  "ctx2": "aa"
}
```

■ 加工规则

```
e_set("op_or", op_or(v("ctx1"),v("ctx2")))
```

■ 加工结果

```
ctx1: ETL
ctx2: aa
op_or: true
```

d. 示例4:

■ 测试数据

```
{
  "ctx1": "true",
  "ctx2": "false"
}
```

■ 加工规则

```
e_set("op_or", op_or(v("ctx1"),v("ctx2")))
```

■ 加工结果

```
ctx1: true
ctx2: false
op_or: true
```

e. 示例5:

■ 测试数据

```
{
  "ctx1": 0,
  "ctx2": "false"
}
```

- 加工规则
e_set("op_or", op_or(ct_int(v("ctx1")),v("ctx2")))
- 加工结果
ctx1: 0
ctx2: false
op_or: true

f. 示例6:

- 测试数据

```
{
  "ctx1": 124,
  "ctx2": "true"
}
```
- 加工规则
e_set("op_or", op_or(v("ctx1"),v("ctx2")))
- 加工结果
ctx1: 124
ctx2: true
op_or: true

op_eq

按照a==b条件进行计算，返回true或false。

- 函数格式
op_eq(value1, value2)

参数说明

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- 返回结果
如果值1与值2相等返回true，否则返回false。
- 函数示例

a. 示例1:

- 测试数据

```
{
  "content": "hello",
  "ctx": "hello"
}
```
- 加工规则
e_set("test_eq", op_eq(v("content"),v("ctx")))
- 加工结果

```
content: hello
ctx: hello
test_eq: true
```

b. 示例2:

■ 测试数据

```
{
  "content": "hello",
  "ctx": "ctx"
}
```

■ 加工规则

```
e_set("test_eq", op_eq(v("content"),v("ctx")))
```

■ 加工结果

```
content: hello
ctx: ctx
test_eq: false
```

op_ge

按照a>=b条件进行计算，返回true或false。

• 函数格式

```
op_ge(value1, value2)
```

• 参数说明

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

• 返回结果

如果值1大于等于值2返回true，否则返回false。

• 函数示例

a. 示例1：假如apple_price的值大于等于orange_price的值，则返回true。

■ 测试数据

```
{
  "apple_price": 16,
  "orange_price": 14
}
```

■ 加工规则

```
e_set("test_ge", op_ge(ct_int(v("apple_price")),ct_int(v("orange_price"))))
```

■ 加工结果

```
apple_price: 16
orange_price: 14
test_ge: true
```

b. 示例2：假如apple_price的值小于orange_price的值，则返回false。

■ 测试数据

```
{
  "apple_price": 12,
  "orange_price": 14
}
```

- 加工规则
e_set("test_ge", op_ge(ct_int(v("apple_price")), ct_int(v("orange_price"))))
- 加工结果
apple_price: 12
orange_price: 14
test_ge: false

op_gt

按照a>b条件进行计算，返回true或false。

- 函数格式

op_gt(value1, value2)

- 参数说明

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- 返回结果

如果值1大于值2返回true，否则返回false。

- 函数示例

- a. 示例1：判断old_number取值是否大于young_number取值，大于返回true否则返回false。

- 测试数据
{
 "old_number": 16,
 "young_number": 14
}

- 加工规则
e_set("op_gt", op_gt(ct_int(v("old_number")), ct_int(v("young_number"))))

- 加工结果
old_number: 16
young_number: 14
test_ge: true

- b. 示例2：判断priority取值是否大于price取值，大于返回true否则返回false。

- 测试数据
{
 "priority": 14,
 "price": 16
}

- 加工规则
e_set("op_gt", op_gt(ct_int(v("priority")), ct_int(v("price"))))

- 加工结果
priority: 14
price: 16
test_ge: false

op_le

按照 $a \leq b$ 条件进行计算，返回true或false。

- 函数格式

```
op_le(value1, value2)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- 返回结果

如果值1小于等于值2返回true，否则返回false。

- 函数示例

a. 示例1：如果priority的值小于等于price的值，返回true否则返回false。

- 测试数据

```
{
  "priority": 16,
  "price": 14
}
```

- 加工规则

```
e_set("op_le",op_le(ct_int(v("priority")),ct_int(v("price"))))
```

- 加工结果

```
priority: 16
price: 14
op_le: false
```

b. 示例2：如果priority的值小于等于price的值，返回true否则返回false。

- 测试数据

```
{
  "priority": 14,
  "price": 16
}
```

- 加工规则

```
e_set("op_le",op_le(ct_int(v("priority")),ct_int(v("price"))))
```

- 加工结果

```
priority: 14
price: 16
test_ge: true
```

op_lt

按照 $a < b$ 条件进行计算，返回true或false。

- 函数格式

```
op_lt(value1, value2)
```

- 参数说明

参数名称	参数类型	是否必填	说明

value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- **返回结果**
如果值1小于值2返回true，否则返回false。
- **函数示例**
 - a. 示例1：如果priority的值小于price的值，返回true否则返回false。
 - 测试数据

```
{
  "priority": 16,
  "price": 14
}
```
 - 加工规则

```
e_set("op_lt",op_lt(ct_int(v("priority")),ct_int(v("price"))))
```
 - 加工结果

```
priority: 16
price: 14
op_lt: false
```
 - b. 示例2：如果priority的值小于price的值，返回true否则返回false。
 - 测试数据

```
{
  "priority": 14,
  "price": 15
}
```
 - 加工规则

```
e_set("op_lt",op_lt(ct_int(v("priority")),ct_int(v("price"))))
```
 - 加工结果

```
priority: 14
price: 15
op_lt: true
```

op_ne

按照a!=b条件进行计算，返回true或false。

- **函数格式**
op_ne(value1, value2)
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- **返回结果**
如果值1不等于值2返回true，否则返回false。
- **函数示例**

a. 示例1:

■ 测试数据

```
{
  "priority": 16,
  "price": 14
}
```

■ 加工规则

```
e_set("op_ne",op_ne(ct_int(v("priority")),ct_int(v("price"))))
```

■ 加工结果

```
priority: 16
price: 14
op_ne: true
```

b. 示例2:

■ 测试数据

```
{
  "priority": 14,
  "price": 14
}
```

■ 加工规则

```
e_set("op_ne",op_ne(ct_int(v("priority")),ct_int(v("price"))))
```

■ 加工结果

```
priority: 14
price: 14
op_ne: false
```

op_len

计算文本字符串中的字符数，可用于字符串和其他返回元组、列表、字典的表达式。

- 函数格式

```
op_len(value )
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	字符串、元组、列表或字典等	是	运算值。

- 返回结果

返回字段的长度。

- 函数示例

- 测试数据

```
{
  "content": "I,love,this,world"
}
```

- 加工规则

```
e_set("op_len",op_len(v("content")))
```

- 加工结果

```
content: I,love,this,world
op_len: 17
```

op_in

判断字符串、元组、列表或字典中是否包含特定元素，返回true或false。

- 函数格式

op_in(value1, value2)
- 参数说明

参数名称	参数类型	是否必填	说明
value1	字符串、元组、列表或字典等	是	字符串、元组、列表或者字典等。
value2	任意	是	判断的元素。

说明 函数中字符串、元组、列表或字典参数在前，元素在后。

- 返回结果

如果字符串、元组、列表或字典a中包含元素b返回true，否则返回false。
- 函数示例
 - 测试数据

{
 "list": [1, 3, 2, 7, 4, 6],
 "num2": 2
}
 - 加工规则

e_set("op_in",op_in(v("list"),v("num2")))
 - 加工结果

list: [1, 3, 2, 7, 4, 6]
num2: 2
op_in: true

op_not_in

判断字符串、元组、列表或字典中是否不包含特定元素，返回true或false。

- 函数格式

op_not_in(value1, value2)
- 参数说明

参数名称	参数类型	是否必填	说明
value1	字符串、元组、列表或字典等	是	字符串、元组、列表或者字典等。
value2	任意	是	判断的元素。

说明 函数中字符串、元组、列表或字典参数在前，元素在后。

- 返回结果

如果字符串、元组、列表或字典中不包含元素返回true，否则返回false。

- 函数示例

- 测试数据

```
{
  "list": [1, 3, 2, 7, 4, 6],
  "num2": 12
}
```

- 加工规则

```
e_set("op_not_in",op_not_in(v("list"),v("num2")))
```

- 加工结果

```
list: [1, 3, 2, 7, 4, 6]
num2: 12
op_not_in: true
```

op_slice

对指定字符串、数组、元组进行截取。

- 函数格式

```
op_slice(value, start=0, end=None, step=None)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value	String	是	函数要切片的值。
start	Num	否	截取的起始位置，默认为位置0。
end	Num	否	截取的结束位置，不包含该位置，默认为字符串结尾位置。
step	Num	否	每次截取的长度。

- 返回结果

返回提取后的字符串。

- 函数示例

a. 示例1：对word字段从起点开始进行截取，结尾为2。

- 测试数据

```
{
  "word": "I,love,this,world"
}
```

- 加工规则

```
e_set("op_slice",op_slice(v("word"),2))
```

- 加工结果

```
word: I,love,this,world
op_slice: I,
```

b. 示例2：对word字段从位置2到位置9进行截取，步长为1。

- 测试数据

```
{
  "word": "I,love,this,world"
}
```

- 加工规则
`e_set("op_slice",op_slice(v("word"),2,9,1))`
- 加工结果
word: I,love,this,world
op_slice: love,th

op_index

根据字符串、数组、元组的下标返回其对应的元素。

- 函数格式

`op_index(value, index)`

- 参数说明

参数名称	参数类型	是否必填	说明
value	String	是	字符串、数组、元组等。
index	Num	否	需要传入的字符串、数组或元组的下标。

- 返回结果

返回下标对应的元素。

- 函数示例

a. 示例1：返回word字段下标为0的元素。

- 测试数据
`{
 "word": "I,love,this,world"
}`
- 加工规则
`e_set("op_index",op_index(v("word"),0))`
- 加工结果
word: I,love,this,world
op_index: I

b. 示例2：返回word字段下标为3的元素。

- 测试数据
`{
 "word": "I,love,this,world"
}`
- 加工规则
`e_set("op_index",op_index(v("word"),3))`
- 加工结果
word: I,love,this,world
op_index: o

op_add

计算多个值的和，可以是字符串或者数字等。

- **函数格式**

```
op_add(value1, value2, ...)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	字符串、元组、列表或字典等	是	运算值1。
value2	必须与值1一样	是	运算值2。

- **返回结果**

返回求和操作后的数值。

- **函数示例**

a. 示例1：计算price_orange和price_apple总金额。

- **测试数据**

```
{
  "price_orange": 2,
  "price_apple": 13
}
```

- **加工规则**

```
e_set("account", op_add(ct_int(v("price_orange")), ct_int(v("price_apple"))))
```

- **加工结果**

```
price_orange: 2,
price_apple: 13,
account: 15
```

b. 示例2：统计bytes_in和bytes_out的和。

- **测试数据**

```
{
  "bytes_in": 214,
  "bytes_out": 123
}
```

- **加工规则**

```
e_set("total_bytes", op_add(ct_int(v("bytes_in")), ct_int(v("bytes_out"))))
```

- **加工结果**

```
bytes_in: 214
bytes_out: 123
total_bytes: 337
```

c. 示例3：给网址添加HTTPS头。

- **测试数据**

```
{
  "host": "xx.com"
}
```

- **加工规则**

```
e_set("website", op_add("https://", v("host")))
```

- **加工结果**

```
host: xx.com
website: https://xx.com
```

op_max

计算多个字段或表达式表示的数值的最大值。

- 函数格式

```
op_max(value1, value2, ...)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1一样	是	运算值2。

- 返回结果

返回多个数值中的最大值。

- 函数示例

- 测试数据

```
{
  "price_orange": 2,
  "priority_apple": 13
}
```

- 加工规则

```
e_set("max_price", op_max(ct_int(v("price_orange")),ct_int(v("priority_apple"))))
```

- 加工结果

```
price_orange: 2
priority_apple: 13
max_price: 13
```

op_min

计算多个字段或表达式表示的数值的最小值。

- 函数格式

```
op_min(value1, value2, ...)
```

- 参数说明

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1一样	是	运算值2。

- 返回结果

返回多个数值中的最小值。

- 函数示例

- 测试数据

```
{
  "price_orange": 2,
  "priority_apple": 13
}
```

- 加工规则

```
e_set("op_min", op_min(ct_int(v("price_orange")),ct_int(v("priority_apple"))))
```

- 加工结果

price_orange: 2
priority_apple: 13
op_min: 2

10.2.1.14 事件检查函数

本文介绍事件检查函数的语法规则，包括参数解释、函数示例等。

函数列表

类型	函数	说明
基本方法	<code>e_has</code>	判断日志字段是否存在。
	<code>e_not_has</code>	判断日志字段是否不存在。支持和其他函数组合使用。
表达式函数	<code>e_search</code>	提供一种简化，类似Lucene语法的事件搜索方式。支持和其他函数组合使用。
	<code>e_match</code>	判断当前日志字段的值是否满足正则表达式。支持和其他函数组合使用。
	<code>e_match_any</code>	判断当前日志字段的值是否满足正则表达式，任意字段匹配返回true，否则返回false。
	<code>e_match_all</code>	判断当前日志字段的值是否满足正则表达式，所有字段匹配返回true，否则返回false。

同时，事件检查函数可以与如下表达式函数配合使用：

类型	函数	说明
基本判断	<code>op_and</code>	逻辑and运算。
	<code>op_or</code>	逻辑or运算。
	<code>op_not</code>	逻辑not运算。
	<code>op_nullif</code>	判断两个表达式的取值。
	<code>op_ifnull</code>	返回第一个值不为None的表达式的值。
	<code>op_coalesce</code>	返回第一个值不为None的表达式的值。

e_has

判断字段是否存在。

- 函数格式

e_has("key")
- 参数说明

参数名称	参数类型	是否必填	说明
key	String	是	日志的字段名。

- **返回结果**

字段存在返回true，不存在返回false。

- **函数示例**

判断日志是否存在content字段，存在则保留，不存在则丢弃。

- 测试数据

```
{  
  "content": 123  
}
```

- 加工规则

```
e_keep(e_has("content"))
```

- 加工结果

```
content: 123
```

e_not_has

判断字段是否不存在。

- **函数格式**

```
e_not_has("key")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
key	String	是	字段名称。

- **返回结果**

字段不存在返回true，存在返回false。

- **函数示例**

判断日志是否存在content字段，不存在则保留该日志，否则丢弃该日志。

- 测试数据

```
{  
  "content": 123  
}
```

- 加工规则

```
e_if_else(e_not_has("content"),e_keep(),e_drop())
```

- 加工结果

日志被丢弃。

- **更多参考**

支持和其他函数组合使用。

e_search

提供一种简化，类似Lucene语法的事件搜索方式。该功能在邀测期间，e_search函数处理日志流量比较小，请谨慎使用。

- **函数格式**

```
e_search(querystring)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
querystring	String	是	查询字符串，用于快速过滤日志的查询字符串。

- **返回结果**

满足条件返回true，否则返回false。

- **函数参考示例**

```
# 全文
e_search("active error") # 全文：两个子串是OR关系，进行搜索。
e_search("active error") # 全文：一个子串搜索。
# 字段:字符串
e_search("status: active") # 单词搜索。
e_search('author: "john smith"') # 带空格子串搜索。
e_search('field: active error') # 相当于field:active OR "error"。
# 完全匹配
e_search('author== "john smith"')
# 通配符搜索，星号(*)匹配零个或多个字符，半角问号(?)匹配一个字符。
e_search("status: active*test") # active*test中仅包含星号(*)，可以不使用双引号(" ")包裹。
e_search("status: active?good") # active?good中仅包含半角问号(?)，可以不使用双引号(" ")包裹。
e_search("status== ac*tive?good") # 完全匹配。
# 搜索值转义，星号(*)或问号(?)需要使用反斜线(\)转义。
e_search('status: "\*\?()[]:=") # "\*\?()[]:=中包含特殊字符，需要使用双引号(" ")包裹，除了星号(*)、半角问号(?)和反斜线(\)需要转义外，其他不用转义。
e_search("status: active\*test") # active\*test中仅包含星号(*)，可以不使用双引号(" ")包裹。
e_search("status: active\?test") # active\?test中仅包含半角问号(?)，可以不使用双引号(" ")包裹。
# 字段名转义
e_search("\*(1+1)\?: abc") # 字段名不能用双引号(" ")包裹，特殊字符用反斜线(\)转义。
e_search("__tag_\:_container_name_: abc") # 用反斜线(\)转义。
e_search("中文字段: abc") # 直接写中文。
# 正则匹配
e_search('content~="正则表达式") # 正则匹配。
# 数字
e_search('count: [100, 200]') # >=100 and <=200
e_search('count: [, 200]') # <=200
e_search('count: [200, *]') # >=200
e_search('age >= 18') # >= 18
e_search('age > 18') # > 18
# 使用关系运算符
e_search("abc OR xyz") # 关系运算符不区分大小写，OR和or效果一样。
e_search("abc and (xyz or zzz)")
e_search("abc and not (xyz and not zzz)")
e_search("abc && xyz") # and
e_search("abc || xyz") # or
e_search("abc || !xyz") # or not
```

- **函数示例**

- **测试数据**

```
{
  "desc": "john smith is a player"
}
```

- **加工规则**

```
e_if(e_search('desc: "john smith"'), e_set("found", "Yes"))
```

- **加工结果**

```
{
  "found": "Yes",
}
```

```
    "desc": "john smith is a player"
  }
```

- **更多参考**

支持和其他函数组合使用。

e_match

判断当前日志字段的值是否满足正则表达式。

- **函数格式**

```
e_match(key, regular_expression, full=true)
```

e_match函数通常与op_not、op_and或者op_or结合使用。

- **参数说明**

参数名称	参数类型	是否必填	说明
key	String	是	字段名。当字段不存在时，视为当前子条件不匹配。例如：字段f1不存在，那么e_match("f1", ...)结果为false。
regular_expression	String	是	正则表达式。如果需要使用纯粹字符串匹配时（非正则表达式），可以使用函数str_regex_escape修饰正则表达式。
full	Bool	否	是否完全匹配，默认为true表示完全匹配。

- **返回结果**

返回字段匹配的判断结果true或false。

- **函数示例**

判断字段k1的值是否为数字。

- **测试数据**

```
{
  "k1": 123
}
```

- **加工规则**

```
e_set("e_match", e_match("k1", r'\d+'))
```

- **加工结果**

```
k1: 123
match: true
```

- **更多参考**

支持和其他函数组合使用。

e_match_any

判断当前日志字段的值是否满足正则表达式，任意字段匹配返回true，否则返回false。

- **函数格式**

```
e_match_any(key1, regular_expression1, key2, regular_expression2, ..., full=true)
```

- 函数中key和regular_expression必须成对出现。

- `e_match_any`函数通常与`op_not`、`op_and`或者`op_or`结合使用。

- **参数说明**

参数名称	参数类型	是否必填	说明
key	String	是	字段名。当字段不存在时，视为当前子条件不匹配。例如：字段f1不存在，那么 <code>e_match_any("f1", ...)</code> 结果为false。
regular_expression	String	是	正则模式。如果需要使用纯粹字符串匹配时（非正则），可以使用函数 <code>str_regex_escape</code> 修饰正则。
full	Bool	否	是否完全匹配，默认为true表示完全匹配。

- **返回结果**

返回字段匹配的判断结果true或false。

- **函数示例**

`e_match_any`匹配，任意字段匹配则返回true。

- **测试数据**

```
{
  "k1": 123,
  "k2": "abc",
  "k3": "abc123"
}
```

- **加工规则**

```
e_set("match",e_match_any('k1', r'\d+', 'k2', '.'))
```

- **加工结果**

```
k1:123
k2:abc
k3:abc123
match:true
```

- **更多参考**

支持和其他函数组合使用。

`e_match_all`

判断当前日志字段的值是否满足正则表达式，所有字段匹配返回true，否则返回false。

- **函数格式**

```
e_match_all(key1, regular_expression1, key2, regular_expression2, ..., full=true)
```

- 函数中key和regular_expression必须成对出现。

- `e_match_all`函数通常与`op_not`、`op_and`或者`op_or`结合使用。

- **参数说明**

参数名称	参数类型	是否必填	说明

字段名	String	是	字段名。当字段不存在时，视为当前子条件不匹配。例如：字段f1不存在，那么e_match_all("f1", ...)结果为false。
正则	String	是	正则模式。如果需要使用纯粹字符串匹配时（非正则），可以使用函数str_regex_escape修饰正则。
full	Bool	否	是否完全匹配，默认为true表示完全匹配。

- **返回结果**
返回字段匹配的判断结果true或false。
- **函数示例**
 - 测试数据

```
{  
  "k1": 123,  
  "k2": "abc",  
  "k3": "abc123"  
}
```
 - 加工规则

```
e_set("match", e_match_all("k1", r"\d+", "k2", r"\d+"))
```
 - 加工结果

```
k1:123  
k2:abc  
k3:abc123  
match:false
```
 - **更多参考**
支持和其他函数组合使用。

10.2.2 全局操作函数

10.2.2.1 映射富化函数

本文介绍映射富化函数的语法规则，包括参数解释、函数示例等。

函数列表

类型	函数	说明
字段映射	e_dict_map	与目标数据字典进行映射，根据输入的字段映射一个新字段。支持和其他函数组合使用。
	e_table_map	与目标表格进行映射，根据输入的字段名称返回字段值。支持和其他函数组合使用。
搜索映射	e_search_dict_map	对关键字（查询字符串）以及其匹配的值的字典数据进行映射。支持和其他函数组合使用。
	e_search_table_map	对某列（查询字符串）以及其匹配的值的表格数据进行映射。

e_dict_map

与目标数据字典进行映射，根据输入的字段映射一个新字段。

- 函数格式

```
e_dict_map(data, field, output_field, case_insensitive=true, missing=None, mode="overwrite")
```

- 参数说明

参数名称	数据类型	是否必填	说明
data	Dict	是	目标数据字典。必须为标准的 {key01:value01,key01:value02,...} 格式，且必须是字符串。例如{"1": "TCP", "2": "UDP", "3": "HTTP", "*": "Unknown"}。
field	String或者String List	是	一个字段名或者多个字段名的列表。多个字段时： - 依次对匹配到的值进行映射。 - 如果匹配命中多条日志，且 mode 的取值为 overwrite 时，则最后一个会覆盖前面的结果。 - 当没有匹配到任何字段，则使用 missing 参数的值作为匹配值。
output_field	String	是	输出字段的名称。
case_insensitive	Boolean	否	匹配时大小写是否不敏感。 如果字典中存在同一个关键字的不同大小写，且 case_insensitive 为 true 时，会优先选择大小写完全匹配的值。如果没有，则随机选择一个。 - true（默认值）：不敏感。 - false：敏感
missing	String	否	无匹配字段时，将该参数的取值赋给输出字段 output_field。默认为 None 表示不做映射赋值操作。 如果字典中包含匹配星号（*），由于星号（*）的优先级高于 missing，此时 missing 参数不生效。
mode	String	否	字段的覆盖模式。默认为 overwrite。 取值为：fill, fill-auto, add, add-auto, overwrite, overwrite-auto

- 返回结果

返回附带新字段的日志。

- 函数示例
 - a. 示例1：根据测试数据中pro字段的值和目标数据字典，输出新字段protocol。
 - 测试数据

```
{
  "data": 123,
  "pro": 1
}
```
 - 加工规则

```
e_dict_map(
  {"1": "TCP", "2": "UDP", "3": "HTTP", "6": "HTTPS", "**": "Unknown"},
  "pro",
  "protocol",
)
```
 - 加工结果

```
data: 123
pro: 1
protocol: TCP
```
 - b. 示例2：根据测试数据中status字段的值和目标数据字典，输出新字段message。
 - 测试数据（三条测试日志）

```
{
  "status": "500"
}
{
  "status": "400"
}
{
  "status": "200"
}
```
 - 加工规则

```
e_dict_map({"400": "错误", "200": "正常", "**": "其他"}, "status", "message")
```
 - 加工结果

```
status: 500
message: 其他
status: 400
message: 错误
status: 200
message: 正常
```
- 更多参考
支持和其他函数组合使用。

e_table_map

与目标表格进行映射，根据输入的字段名称返回字段值。

- 函数格式

```
e_table_map(data, field, output_fields, missing=None, mode="fill-auto")
```
- 参数说明

参数名称	数据类型	是否必填	说明
data	Table	是	目标表格。

field	String、String List或Tuple List	是	日志中映射到表格的源字段。如果日志中不存在对应字段，则不进行任何操作。
output_fields	String、String List或Tuple List	是	映射后的字段。例如["province", "pop"]。
missing	String	否	无匹配字段时，将该参数的取值赋给输出字段output_fields。默认为None表示不做映射赋值操作。如果目标字段是多列，则missing可以是一个长度与目标字段数一致的默认值列表。 说明 如果表格中包含匹配星号（*），由于星号（*）的优先级高于missing，此时missing参数将不起作用。
mode	String	否	字段的覆盖模式。默认为fill-auto。

- **返回结果**

返回附带新字段值的日志。

- **函数示例**

a. 示例1：在映射表格中查找对应行，根据city字段返回province字段的值。

- **测试数据**

```
{
  "data": 123,
  "city": "nj"
}
```

- **加工规则**

```
e_table_map(
  tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"), "city", "province"
)
```

- **加工结果**

```
data: 123
city: nj
province: js
```

b. 示例2：在映射表格中查找对应行，根据city字段返回province字段和pop字段的值。

- **测试数据**

```
{
  "data": 123,
  "city": "nj"
}
```

- **加工规则**

```
e_table_map(
  tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),
  "city",
  ["province", "pop"],
)
```

■ 加工结果

```
data: 123
city: nj
province: js
pop: 800
```

- c. 示例3：使用tab_parse_csv函数构建映射表格，根据city字段返回province字段和pop字段的值。

■ 测试数据

```
{
  "data": 123,
  "city": "nj"
}
```

■ 加工规则

```
e_table_map(
  tab_parse_csv("city#pop#province\nnj#800#js\nsh#2000#sh", sep="#"),
  "city",
  ["province", "pop"],
)
```

■ 加工结果

```
data: 123
city: nj
province: js
pop: 800
```

- d. 示例4：使用tab_parse_csv函数构建映射表格，根据city字段返回province字段和pop字段的值。

■ 测试数据

```
{
  "data": 123,
  "city": "nj"
}
```

■ 加工规则

```
e_table_map(
  tab_parse_csv(
    "city,pop,province\nnj|,800|,js\nshang hai,2000|,SHANG,HAI|", quote="|"
  ),
  "city",
  ["province", "pop"],
)
```

■ 加工结果

```
data: 123
city: nj
province: js
pop: 800
```

- e. 示例5：日志匹配字段与映射表格中字段不一样。在映射表格中查找对应行，根据cty或city字段返回province字段的值。

■ 测试数据

```
{
  "data": 123,
  "city": "nj"
}
```

■ 加工规则

```
e_table_map(
  tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),
  [("city", "city")],
)
```

```
"province"  
)
```

- 加工结果

```
data: 123  
city: nj  
province: js
```

- f. 示例6：日志匹配字段与映射表格中字段不一样，并且对输出字段进行重命名。

- 测试数据

```
{  
  "data": 123,  
  "city": "nj"  
}
```

- 加工规则

```
e_table_map(  
  tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),  
  [("city", "city")],  
  [("province", "pro")],  
)
```

- 加工结果

```
data: 123  
city: nj  
pro: js
```

- g. 示例7：多个日志匹配字段。

- 测试数据

```
{  
  "data": 123,  
  "city": "nj",  
  "pop": 800  
}
```

- 加工规则

```
e_table_map(  
  tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),  
  ["city", "pop"],  
  "province",  
)
```

- 加工结果

```
data: 123  
city: nj  
pop: 800  
province: js
```

- h. 示例8：多个日志匹配字段，且日志匹配字段与映射表格字段不一样。

- 测试数据

```
{  
  "data": 123,  
  "city": "nj",  
  "pp": 800  
}
```

- 加工规则

```
e_table_map(  
  tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),  
  [("city", "city"), ("pp", "pop")],  
  "province",  
)
```

- 加工结果

```
data: 123
city: nj
pp: 800
province: js
```

- 更多参考
支持和其他函数组合使用。

e_search_dict_map

对关键字（查询字符串）以及其匹配的值的字典数据进行映射。

- 函数格式

```
e_search_dict_map(data, output_field, multi_match=false, multi_join=" ", missing=None, mode="overwrite")
```
- 参数说明

参数名称	数据类型	是否必填	说明
data	Dict	是	映射关系的字典。必须为标准的{key01:value01,key01:value02,...}格式，且关键字key必须是查询字符串。
output_field	String	是	输出字段的名称。
multi_match	Boolean	否	是否允许匹配多个字段。默认为false表示不匹配多个字段，会返回匹配到的最后一个字段值。支持使用multi_join拼接多个匹配的值。
multi_join	String	否	匹配多个字段时，多值的连接字符串，默认为空格。当multi_match值为true时生效。
missing	String	否	无匹配字段时，将该参数的取值赋给输出字段output_field。默认为None表示不做映射赋值操作。 如果字典中包含默认匹配星号（*），由于星号（*）的优先级高于missing，此时missing将不起作用。
mode	String	否	字段的覆盖模式。默认为overwrite。

- 返回结果
返回查询匹配中后的映射结果。
- 函数示例
 - 示例1：匹配模式。

- 测试数据

```
{
  "data":123,
  "pro":1
}
```


- 加工规则

e_search_dict_map ({ "pro==1": "TCP", "pro==2": "UDP", "pro==3": "HTTP"}, "protocol")
- 加工结果

data:123
pro:1
protocol:TCP
- b. 示例2：根据字段值的不同开头进行映射。

■ 测试数据

{
 "status":"200,300"
}

■ 加工规则

e_search_dict_map(
 {
 "status:2??": "ok",
 "status:3??": "redirect",
 "status:4??": "auth",
 "status:5??": "server_error",
 },
 "status_desc",
 multi_match=true,
 multi_join="测试",
)

■ 加工结果

status:200,300
status_desc:ok测试redirect
- 更多参考
- 支持和其他函数组合使用。

e_search_table_map

对某列（ 查询字符串 ） 以及其匹配的值的表格数据进行映射。

- 函数格式
- e_search_table_map(data, inpt, output_fields, multi_match=false, multi_join=" ", missing=None, mode="fill-auto")
- 参数说明

参数名称	数据类型	是否必填	说明
data	Table	是	映射关系的表格，表格某一列必须是查询字符串。
inpt	String	是	表格中用于匹配搜索的字段名。
output_fields	String, String List或Tuple List	是	表格中映射出的字段，可以是字符串、列表或者其名称映射元组的列表。

multi_match	Boolean	否	是否允许匹配多个字段。默认为false表示不匹配多个字段，会返回匹配到的第一个字段值。支持使用multi_join来拼接多个匹配的值。
multi_join	String	否	匹配多个字段时，多值的连接字符串，默认为空格。当multi_match值为true时生效。
missing	String	否	无匹配字段时，将该参数的取值赋给输出字段output_fields。默认为None表示不做映射赋值操作。 如果表格中包含默认匹配*，由于*的优先级高于missing，此时missing将不起作用。
mode	String	否	字段的覆盖模式。默认为fill-auto。

- 返回结果
返回查询匹配中后的映射结果。
- 函数示例
 - 示例1：根据映射关系的表格，将日志中city字段映射出pop和province字段。
 - 测试数据

```
{
  "data": 123,
  "city": "sh"
}
```

例如，以下映射关系的表格，其中search列是查询字符串。

search	pop	province
city==nj	800	js
city==sh	2000	sh
 - 加工规则

```
e_search_table_map(
  tab_parse_csv("search,pop,province\ncity==nj,800,js\ncity==sh,2000,sh"),
  "search",
  ["pop", "province"],
)
```
 - 加工结果

```
data: 123
city: sh
province: sh
pop: 2000
```
 - 示例2：overwrite模式。

■ 测试数据

```
{
  "data": 123,
  "city": "nj",
  "province": ""
}
```

■ 加工规则

```
e_search_table_map(
  tab_parse_csv("search,pop,province\ncity==nj,800,js\ncity==sh,2000,sh"),
  "search",
  "province",
  mode="overwrite",
)
```

■ 加工结果

```
pop: 800
data: 123
city: nj
province: js
```

c. 示例3：无匹配时目标字段的值由missing指定。

■ 测试数据

```
{
  "data": 123,
  "city": "wh",
  "province": ""
}
```

■ 加工规则

```
e_search_table_map(
  tab_parse_csv("search,pop,province\ncity==nj,800,\ncity==sh,2000,sh"),
  "search",
  "province",
  missing="Unknown",
)
```

■ 加工结果

```
data: 123
city: wh
province: Unknown
```

d. 示例4：允许匹配多个字段（multi_match模式）。

■ 测试数据

```
{
  "data": 123,
  "city": "nj,sh",
  "province": ""
}
```

■ 加工规则

```
e_search_table_map(
  tab_parse_csv("search,pop,province\ncity:nj,800,js\ncity:sh,2000,sh"),
  "search",
  "province",
  multi_match=true,
  multi_join=",",
)
```

■ 加工结果

```
data: 123
city: nj,sh
province: js,sh
```

10.2.2.2 字段值提取函数

本文介绍字段值提取函数的语法规则，包括参数解释、函数示例等。

函数列表

类型	函数	说明
正则提取	<code>e_regex</code>	根据正则表达式提取字段的值并赋值给其他字段。支持和其他函数组合使用。
JSON提取	<code>e_json</code>	对特定字段中的JSON对象进行JSON操作，包括JSON展开、JMES提取或者JMES提取后再展开。支持和其他函数组合使用。
分隔符提取	<code>e_csv</code> 、 <code>e_psv</code> 、 <code>e_tsv</code>	使用自定义的分隔符与预定义的字段名，从特定字段中提取多个字段。 <code>e_csv</code>：默认分隔符为半角逗号(,)。 <code>e_psv</code>：默认分隔符为竖线()。 <code>e_tsv</code>：默认分隔符为\t。 支持和其他函数组合使用。
KV模式提取	<code>e_kv</code>	通过quote提取多个源字段中的键值对信息。支持和其他函数组合使用。
	<code>e_kv_delimit</code>	通过分隔符提取源字段中的键值对信息。

e_regex

根据正则表达式提取字段的值并赋值给其他字段。

- 函数格式**

e_regex(key,正则表达式,fields_info,mode="fill-auto",pack_json=None)
- 参数说明**

参数名称	参数类型	是否必填	说明
key	任意	是	源字段名。如果字段不存在，则不进行任何操作。特殊字段名的设置请参见事件类型。
正则表达式	String	是	提取字段的正则表达式。支持捕获组和非捕获组正则表达式。 非捕获有时需要使用分组，且使用前缀?:。例如\w+@\w+\. \w(?:\. \cn)?。关于分组不捕获请参见非捕获组。
fields_info	String/ List/ Dict	否	匹配后目标字段名。当正则表达式参数没有配置命名捕获的名称时，必须配置该参数。

mode	String	否	字段的覆盖模式。默认为fill-auto。关于更多字段值及含义请参见字段提取检查与覆盖模式。
pack_json	String	否	将正则表达式的所有匹配结果打包放入pack_json指定的字段中。默认值是None，表示不打包。

• 返回结果

返回附带新字段值的日志。

• 函数示例

a. 示例1：提取字段中符合表达式的值。

- 测试数据

```
{
  "msg": "192.168.0.1 http://... 127.0.0.0"
}
```
- 加工规则

```
# 提取字段msg中出现的第一个IP地址。
e_regex("msg",r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}", "ip")
```
- 加工结果

```
msg: 192.168.0.1 http://... 127.0.0.0
ip: 192.168.0.1
```

b. 示例2：提取字段中符合表达式的多个值。

- 测试数据

```
{
  "msg": "192.168.0.1 http://... 127.0.0.0"
}
```
- 加工规则

```
# 提取字段msg中出现的两个IP地址，分别赋值给server_ip和client_ip。
e_regex("msg",r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}",["server_ip","client_ip"])
```
- 加工结果

```
msg: 192.168.0.1 http://... 127.0.0.0
server_ip: 192.168.0.1
client_ip: 127.0.0.0
```

c. 示例3：通过捕获组提取符合表达式的值。

- 测试数据

```
{
  "content": "start sys version: deficiency, err: 2"
}
```
- 加工规则

```
# 使用正则表达式捕获content中的version和error值。
e_regex("content",r"start sys version: (\w+),\s*err: (\d+)",["version","error"])
```
- 加工结果

```
content: start sys version: deficiency, err: 2
error: 2
version: deficiency
```

d. 示例4：通过命名捕获组提取字段值。

- 测试数据

```
{
  "content": "start sys version: deficiency, err: 2"
}
```
- 加工规则

```
e_regex("content", r"start sys version: (?P<version>\w+),\s*err: (?P<error>\d+)")
```
- 加工结果

```
content: start sys version: deficiency, err: 2
error: 2
version: deficiency
```
- e. 示例5：使用正则表达式捕获字段dict中的值，并动态命名字段名和其值。
 - 测试数据

```
{
  "dict": "verify:123"
}
```
 - 加工规则

```
e_regex("dict", r"(\w+):(\d+)", {r"k_1": r"v_2"})
```
 - 加工结果

```
dict: verify:123
k_verify: v_123
```
- f. 示例6：提取字段中符合表达式的值，并打包赋值给name字段。
 - 测试数据

```
{
  "msg": "192.168.0.1 http://... 127.0.0.0"
}
```
 - 加工规则

```
e_regex("msg", r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}", "ip", pack_json="name")
```
 - 加工结果

```
msg:192.168.0.1 http://... 127.0.0.0
name:{"ip": "192.168.0.1"}
```
- g. 示例7：使用正则表达式提取字段dict中的值，动态命名字段名和其值，并打包赋值给name字段。
 - 测试数据

```
{
  "dict": "x:123, y:456, z:789"
}
```
 - 加工规则

```
e_regex("dict", r"(\w+):(\d+)", {r"k_1": r"v_2"}, pack_json="name")
```
 - 加工结果

```
dict:x:123, y:456, z:789
name:{"k_x": "v_123", "k_y": "v_456", "k_z": "v_789"}
```
- h. 示例8：通过捕获组提取符合表达式的值，并打包赋值给name字段。
 - 测试数据

```
{
  "content": "start sys version: deficiency, err: 2"
}
```
 - 加工规则

```
e_regex( "content", r"start sys version: (\w+),\s*err: (\d+)", ["version",  
"error"],pack_json="name")
```

- 加工结果
content:start sys version: deficiency, err: 2
name:{"version": "deficiency", "error": "2"}

- 更多参考
支持和其他函数组合使用。

e_json

对特定字段中的JSON对象进行JSON操作，包括JSON展开、JMES提取或者JMES提取后再展开。

- 函数格式
e_json(key, expand=None, depth=100, prefix="__", suffix="__", fmt="simple", sep=".",
expand_array=true, fmt_array="{parent}_{index}",
include_node=r"[\u4e00-\u9fa5\u0800-\u4e00a-zA-Z][\w\-\.\~]*",
exclude_node="", include_path="", exclude_path="",
jmes="", output="", jmes_ignore_none=false, mode='fill-auto'
)

- 参数说明

参数名称	参数类型	是否必填	说明
key	String	是	源字段名。如果字段不存在，则不进行任何操作。
expand	Boolean	否	是否将字段展开。 - 没有配置jmes参数时，则默认为true，表示展开。 - 配置jmes参数时，则默认为false，表示不展开。
depth	Number	否	字段展开的深度。取值范围为1~2000，1表示只展开第一层，默认为100层。
prefix	String	否	展开时添加为字段名的前缀。
suffix	String	否	展开时添加为字段名的后缀。

fmt	String	否	格式化方式。取值： • simple（默认值）：表示将节点名作为字段名。展示形式为{prefix}{current}{suffix}。 • full：表示将父节点与当前节点合并作为字段名。展示形式为{parent_list_str}{sep}{prefix}{current}{suffix}。分隔符是由sep参数指定，默认为。 • parent：表示用完整的路径作为字段名。展示形式为{parent}{sep}{prefix}{current}{suffix}。分隔符是由sep参数指定，默认为。 • root：表示将根节点与当前节点合并作为字段名。展示形式为{parent_list[0]}{sep}{prefix}{current}{suffix}。分隔符由sep参数指定，默认为。
sep	String	否	父子节点格式化的分隔符。当fmt取值为full、parent或root时需要设置。默认为。
expand_array	Boolean	否	是否将数组展开。默认为true表示展开数组。
fmt_array	String	否	数组展开的格式化方式，格式为{parent_rlist[0]}_{index}。也可以使用最多五个占位符自定义格式化字符串：parent_list, current, sep, prefix, suffix。
include_node	String/ Number	否	节点允许名单，表示过滤时包含的节点名称。默认只有中文、数字、字母和_.-的节点才会被自动展开。
exclude_node	String	否	节点限制名单，表示过滤时排除的节点名称。
include_path	String	否	节点允许名单，表示过滤时包含的节点路径。
exclude_path	String	否	节点限制名单，表示过滤时排除的节点路径。
jmes	String	否	将字段值转化为JSON对象并通过JMES提取特定值。
output	String	否	通过JMES提取特定值时输出的字段名。
jmes_ignore_none	Boolean	否	当JMES提取不到值时是否忽略。默认为true表示忽略，否则输出一个空字符串。
mode	String	否	字段的覆盖模式。默认为fill-auto。

- JSON展开过滤

- 如果设置了节点允许名单，则内容必须包含在节点允许名单中然后才会在结果中出现。节点允许名单正则示例：e_json("json_data_filed", ..., include_node=r'key\d+')。
- 如果设置了节点限制名单，则内容必须包含在节点限制名单中然后才不会在结果中出现。节点限制名单正则示例：e_json("json_data_filed", ..., exclude_node=r'key\d+')。
- 展开节点路径：正则include_path与exclude_path从路径开头匹配，匹配路径是以.分隔。

- JMES过滤

使用JMES选择、计算。

- 选择特定JSON路径下的元素属性列表：e_json(..., jmes="cve.vendors[*].product",output="product")
- 用逗号拼接特定JSON路径下的元素属性：e_json(..., jmes="join(',', cve.vendors[*].name)",output="vendors")
- 计算特定JSON路径下元素的最大属性值：e_json(..., jmes="max(words[*].score)",output="hot_word")
- 当特定路径不存在或为空时，返回一个空字符串：e_json(..., jmes="max(words[*].score)",output="hot_word", jmes_ignore_none=false)

- parent_list和parent_rlist，以如下示例说明。

测试数据：

```
{
  "data": { "k1": 100,"k2": {"k3": 200,"k4": {"k5": 300}}}
}
```

parent_list是将父节点从左到右排列。

```
e_json("data", fmt='{parent_list[0]}-{parent_list[1]}#{current}')
```

得到的日志：

```
data:{ "k1": 100,"k2": {"k3": 200,"k4": {"k5": 300}}}
data-k2#k3:200
data-k2#k5:300
```

parent_rlist是将父节点从右到左排列。

```
e_json("data", fmt='{parent_rlist[0]}-{parent_rlist[1]}#{current}')
```

得到的日志：

```
data:{ "k1": 100,"k2": {"k3": 200,"k4": {"k5": 300}}}
k2-data#k3:200
k4-k2#k5:300
```

- 返回结果

返回附带新字段值的日志。

- 函数示例

- a. 示例1：字段展开操作。

- 测试数据

```
{
  "data": {"k1": 100, "k2": 200}
}
```

- 加工规则
`e_json("data",depth=1)`
- 加工结果
`data: {"k1": 100, "k2": 200}`
`k1: 100`
`k2: 200`
- b. 示例2：给字段名添加前缀和后缀。
 - 测试数据
`{`
`"data": {"k1": 100, "k2": 200}`
`}`
 - 加工规则
`e_json("data", prefix="data_", suffix="_end")`
 - 加工结果
`data: {"k1": 100, "k2": 200}`
`data_k1_end: 100`
`data_k2_end: 200`
- c. 示例3：将字段按照不同格式展开。
 - 测试数据
`{`
`"data": {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }`
`}`
 - fmt=full格式
`e_json("data", fmt='full')`
`data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }`
`data.k1: 100`
`data.k2.k3: 200`
`data.k2.k4.k5: 300`
 - fmt=parent格式
`e_json("data", fmt='parent')`
`data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }`
`data.k1: 100`
`k2.k3: 200`
`k4.k5: 3000`
 - fmt=root格式
`e_json("data", fmt='root')`
`data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }`
`data.k1: 100`
`data.k3: 200`
`data.k5: 300`
- d. 示例4：使用指定分隔符、字段名前缀和字段名后缀提取JSON
 - 测试数据
`{`
`"data": {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }`
`}`
 - 加工规则
`e_json("data", fmt='parent', sep="@", prefix="__", suffix="__")`
 - 加工结果
`data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }`
`data@__k1__: 100`
`k2@__k3__: 200`
`k4@__k5__: 300`

e. 示例5：指定fmt_array参数，按照数组方式提取JSON。

- 测试数据
- ```
{
 "people": [{"name": "xm", "gender": "boy"}, {"name": "xz", "gender": "boy"}, {"name": "xt", "gender": "girl"}]
}
```
- 加工规则
- ```
e_json("people", fmt='parent', fmt_array="{parent_rlist[0]}-{index}")
```
- 加工结果
- ```
people: [{"name": "xm", "gender": "boy"}, {"name": "xz", "gender": "boy"}, {"name": "xt", "gender": "girl"}]
people-0.name: xm
people-0.gender: boy
people-1.name: xz
people-1.gender: boy
people-2.name: xt
people-2.gender: girl
```

f. 示例6：使用JMES提取JSON对象。

- 测试数据
- ```
{
  "data": { "people": [{"first": "James", "last": "d"}, {"first": "Jacob", "last": "e"}], "foo": {"bar": "baz"}}
}
```
- 加工规则
- ```
e_json("data", jmes='foo', output='jmes_output0')
e_json("data", jmes='foo.bar', output='jmes_output1')
e_json("data", jmes='people[0].last', output='jmes_output2')
e_json("data", jmes='people[*].first', output='jmes_output3')
```
- 加工结果
- ```
data: { "people": [{"first": "James", "last": "d"}, {"first": "Jacob", "last": "e"}], "foo": {"bar": "baz"}}
jmes_output0: {"bar": "baz"}
jmes_output1: baz
jmes_output2: d
jmes_output3: ["James", "Jacob"]
```

- 更多参考
支持和其他函数组合使用。

e_csv、e_psv、e_tsv

使用自定义的分隔符与预定义的字段名，从特定字段中提取多个字段。

- e_csv：默认分隔符为半角逗号（,）。
 - e_psv：默认分隔符为竖线（|）。
 - e_tsv：默认分隔符为\t。
 - 函数格式
- ```
e_csv(源字段名, 目标字段列表, sep=",", quote="", restrict=true, mode="fill-auto")
e_psv(源字段名, 目标字段列表, sep="|", quote="", restrict=true, mode="fill-auto")
e_tsv(源字段名, 目标字段列表, sep="\t", quote="", restrict=true, mode="fill-auto")
```
- 参数说明

| 参数名称 | 参数类型 | 是否必填 | 说明 |
|------|------|------|----|
|------|------|------|----|

|          |         |   |                                                                                                                                            |
|----------|---------|---|--------------------------------------------------------------------------------------------------------------------------------------------|
| 源字段名     | 任意      | 是 | 源字段名。如果字段不存在，则不进行任何操作。                                                                                                                     |
| 目标字段列表   | 任意      | 是 | 字段值经过分隔符分隔后的每个值对应的字段名。可以是字符串的列表，例如：["error", "message", "result"]。<br>当字段名中不包含逗号时，也可以直接用逗号作为分隔字符，例如："error, message, result"。              |
| sep      | String  | 否 | 分隔符，只能是单个字符。                                                                                                                               |
| quote    | String  | 否 | 引用符，用于包裹值的字符。当值包含分隔符时需要使用。                                                                                                                 |
| restrict | Boolean | 否 | 是否采用严格模式，默认为false表示非严格模式。当分隔值的个数与目标字段列表数不一致时： <ul style="list-style-type: none"><li>严格模式下不进行任何操作。</li><li>非严格模式下对前几个可以配对的字段进行赋值。</li></ul> |
| mode     | String  | 否 | 字段的覆盖模式。默认为fill-auto。                                                                                                                      |

- **返回结果**

返回附带新字段值的日志。

- **函数示例**

以e\_csv为例，e\_psv和e\_tsv功能类似。

- **测试数据**

```
{
 "content": "192.168.0.100,10/Jun/2019:11:32:16 +0800,example.aadoc.com,GET /zf/11874.html
HTTP/1.1,200,0.077,6404,192.168.0.100:8001,200,0.060,https://image.developer.aadoc.com/s?q=
%E8%9B%8B%E8%8A%B1%E9%BE%99%E9%A1%BB%E9%9D%A2%E7%9A%84%E5%81%9A
%E6%B3%95&from=wy878378&uc_param_str=dnntnwepffrgibjbprrsvdsei,-,Mozilla/5.0 (Linux;
Android 9; HWI-AL00 Build/HUAWEIHWI-AL00) AppleWebKit/537.36,-,-"
}
```

- **加工规则**

```
e_csv("content", "remote_addr,
time_local,host,request,status,request_time,body_bytes_sent,upstream_addr,upstream_status,
upstream_response_time,http_referer,http_x_forwarded_for,http_user_agent,session_id,guid")
```

- **加工结果**

```
content: 192.168.0.100,10/Jun/2019:11:32:16 +0800,example.aadoc.com,GET /zf/11874.html
HTTP/1.1,200,0.077,6404,192.168.0.100:8001,200,0.060,https://image.developer.aadoc.com/s?q=
%E8%9B%8B%E8%8A%B1%E9%BE%99%E9%A1%BB%E9%9D%A2%E7%9A%84%E5%81%9A
%E6%B3%95&from=wy878378&uc_param_str=dnntnwepffrgibjbprrsvdsei,-,Mozilla/5.0 (Linux;
Android 9; HWI-AL00 Build/HUAWEIHWI-AL00) AppleWebKit/537.36 (KHTML, like Gecko)
Version/4.0 Mobile Safari/537.36,-,-
body_bytes_sent: 6404
guid: -
host: example.aadoc.com
http_referer: https://image.developer.aadoc.com/s?q=%E8%9B%8B%E8%8A%B1%E9%BE
%99%E9%A1%BB%E9%9D%A2%E7%9A%84%E5%81%9A
%E6%B3%95&from=wy878378&uc_param_str=dnntnwepffrgibjbprrsvdsei
http_user_agent: Mozilla/5.0 (Linux; Android 9; HWI-AL00 Build/HUAWEIHWI-AL00)
AppleWebKit/537.36
http_x_forwarded_for: -
remote_addr: 192.168.0.100
```

```
request: GET /zf/11874.html HTTP/1.1
request_time: 0.077
session_id: -
status: 200
time_local: 10/Jun/2019:11:32:16 +0800
topic: syslog-forwarder
upstream_addr: 192.168.0.100:800
1upstream_response_time: 0.060
upstream_status: 200
```

- **更多参考**  
支持和其他函数组合使用。

## e\_kv

通过quote提取多个源字段中的键值对信息。

- **函数格式**  
e\_kv(源字段或源字段列表, sep="=", quote="", escape=false, prefix="", suffix="", mode="fill-auto")
- **参数说明**

| 参数名称          | 参数类型          | 是否必填 | 说明                                                                                                                                                                                        |
|---------------|---------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 源字段或源<br>字段列表 | 字符串或<br>字符串列表 | 是    | 字段名或多个字段名的列表。                                                                                                                                                                             |
| sep           | String        | 否    | 关键字与值的正则表达式的分隔符串，<br>默认为=，不限于单个字符。<br><b>说明</b> 可以使用非捕获分组，但不能使用<br>捕获分组。                                                                                                                  |
| quote         | String        | 否    | 引用符，用于包裹值的字符。默认为"<br>。<br><b>说明</b> 提取的动态键值对的值一般需要<br>quote来包括，例如：a="abc"<br>，b="xyz"如果提取对象中不包含，则只<br>提取如下字符集的值：中文字母数字<br>_-.%~。例如a=中文ab12_-.%~ abc<br>b=123可以提取a: 中文ab12_-.%~，b:<br>123。 |
| escape        | Boolean       | 否    | 是否自动提取反转字符的值。默认为<br>false表示否。例如key="abc\"xyz"默认<br>提取字段key的值为abc\，设置<br>escape=true时，提取的值为abc"xyz。                                                                                        |
| prefix        | String        | 否    | 给提取的字段名添加前缀。                                                                                                                                                                              |
| suffix        | String        | 否    | 给提取的字段名添加后缀。                                                                                                                                                                              |
| mode          | String        | 否    | 字段的覆盖模式。默认为fill-auto。                                                                                                                                                                     |

- **返回结果**  
返回附带新字段值的日志。
- **函数示例**

## a. 示例1：使用默认分隔符=提取键值对信息。

## ■ 测试数据

```
{
 "http_refer": "https://video.developer.aadoc.com/s?q=asd&a=1&b=2"
}
```

如果测试数据为request\_uri: a1=1&a2=&a3=3, a2值为空, 则使用e\_kv()函数无法提取出a2。您可以使用e\_regex()函数进行提取, 例如e\_regex("request\_uri", r'(\w+)=(\[^\&]\*\)', {r"\1":r"\2"}, mode="overwrite")。

## ■ 加工规则

```
e_kv("http_refer")
```

## ■ 加工结果

```
http_refer: https://video.developer.aadoc.com/s?q=asd&a=1&b=2
q: asd
a: 1
b: 2
```

## b. 示例2：给字段名增加前缀和后缀。

## ■ 测试数据

```
{
 "http_refer": "https://video.developer.aadoc.com/s?q=asd&a=1&b=2"
}
```

## ■ 加工规则

```
e_kv(
 "http_refer",
 sep="=",
 quote="",
 escape=false,
 prefix="data_",
 suffix="_end",
 mode="fill-auto",
)
```

## ■ 加工结果

```
http_refer: https://video.developer.aadoc.com/s?q=asd&a=1&b=2
data_q_end: asd
data_a_end: 1
data_b_end: 2
```

## c. 示例3：提取字段content2中的键值对信息，使用escape参数提取反转字符的值。

## ■ 测试数据

```
{
 "content2": "k1:\\v1\\\"abc\\", k2:\\v2\\", k3: \\\"v3\\\""
}
```

## ■ 加工规则

```
e_kv("content2", sep=":", escape=true)
```

## ■ 加工结果

```
content2: k1:"v1\"abc", k2:"v2", k3: "v3"
k1: v1"abc
k2: v2
k3: v3
```

## ● 更多参考

支持和其他函数组合使用。

## e\_kv\_delimit

通过分隔符提取源字段中的键值对信息。

- 函数格式

`e_kv_delimit(源字段或源字段列表, pair_sep=r"\s", kv_sep="=", prefix="", suffix="", mode="fill-auto")`

- 参数说明

| 参数名称      | 参数类型      | 是否必填 | 说明                                                                                                                                         |
|-----------|-----------|------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 源字段或源字段列表 | 字符串或字符串列表 | 是    | 字段名或多个字段名的列表。                                                                                                                              |
| pair_sep  | String    | 否    | 用于分隔键值对的正则字符集，默认为\s。例如\s\w、abc\s等。<br><b>说明</b> 如果您需要使用字符串对字段进行分隔，推荐您使用str_replace或regex_replace将字符串转换成字符作为分隔符，然后再使用e_kv_delimit函数对字段进行分隔。 |
| kv_sep    | String    | 否    | 用于分隔键值对的正则字符串，默认为=，不限于单个字符。<br>可以使用非捕获分组，但不能使用捕获分组。                                                                                        |
| prefix    | String    | 否    | 给提取的字段名添加前缀。                                                                                                                               |
| suffix    | String    | 否    | 给提取的字段名添加后缀。                                                                                                                               |
| mode      | String    | 否    | 字段的覆盖模式。默认为fill-auto。                                                                                                                      |

- 返回结果

返回附带新字段值的日志。

- 函数示例

a. 示例1：使用默认分隔符=提取键值对信息。

- 测试数据

```
{
 "data": "i=c1 k1=v1 k2=v2 k3=v3"
}
```

- 说明**

如果测试数据为request\_uri: a1=1&a2=&a3=3，a2值为空，则使用e\_kv\_delimit()函数无法提取出a2。您可以使用e\_regex()函数进行提取，例如e\_regex("request\_uri",r'(\w+)=(\^[^&]\*)',{r"\1":r"\2"}，mode="overwrite")。

- 加工规则

```
e_kv_delimit("data")
```

- 加工结果

```
data: i=c1 k1=v1 k2=v2 k3=v3
i: c1
k2: v2
```

```
k1: v1
k3: v3
```

b. 示例2：使用分隔符&?提取键值对信息。

■ 测试数据

```
{
 "data": "k1=v1&k2=v2?k3=v3"
}
```

■ 加工规则

```
e_kv_delimit("data",pair_sep=r"&?")
```

■ 加工结果

```
data: k1=v1&k2=v2?k3=v3
k2: v2
k1: v1
k3: v3
```

c. 示例3：使用正则表达式提取键值对信息。

■ 测试数据

```
{
 "data": "k1=v1 k2:v2 k3=v3"
}
```

■ 加工规则

```
e_kv_delimit("data", kv_sep=r"(?:=|:)")
```

■ 加工结果

```
data: k1=v1 k2:v2 k3=v3
k2: v2
k1: v1
k3: v3
```

### 10.2.2.3 字段操作函数

本文介绍字段操作函数的语法规则，包括参数解释、函数示例等。

#### 函数列表

| 函数                            | 说明                                                  |
|-------------------------------|-----------------------------------------------------|
| <a href="#">V</a>             | 获得日志特定字段的值。当同时传入多个字段名时，返回日志中第一个存在的字段的值。支持和其他函数组合使用。 |
| <a href="#">e_set</a>         | 添加新字段或为现有字段设置新的字段值。支持和其他函数组合使用。                     |
| <a href="#">e_drop_fields</a> | 删除符合条件的日志字段。支持和其他函数组合使用。                            |
| <a href="#">e_keep_fields</a> | 保留符合条件的日志字段。                                        |
| <a href="#">e_pack_fields</a> | 打包日志字段，并输出到新的字段中。                                   |
| <a href="#">e_rename</a>      | 重命名符合条件的日志字段名称。支持和其他函数组合使用。                         |



## V

调用v函数获得日志特定字段的值。当同时传入多个字段名时，返回日志中第一个存在的字段的值。

- 函数格式

```
v(key, ..., default=None)
```

- 参数说明

| 参数      | 参数类型   | 是否必填 | 说明                               |
|---------|--------|------|----------------------------------|
| key     | String | 是    | 指定字段名。                           |
| default | 任意     | 否    | 指定的字段名不存在时，返回default的值。默认值为None。 |

- 返回结果

返回日志中第一个存在的字段值。不存在时返回default参数的值。

- 函数示例

将content字段的值赋给test\_content字段。

```
- 测试数据
{
 "content": "hello"
}
```

```
- 加工规则
e_set("test_content", v("content"))
```

```
- 加工结果
content: hello
test_content: hello
```

- 更多参考

支持和其他函数组合使用。

## e\_set

调用e\_set函数添加新字段或为现有字段设置新的字段值。

- 函数格式

```
e_set(key1, value1, key2, value2, mode="overwrite")
```

- 函数中key1和value1必须成对出现。
- 通过e\_set函数设置时间字段F\_TIME或\_\_time\_\_时，必须设置为数字、字符串。

```
e_set(F_TIME, "abc") # 错误
e_set(F_TIME, "12345678") # 正确
```

- 参数说明

| 参数  | 参数类型   | 是否必填 | 说明                       |
|-----|--------|------|--------------------------|
| key | String | 是    | 目标字段名，也可以通过字符串表达式获得该字段名。 |

|       |        |   |                                                                                      |
|-------|--------|---|--------------------------------------------------------------------------------------|
| value | 任意     | 是 | 新的字段值。非字符串都转化成字符串放入日志中，其中元组、列表、字典会转换成JSON对象的字符串。<br><b>说明</b> 如果传递的值是None，则不会进行更新操作。 |
| mode  | String | 否 | 字段的覆盖模式。默认为overwrite。                                                                |

- **返回结果**  
返回更新后的日志。
- **函数示例**
  - a. 示例1：为字段设置固定值。  
添加一个新字段city，字段值为上海。  

```
e_set("city", "上海")
```
  - b. 示例2：复制字段值。  
调用单个表达式函数，将现有字段ret的值，赋给新字段result。
    - 原始数据  

```
{"ret": "value"}
```
    - 加工规则  

```
e_set("result", v("ret"))
```
    - 加工结果  

```
ret: value
result: value
```
  - c. 示例3：动态设置值。  
调用组合表达式函数，获取第一个存在的字段值，返回其小写格式并赋值给字段result。  

```
e_set("result", str_lower(v("ret", "return")))
```
  - d. 示例4：多次设置字段值。
    - 原始数据  

```
{
 "ret": "fail"
}
```
    - 加工规则  

```
e_set("event_type", "login event", "event_info", "login host")
```
    - 加工结果  

```
ret: fail
event_type: login event
event_info: login host
```
- **更多参考**  
支持和其他函数组合使用。

e\_drop\_fields

调用e\_drop\_fields函数删除符合条件的日志字段。

- **函数格式**

```
e_drop_fields(key1, key2, ..., regex=false)
```

- 参数说明

| 参数    | 参数类型    | 是否必填 | 说明                                                                             |
|-------|---------|------|--------------------------------------------------------------------------------|
| key   | String  | 是    | 日志字段名，可以为正则表达式。当字段名完全满足条件时删除该字段，保留不满足条件的字段。关于正则表达式的更多信息，请参见正则表达式。至少需要配置一个日志字段。 |
| regex | Boolean | 否    | 如果设置为false，表示不使用正则表达式进行匹配。当不配置该参数时，系统默认取值为true。                                |

- 返回结果

返回删除后的日志。

- 函数示例

如果content字段的值为123，则删除content字段和age字段。

- 测试数据

```
{
 "age": 18,
 "content": 123,
 "name": "twiss"
}
```

- 加工规则

```
e_if(e_search("content==123"), e_drop_fields("content", "age", regex=true))
```

- 加工结果

```
name: twiss
```

- 更多参考

支持其他函数组合使用。

## e\_keep\_fields

调用e\_keep\_fields函数保留符合条件的日志字段。

云日志服务中包含内置的元字段，例如\_\_time\_\_、\_\_topic\_\_等。如果在调用e\_keep\_fields函数时没有保留\_\_time\_\_字段，则日志时间将被重置为系统当前时间。如果您不希望重置元字段的值，需要将元字段放入列表中，常见格式为F\_TIME, F\_META, F\_TAGS, "f1", "f2"。

- 函数格式

```
e_keep_fields(key1, key2, ..., regex=false)
```

- 参数说明

| 参数  | 参数类型   | 是否必填 | 说明                                                     |
|-----|--------|------|--------------------------------------------------------|
| key | String | 是    | 日志字段名，可以为正则表达式。当字段名完全满足条件时保留该字段，删除不满足条件的字段。至少需要配置一个字段。 |

|       |         |   |                                                 |
|-------|---------|---|-------------------------------------------------|
| regex | Boolean | 否 | 如果设置为false，表示不使用正则表达式进行匹配。当不配置该参数时，系统默认取值为true。 |
|-------|---------|---|-------------------------------------------------|

- **返回结果**  
返回保留的日志。
- **函数示例**  
如果content字段的值是123，则保留content和age字段。
  - 测试数据

```
{
 "age": 18,
 "content": 123,
 "name": "twiss"
}
```
  - 加工规则

```
e_if(e_search("content==123"), e_keep_fields("content", "age"))
```
  - 加工结果

```
age: 18
content: 123
```

e\_pack\_fields

用e\_pack\_fields函数将日志字段进行打包，并输出到新的字段中。

- **函数格式**  
e\_pack\_fields(output\_fields,include=".\*",exclude=None,drop\_packed=true)
- **参数说明**

| 参数           | 类型      | 是否必填 | 说明                                                                                                                                     |
|--------------|---------|------|----------------------------------------------------------------------------------------------------------------------------------------|
| output_field | String  | 是    | 指定打包后输出的字段名称。其值为JSON格式。                                                                                                                |
| include      | String  | 否    | 白名单配置，符合正则表达式的字段会被打包。默认为".*"，表示全部匹配。                                                                                                   |
| exclude      | String  | 否    | 黑名单配置，符合正则表达式的字段不会被打包。默认为None，表示不进行匹配判断。                                                                                               |
| drop_packed  | Boolean | 否    | 打包数据后是否删除被打包的原数据，默认为true。 <ul style="list-style-type: none"><li>● true（默认值）：输出结果中删除被打包的原数据。</li><li>● false：输出结果中不删除被打包的原数据。</li></ul> |

- **返回结果**  
返回被打包后的日志数据。
- **函数示例**

- a. 示例1：将日志所有字段打包到test字段，默认删除被打包的原始字段。

■ 测试数据

```
{
 "test1":123,
 "test2":456,
 "test3":789
}
```

■ 加工规则

```
e_pack_fields("test")
```

■ 加工结果

```
test:{"test1": 123, "test2": 456, "test3": 789}
```

- b. 示例2：将日志所有字段打包到test字段，不删除被打包的原始字段。

■ 测试数据

```
{
 "test1":123,
 "test2":456,
 "test3":789
}
```

■ 加工规则

```
e_pack_fields("test",drop_packed=false)
```

■ 加工结果

```
test:{"test1": 123, "test2": 456, "test3": 789}
test1:123
test2:456
test3:789
```

- c. 示例3：打包test和abcd字段到content字段，不删除被打包的原始字段。

■ 测试数据

```
{
 "abcd@#%":123,
 "test":456,
 "abcd":789
}
```

■ 加工规则

```
e_pack_fields("content", include="\w+", drop_packed=false)
```

■ 加工结果

```
abcd:789
abcd@#%:123
content:{"test": 456, "abcd": 789}
test:456
```

- d. 示例4：不打包test和abcd字段，其余字段打包到content字段，删除被打包的原始字段。

■ 测试数据

```
{
 "abcd@#%":123,
 "test":456,
 "abcd":789
}
```

■ 加工规则

```
e_pack_fields("content", exclude="\w+", drop_packed=true)
```

■ 加工结果

```
abcd:789
content:{"abcd@#%": "123"}
test:456
```

e\_rename

调用e\_rename函数重命名符合条件的日志字段名称。

- 函数格式**  
e\_rename("key1", "new key1", "key2", "new key2", ..., regex=false)  
函数中key和new key必须成对出现。

- 参数说明**

| 参数      | 参数类型    | 是否必填 | 说明                                              |
|---------|---------|------|-------------------------------------------------|
| key     | String  | 是    | 日志字段名，可以为正则表达式。当字段名完全满足条件时，重命名该字段。至少需要配置一个字段。   |
| new key | String  | 是    | 重命名后的字段名。                                       |
| regex   | Boolean | 否    | 如果设置为false，表示不使用正则表达式进行匹配。当不配置该参数时，系统默认取值为true。 |

- 返回结果**  
返回重命名后的字段。
- 函数示例**

a. 示例1：将字段host重命名为client\_hos。

测试数据

```
{
 "host": 1006
}
```

加工规则

```
e_rename("host","client_host")
```

加工结果

```
client_host: 1006
```

b. 示例2：不存在字段时，不进行重命名。

测试数据

```
{
 "host": 1006
}
```

加工规则

```
e_rename("url","rename_url")
```

加工结果

```
host: 1006
```

10.2.2.4 事件操作函数

本文介绍事件操作函数的语法规则，包括参数解释、函数示例等。

函数列表

| 类型   | 函数                             | 说明                                                                                                                                                                                                                                                                                                                                                    |
|------|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 事件操作 | <b>e_drop</b>                  | 根据条件判断是否丢弃日志。支持和其他函数组合使用。                                                                                                                                                                                                                                                                                                                             |
|      | <b>e_keep</b>                  | 根据条件判断是否保留日志。<br>e_keep函数和e_drop函数都会丢弃日志。<br>e_keep函数在不满足条件时丢弃，而e_drop函数则是在满足条件时丢弃。<br># 以下4个加工规则等价<br>e_if_else(e_search("f1==v1"), e_keep(), e_drop())<br>e_if_else(e_search("not f1==v1"), e_drop())<br>e_keep(e_search("f1==v1"))<br>e_drop(e_search("not f1==v1"))<br># 以下加工规则无意义<br>e_if(e_search("..."), e_keep())<br>e_keep()<br>支持和其他函数组合使用。 |
| 事件分裂 | <b>e_split</b>                 | 基于日志字段的值分裂出多条日志，并且支持通过JMES提取字段后再进行分裂。支持和其他函数组合使用。                                                                                                                                                                                                                                                                                                     |
| 输出事件 | <b>e_output、<br/>e_coutput</b> | 输出日志到指定的日志流别名中（日志流别名在创建加工任务时配置目标日志流时指定），并可配置输出时的tag信息。 <ul style="list-style-type: none"><li>e_output：执行到e_output函数时，输出日志到指定的日志流中，且对应的日志不再执行后面的加工规则。</li><li>e_coutput：执行到e_coutput函数时，输出日志到指定的日志流中，且对应的日志继续执行后面的加工规则。</li></ul> 支持和其他函数组合使用。                                                                                                         |

e\_drop

根据条件判断是否丢弃日志。

- 函数格式**  
e\_drop(condition=true)  
支持固定标识DROP，等价于e\_drop()。
- 参数说明**

| 参数名称 | 参数类型 | 是否必填 | 说明 |
|------|------|------|----|
|------|------|------|----|

|           |      |   |                          |
|-----------|------|---|--------------------------|
| condition | Bool | 否 | 默认为true，一般传递一个条件判断函数的结果。 |
|-----------|------|---|--------------------------|

- **返回结果**

满足条件则丢弃日志并返回None，否则返回原日志。

- **函数示例**

a. 示例1：当\_\_programe\_\_字段的值为access时丢弃日志，否则保留该日志。

- **测试数据**

```
[
 {
 "__programe__": "access",
 "age": 18,
 "content": 123,
 "name": "maki"
 },
 {
 "__programe__": "error",
 "age": 18,
 "content": 123,
 "name": "maki"
 }
]
```

- **加工规则**

```
e_if(e_search("__programe__==access"), DROP)
```

- **加工结果**

丢弃\_\_programe\_\_字段值为access的日志，保留\_\_programe\_\_字段的值为error的日志。

```
__programe__: error
age: 18
content: 123
name: maki
```

b. 示例2：条件判断结果为true，丢弃日志。

- **测试数据**

```
{
 "k1": "v1",
 "k2": "v2",
 "k3": "k1"
}
```

- **加工规则**

```
e_drop(e_search("k1==v1"))
```

- **加工结果**

因为k1==v1条件为true，因此丢弃该日志。

c. 示例3：条件判断结果为false，保留日志。

- **测试数据**

```
{
 "k1": "v1",
 "k2": "v2",
 "k3": "k1"
}
```



- 加工规则  
`e_drop(e_search("not k1==v1"))`

- 加工结果  
k1: v1  
k2: v2  
k3: k1

d. 示例4：不设置判断条件时，使用默认值true，丢弃日志。

- 测试数据  
{  
  "k1": "v1",  
  "k2": "v2",  
  "k3": "k1"  
}

- 加工规则  
`e_drop()`

- 加工结果  
丢弃日志。

- **更多参考**

支持和其他函数组合使用。

## e\_keep

据条件判断是否保留日志。

- **函数格式**

`e_keep(condition=true)`

支持固定标识KEEP，等价于e\_keep()。

- **参数说明**

| 参数名称      | 参数类型 | 是否必填 | 说明                       |
|-----------|------|------|--------------------------|
| condition | Bool | 否    | 默认为true，一般传递一个条件判断函数的结果。 |

- **返回结果**

满足条件则返回原日志，不满足时丢弃日志。

- **函数示例**

a. 示例1：当\_\_programe\_\_字段的值是access的时候保留日志，否则丢弃日志。

- 测试数据  
[  
  {  
     "\_\_programe\_\_": "access",  
     "age": 18,  
     "content": 123,  
     "name": "maki"  
  },  
  {  
     "\_\_programe\_\_": "error",  
     "age": 18,  
     "content": 123,  
  }

```
"name": "maki"
}
]
```

- 加工规则

```
e_keep(e_search("__programe__==access"))
#等价于
e_if(e_search("not __programe__==access"), e_drop())
#等价于
e_if_else(e_search("__programe__==access"), e_keep(), e_drop())
```

- 加工结果

保留\_\_programe\_\_字段值为access的日志。

```
__programe__: access
age: 18
content: 123
name: maki
```

b. 示例2：条件判断结果为true，保留日志。

- 测试数据

```
{
 "k1": "v1",
 "k2": "v2",
 "k3": "k1"
}
```

- 加工规则

```
e_keep(e_search("k1==v1"))
```

- 加工结果

```
k1: v1
k2: v2
k3: k1
```

c. 示例3：条件判断结果为false，丢弃日志。

- 测试数据

```
{
 "k1": "v1",
 "k2": "v2",
 "k3": "k1"
}
```

- 加工规则

```
e_keep(e_search("not k1==v1"))
```

- 加工结果：丢弃日志。

d. 示例4：判断条件为false。

- 测试数据

```
{
 "k1": "v1",
 "k2": "v2",
 "k3": "k1"
}
```

- 加工规则

```
e_keep(false)
```

- 加工结果：丢弃日志。

- 更多参考

支持和其他函数组合使用。

## e\_split

基于日志字段的值分裂出多条日志，并且支持通过JMES提取字段后再进行分裂。

- **函数格式**

```
e_split(字段名, sep=',', quote='', lstrip=true, jmes=None, output=None)
```

分裂规则：

- a. 如果配置了jmes参数，则将日志字段的值转化为JSON列表，并使用JMES提取值作为下一步的值。如果没有配置jmes参数，则将字段的值直接作为下一步的值。
- b. 如果上一步的值是一个列表或JSON列表格式的字符串，则按照此列表分裂并结束处理。否则使用sep、quote或lstrip将上一步的值进行CSV解析，根据解析后的多个值进行分裂并结束处理。

- **参数说明**

| 参数名称   | 参数类型   | 是否必填 | 说明                                  |
|--------|--------|------|-------------------------------------|
| 字段名    | String | 是    | 需要分裂的字段名。                           |
| sep    | String | 否    | 用于分隔多个值的分隔符。                        |
| quote  | String | 否    | 用于引用多个值的配对类字符的引用符。                  |
| lstrip | String | 否    | 是否将值左边的空格去掉，默认为true。                |
| jmes   | String | 否    | 将字段值转化为JSON对象，并使用JMES提取特定值，再进行分裂操作。 |
| output | String | 否    | 设置一个新的字段名，默认覆盖旧字段名。                 |

- **返回结果**

返回日志列表，列表中字段的值都是源列表中的值。

- **函数示例**

- **测试数据**

```
{
 "__topic__": "",
 "age": 18,
 "content": 123,
 "name": "maki"
}
```

- **加工规则**

```
e_set("__topic__", "V_SENT,V_RECV,A_SENT,A_RECV")
e_split("__topic__")
```

- **加工结果**

```
[
 {
 "__topic__": "V_SENT",
 "name": "maki",
 "age": 18,
 "content": 123
 },
 {
 "__topic__": "V_RECV",
 "name": "maki",
 "age": 18,
 "content": 123
 }
]
```

```
},
{
 "__topic__": "A_SENT",
 "name": "maki",
 "age": 18,
 "content": 123
},
{
 "__topic__": "A_RECV",
 "name": "maki",
 "age": 18,
 "content": 123
}
]
```

- **更多参考**  
支持和其他函数组合使用。

## e\_output、e\_coutput

输出日志到指定的日志流中，并可配置输出时tag信息。

- **函数格式**  
e\_output(logstream, tags=None)  
e\_coutput(logstream, tags=None)  
预览时不会输出日志到目标日志流中，而是输出到页面，供您调试。
- **参数说明**

| 参数名称      | 参数类型   | 是否必填 | 说明                                 |
|-----------|--------|------|------------------------------------|
| logstream | String | 否    | 需要输出日志流别名，日志流别名在创建加工任务时配置目标日志流时指定。 |
| tags      | Dict   | 否    | 为日志设置新的标签，以字典格式传入。                 |

### 加工结果

- e\_output：输出日志到指定的日志流中，且对应的日志不再执行后面的加工规则。
- e\_coutput：输出日志到指定的日志流中，且对应的日志继续执行后面的加工规则。
- **函数示例**
  - 示例1：将k2满足正则表达式，输出到target2中。

- **测试数据**

```
{
 "k1": "v1",
 "k2": "v2",
 "x1": "v3",
 "x5": "v4"
}
```

- **加工规则**

此处e\_drop()函数的作用是把e\_if()函数过滤掉的数据做删除处理。如果不添加该函数，则被过滤的数据被投递到默认的存储目标中。创建任务时配置的第一条目标日志流为默认目标。

```
e_if(e_match("k2", r"\w+"), e_output("target2"))
e_drop()
```

■ 加工结果

```
{
 "k1": "v1",
 "k2": "v2",
 "x1": "v3",
 "x5": "v4"
}
```

在结果预览处可以看到目标日志流别名变成了target2。

b. 示例2：将k2满足正则表达式，输出到target2中，并设置tag。

■ 测试数据

```
{
 "k1": "v1",
 "k2": "v2",
 "x1": "v3",
 "x5": "v4"
}
```

■ 加工规则

```
e_if(e_match("k2", r"\w+"), e_output("target2", tags={"topic": "topic1"}))
e_drop()
```

■ 加工结果

```
{
 "k1": "v1",
 "__tag__": {
 "topic": "topic1"
 },
 "k2": "v2",
 "x1": "v3",
 "x5": "v4"
}
```

在结果预览处可以看到目标日志流别名变成了target2。

● 更多参考

支持和其他函数组合使用。

10.2.2.5 流程控制函数

本文主要介绍流程控制函数的语法规则，包括参数解释、函数示例等。

函数列表

| 函数                        | 说明                                                                                                                                                                                       |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">e_compose</a> | 用于组合一系列操作。 <ul style="list-style-type: none"><li>• 常用于e_if、e_switch、e_if_else中组合操作。</li><li>• 依次调用操作，将日志传递转换并返回最后的日志。</li><li>• 对于某一条日志，如果其中某一操作删除了日志，则不会再执行后续操作。</li></ul> 支持和其他函数组合使用。 |

| 函数               | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>e_if</b>      | <p>条件与操作组合。</p> <ul style="list-style-type: none"><li>满足条件则进行对应操作，不满足条件则不进行对应操作，直接进行下一个条件判断。</li><li>对于某一条日志，如果其中某一操作删除了日志，则不会再执行后续操作。</li></ul> <pre>e_if(<br/>  e_has("a"), e_output("target-a"),<br/>  e_has("b"), e_output("target-b"),<br/>)</pre> <p>例如，该加工规则相当于以下Python代码结构：</p> <pre>if e_has("a"):<br/>  e_output("target-a")<br/>if e_has("b"):<br/>  e_output("target-b")</pre> <p>支持和其他函数组合使用。</p>                                                                                                                                               |
| <b>e_if_else</b> | <p>根据条件判断的结果进行对应操作。</p> <pre>e_if_else(e_has("a"), e_output("target-a"), e_output("target-b"))</pre> <p>例如，该加工规则相当于以下Python代码结构：</p> <pre>if e_has("a"):<br/>  e_output("target-a")<br/>else:<br/>  e_output("target-b")</pre>                                                                                                                                                                                                                                                                                                                               |
| <b>e_switch</b>  | <p>条件与操作的组合。</p> <ul style="list-style-type: none"><li>满足条件则进行对应操作并返回结果，不满足条件则不进行对应操作，直接进行下一个条件判断。</li><li>如果没有满足任何条件，但配置了默认参数，则执行默认配置的操作并返回结果。</li><li>对于某一条日志，如果其中某一操作删除了日志，则不会再执行后续操作。</li></ul> <pre>e_switch(<br/>  e_has("a"), e_output("target-a"),<br/>  e_has("b"), e_output("target-b"),<br/>  default=e_output("target-default"),<br/>)</pre> <p>例如，该加工规则相当于以下Python代码结构：</p> <pre>if e_has("a"):<br/>  e_output("target-a")<br/>elif e_has("b"):<br/>  e_output("target-b")<br/>else:<br/>  e_output("target-default")</pre> <p>支持和其他函数组合使用。</p> |

**e\_compose**

- 组合多个操作。
- 函数格式**  
e\_compose(操作1, 操作2, ……)

- 参数说明

| 参数名称 | 参数类型   | 是否必填 | 说明          |
|------|--------|------|-------------|
| 操作1  | 全局操作函数 | 是    | 全局操作函数或其组合。 |
| 操作2  | 全局操作函数 | 否    | 全局操作函数或其组合。 |

- 返回结果

返回操作后日志。

- 函数示例

如果content字段的值为123，则先删除age字段和name字段，然后将content字段的值设置为ctx。

- 测试数据

```
{
 "content": 123,
 "age": 23,
 "name": "twiss"
}
```

- 加工规则

```
e_if(
 e_search("content==123"),
 e_compose(e_drop_fields("age|name"),
 e_rename("content", "ctx")),
)
```

- 加工结果

ctx: 123

- 更多参考

支持和其他函数组合使用。

## e\_if

据判断条件执行操作。

- 函数格式

```
e_if(条件, 操作)
e_if(条件1, 操作1, 条件2, 操作2, ……)
```

函数中条件和操作必须成对出现。

- 参数说明

| 参数名称 | 参数类型   | 是否必填 | 说明                         |
|------|--------|------|----------------------------|
| 条件   | 任意     | 是    | 表达式或其组合。其结果不是布尔值时，会进行真假判断。 |
| 操作   | 全局操作函数 | 否    | 全局操作函数或其组合。                |

- 返回结果

返回加工处理后的日志。

- 函数示例

## a. 示例1：字段值匹配后再进行操作。

result字段值为failed或failure时，设置\_\_topic\_\_字段的值为login\_failed\_event。

## ■ 测试数据

```
{
 "result": "failed"
}
```

## ■ 加工规则

```
e_if(e_match("result", r"failed|failure"), e_set("__topic__", "login_failed_event"))
```

## ■ 加工结果

```
result: failed
__topic__: login_failed_event
```

## b. 示例2：根据字段值判断后再提取数据。

当request\_body字段存在且值非空时，调用字段类操作函数JSON将request\_body字段展开成多个值。

## ■ 测试数据

```
{
 "request_body": {"k1": 100, "k2": 200}
}
```

## ■ 加工规则

```
e_if(v("request_body"), e_json("request_body"))
```

## ■ 加工结果

```
request_body: {"k1": 100, "k2": 200}
k1: 100
k2: 200
```

## c. 示例3：高级判断后再进行操作。

当valid字段的值为小写failed时，丢弃日志。

## ■ 测试数据

```
{
 "valid": "failed"
}
```

## ■ 加工规则

```
e_if(op_eq(str_lower(v("valid")), "failed"), e_drop())
```

## ■ 加工结果：丢弃日志

## d. 示例4：多个条件按顺序操作。

## ■ 测试数据

```
{
 "valid": "failed"
}
```

## ■ 加工规则

```
e_if(True, e_set("__topic__", "default_login"), e_match("valid", "failed"), e_set("__topic__", "login_failed_event"))
```

## ■ 加工结果

```
valid: failed
__topic__: login_failed_event
```

## ● 更多参考

支持和其他函数组合使用。



## e\_if\_else

根据判断条件的结果执行操作。

- **函数格式**

`e_if_else(条件, 真时操作, 假时操作)`

- **参数说明**

| 参数名称 | 参数类型   | 是否必填 | 说明                         |
|------|--------|------|----------------------------|
| 条件   | 任意     | 是    | 表达式或其组合。其结果不是布尔值时，会进行真假判断。 |
| 真时操作 | 全局操作函数 | 是    | 全局操作函数或其组合。                |
| 假时操作 | 全局操作函数 | 是    | 全局操作函数或其组合。                |

- **返回结果**

返回不同条件对应的操作结果。

- **函数示例**

如果result字段的值为ok或pass，或者status字段的值为200，则保留日志。

- **测试数据**

```
{
 "result": "ok",
 "status": 400
}
{
 "result": "Pass",
 "status": 200
}
{
 "result": "failure",
 "status": 500
}
```

- **加工规则**

```
e_if_else(
 op_or(e_match("result", r"(?i)ok|pass"), e_search("status== 200")), e_keep(), e_drop()
)
```

- **加工结果**

```
result: ok
status: 400
result: Pass
status: 200
```

## e\_switch

组合多个条件和操作。

- **函数格式**

`e_switch(条件1, 操作1, ……, default=None)`

**说明** 函数中条件和操作必须成对出现。

- **参数说明**

| 参数名称    | 参数类型   | 是否必填 | 说明                          |
|---------|--------|------|-----------------------------|
| 条件      | 任意     | 是    | 表达式或其组合。其结果不是布尔值时，会进行真假判断。  |
| 操作      | 全局操作函数 | 是    | 全局操作函数或其组合。                 |
| default | 全局操作函数 | 否    | 默认的全局操作函数或其组合。没有条件满足时执行该操作。 |

- **返回结果**

返回加工处理后的日志。

- **函数示例**

- a. 如果content字段的值为123，则将\_\_topic\_\_字段的值设置为Number。如果data字段的值为123，则将\_\_topic\_\_字段的值设置为PRO。

- **测试数据**

```
{
 "__topic__": ,
 "age": 18,
 "content": 123,
 "name": "maki",
 "data": 342
}
{
 "__topic__": ,
 "age": 18,
 "content": 23,
 "name": "maki",
 "data": 123
}
```

- **加工规则**

```
e_switch(
 e_search("content==123"),
 e_set("__topic__", "Number", mode="overwrite"),
 e_search("data==123"),
 e_set("__topic__", "PRO", mode="overwrite"),
)
```

- **加工结果**

```
__topic__: Number
age: 18
content: 123
name: maki
data: 342
__topic__: PRO
age: 18
content: 23
name: maki
data: 123
```

- b. 通过e\_switch语法和e\_output语法结合，将符合规则的日志投递到不同的Logstream。其中default=e\_drop()，表示将不满足规则的日志丢弃，不做投递处理。若不设置default参数，则表示不满足规则日志都会被投递到配置的第一个Logstream中。

output的加工结果不会显示到加工结果框中。

## ■ 测试数据

```
{
 "__topic__": "sas-log-dns",
 "test": "aa",
 "__topic__": "aegis-log-network",
 "test": "ecs",
 "__topic__": "local-dns",
 "test": "sls",
 "__topic__": "aegis-log-login",
 "test": "sls"
}
```

## ■ 加工规则

```
e_switch(e_match("__topic__", "sas-log-dns"),
e_output(name="target1"),
e_match("__topic__", "sas-log-process"),
e_output(name="target2"),
e_match("__topic__", "local-dns"),
e_output(name="target3"),
e_match("__topic__", "aegis-log-network"),
e_output(name="target4"),
e_match("__topic__", "aegis-log-login"),
e_output(name="target5"),
default=e_drop())
```

## ● 更多参考

支持和其他函数组合使用。

## 10.3 使用定时 SQL 进行日志加工

云日志服务提供定时SQL功能，用于定时分析日志内容。定时SQL支持标准的SQL语法，按照调度规则周期性执行日志分析，并将分析结果存储到目标日志流。

### 说明

目前此功能仅在华北-北京四、华南-广州、华东-上海一局点支持白名单用户提交工单申请使用，详细操作请参考[提交工单](#)，其他局点暂不支持该功能。

### 前提条件

- 已成功采集到日志。
- 对源日志内容已完成结构化配置，具体操作请参考[设置云端结构化解析日志](#)。

### 限制条件

最多可创建20个定时SQL任务。

### 创建定时 SQL

**步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。

**步骤2** 在左侧导航栏中选择“日志加工>定时SQL”，单击“创建定时SQL”。

或在左侧导航栏中选择“日志管理”，单击日志组或日志流名称，进入日志流详情页面，单击右上角日志设置，在弹出页面中，选择“定时SQL”，单击“创建定时SQL”。

- 通过该方式创建定时SQL时，源日志组/日志流为当前所选的日志组/日志流，且无法修改。

- 通过该方式创建的定时SQL任务，会同步显示在定时SQL列表中，单击任务名称可[查看详细信息](#)。

**步骤3** 在创建定时SQL页面中，配置相关参数。

- 在计算配置中，完成如下配置后，然后单击“下一步”。

**表 10-7** 计算配置参数

| 参数       | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 任务名称     | 定时SQL任务的名称。仅支持输入英文、数字、中文、中划线、下划线，且不能以中划线、下划线开头或结尾，长度范围为1~64个字符。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 描述       | 定时SQL任务的描述。长度不超过1000个字符。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 源日志组/流名称 | 选择已完成结构化配置的日志组/流，即表示源日志组/流中的日志内容通过定时SQL处理后，将存储到目标日志组/日志流中。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 统计类型     | <ul style="list-style-type: none"><li>搜索分析：新SQL引擎，使用管道符（搜索 分析）的方式。</li><li>SQL统计：旧SQL引擎，使用旧版SQL分析方式。旧版SQL引擎即将下线，推荐您使用搜索分析，功能更丰富。更多语法请参考<a href="#">SQL函数</a>。</li></ul>                                                                                                                                                                                                                                                                                                                                                                                             |
| SQL语句    | 输入的查询和分析语句，定时SQL运行时，云日志服务将执行该查询和分析语句分析日志。<br>选择时间范围，对该范围内的日志内容进行SQL查询；单击“预览”可查询预览结果。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 目标存储位置   | <p>选择LTS日志流，加工结果将会存储到LTS作为一条日志。</p> <ul style="list-style-type: none"><li>日志组：选择需要存储的日志组。</li><li>日志流：选择需要存储的日志流。</li></ul> <p>选择AOM Prometheus实例，加工结果将会存储到AOM中为自定义指标。</p> <p>委托授权：创建日志生成指标任务，需要您授权LTS创建云服务委托：lts_admin_trust。</p> <ul style="list-style-type: none"><li>Prometheus实例：选择需要存储的Prometheus实例。</li><li>指标列：LTS会根据您在SQL语句中输入的语法聚合数据，您可以选择查询结果中列值为数值类型的一列作为指标列。只支持添加5个指标列。</li><li>维度列：LTS会根据您SQL语句中输入的语法聚合数据，您可以选择查询结果中的一列或者多列作为维度列。只支持添加5个维度列。</li><li>时间列：默认，系统将以每次任务查询时间范围中的开始时间作为时序数据的时间。</li><li>标签：单击添加标签，分别填写标签键key和标签值value。支持添加20个标签。</li></ul> |

- 在调度配置中，完成如下配置。

表 10-8 调度配置参数

| 参数     | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 调度间隔   | <p>调度定时SQL任务的频率，每调度一次定时SQL任务将产生一个执行实例。调度间隔决定每个执行实例的调度时间。</p> <ul style="list-style-type: none"><li>每小时：每隔一小时调度一次定时SQL任务。</li><li>每天：在每天的某个固定时间点调度一次定时SQL任务。</li><li>每周：在周几的某个固定时间点调度一次定时SQL任务。</li><li>固定间隔：按照固定间隔调度定时SQL任务。</li><li>CRON：通过CRON表达式指定时间间隔，按照指定的时间间隔调度定时SQL任务。CRON表达式的最小精度为分钟，格式为24小时制，示例如下：</li></ul> <ul style="list-style-type: none"><li>0/10 * * * *从00:00开始，每隔整10分钟查询一次，分别为10分钟、20分钟、30分钟、40分钟、50分钟、60分钟。例如：当前时间为16:37，下一次查询时间为16:50。</li><li>0 0/5 * * * *从00:00开始，每隔5小时查询一次，分别为0时、5时、10时、15时、20时。例如：当前时间为16:37，下一次查询时间为20:00。</li><li>0 14 * * *每天14:00查询一次。</li><li>0 0 10 * *每月10日00:00查询一次。</li></ul> |
| 调度时间范围 | <p>调度的时间范围，具体说明如下：</p> <ul style="list-style-type: none"><li>某时间开始：实例调度的开始时间，不设时间范围。当该作业被删除时，则不产生新实例。</li><li>特定时间范围：按照调度间隔生成的实例调度时间必须在该范围内，超出范围则不产生新实例。</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 起始时间   | <p>调度的起始时间。</p> <p>当调度时间范围选择某时间开始时，需要设置开始时间；当调度时间范围选择特定时间范围时，需要设置起始时间。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

| 参数      | 说明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQL时间窗口 | <p>定时SQL任务运行时，仅分析该时间范围内的日志。时间窗口表达式的最小精度为秒（s：秒，m：分，h：小时），且SQL时间窗口设置不能超过24小时。可在下拉框中，选择5分钟、15分钟、1小时和1天。如果需要设置其他时间，可在下拉框中选择自定义，通过时间窗口表达式进行设置。时间表达式为：[+/-{num}h+/-{num}m+/-{num}s@h]。具体说明如下：</p> <ul style="list-style-type: none"><li>- “[”表示包含边界。</li><li>- “)”表示不包含边界。</li><li>- “+”表示时间从当前时间开始往后算起，例如：当前时间是16:00，+1h则表示17:00。不建议使用“+”。</li><li>- “-”表示时间从当前时间往前算起，例如：当前时间是16:00，-1h则表示15:00。</li><li>- “{num}”表示时间取值，为任意整数。逗号前后的时间差值不能超过24小时。</li><li>- “@”表示取整。@h：小时取整，分钟和秒忽略不计；@m：分钟取整，秒忽略不计；@s：秒取整。</li></ul> <p>时间表达式，举例如下：</p> <ul style="list-style-type: none"><li>- 5分钟：[-5m@m,@m)</li><li>- 15分钟：[-15m@m,@m)</li><li>- 1小时：[-1h@h,@h)</li><li>- 1天：[-24h@h,@h)</li><li>- 自定义（不超过一天）：[-65m@h,-5m@h)</li></ul> <p>时间表达式最大值为调度间隔的五倍。</p> |
| SQL超时   | <p>执行SQL分析操作失败时，自动重试的阈值。当超时时间超过指定的最大时间和超时次数超过最大次数时，该执行实例结束，状态为失败。</p> <ul style="list-style-type: none"><li>- SQL超时时间取值为60~180秒之间。</li><li>- SQL超时次数取值为1~10次之间。</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

**步骤4** 完成后，单击“确定”。

----结束

## 查看定时 SQL

在“定时SQL”页签，单击目标作业名称进入详情页面，可查看定时SQL任务的基本信息和执行实例。

每一个实例的上报的日志条数最多为100行。

## 修改定时 SQL

**步骤1** 在“定时SQL”页签，单击目标作业名称操作列对应的“修改”。

- 步骤2** 在弹出的页面中，修改[表10-8](#)。  
定时SQL的任务名称和源日志组/流名称无法进行修改。
- 结束

## 删除定时 SQL

- 步骤1** 在“定时SQL”页签，单击目标作业名称操作列对应的“删除”。
- 或单击目标作业名称，进入定时SQL任务的基本信息页面中，单击右上角“删除”。
- 步骤2** 在弹出的页面中，单击“确定”可删除定时SQL任务。
- 结束

## 10.4 使用 FunctionGraph 服务提供的函数模板进行日志加工

云日志服务提供函数加工，您可以基于函数服务提供的函数模板或者自定义函数，实现日志规整、流转、脱敏、过滤等功能。

### 进入函数模板页面

- 步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。
- 步骤2** 在左侧导航栏中选择“日志加工 > 函数加工”，进入函数模板页面。
- 步骤3** 根据实际需要，选择函数模板创建函数。
- 具体的操作步骤，请参见[函数工作流《用户指南》](#)。
- 结束

## 10.5 日志生成指标（邀测）

支持创建日志指标规则，将上报到LTS的日志数据提取为指标来统一管理，便于后续在应用运维管理控制台的指标浏览、仪表盘界面实时查看数据。

### 说明

目前此功能在邀测中，仅支持华北-北京四、华北-乌兰察布一局点的白名单用户内测使用。

### 背景信息

- 用户在LTS页面只需按照业务需要创建指标规则即可生成自己的统计报表，设置单个日志过滤条件或通过添加关联关系和添加组设置多个日志过滤条件，保留符合条件的日志，对用户特定时间范围内已结构化的日志进行动态统计，并将统计结果动态呈现到AOM的Prometheus实例，操作简单且功能强大。
- 创建的每个指标规则只能生成一个结果，多个结果则需要创建多条指标规则。
- 日志生成指标要求日志时间的顺序偏差在较小范围内（5s统计频率允许偏差5s，1min统计频率允许偏差1min，5min统计频率允许偏差1min30s），建议优先使用ICAgent结构化方式上报日志，云端结构化方式会引起日志时间乱序严重从而导致无法在统计周期内处理日志，使得统计结果存在偏差。

## 前提条件

- 已在应用运维管理控制台创建Prometheus实例。
- 已将日志接入到LTS。
- 已配置结构化数据，当前仅支持已配置结构化的数据进行处理。

## 限制条件

单个用户最多可创建10个日志指标规则，所有规则中添加的指标总数不能超过10。

## 创建日志指标规则

**步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。

**步骤2** 在左侧导航栏中选择“日志加工”。

**步骤3** 在“生成指标”页签，单击“创建规则”。

**步骤4** 配置日志的基本信息。

1. 填写规则名称，最多256位，只能包含字母、数字、下划线、中划线。
2. 填写描述信息。
3. 启用状态默认打开。
4. 默认勾选委托授权，创建日志生成指标任务，需要您授权LTS和AOM创建云服务委托：lts\_admin\_trust、aom\_admin\_trust
5. 任务监控，开启后会将每次任务执行状态写入日志流lts-system/lts-logtometric-statistics，您可以查看日志生成指标任务监控中心或者配置告警规则，及时发现加工过程中可能出现的异常问题。

**步骤5** 配置数据源。

1. 选择源日志组，若没有，则单击创建日志组。
  - 超出存储时间的日志将会被自动删除，您可以按需将日志数据转储至OBS桶中长期存储。
  - 如果您的日志尚未接入LTS，请参考[日志接入](#)，创建日志接入规则，并配置结构化解析规则。
2. 选择日志流，若没有，则单击创建日志流。
3. 日志采样，开启后会对日志源进行随机采样，采样率支持设置为0.1～0.9。

**步骤6** 配置指标存储位置。

1. 日志生成的指标会被存储到AOM中为自定义指标，请选择要存储的Prometheus实例。若没有，则单击创建实例。
2. 自定义日志生成指标的名称。只支持输入英文、数字、下划线、冒号，且不能以数字、下划线、冒号开头。
3. 填写指标含义。

**步骤7** 单击“下一步”。

**步骤8** 在“指标预览”下方预览信息。该预览信息是基于用户配置的日志过滤和统计规则，对日志流执行SQL查询模拟生成的指标结果，依赖用户先将日志采集到LTS，并配置好结构化解析规则和索引配置，否则此处预览结果展示为空。

**步骤9** 配置日志的统计方式。



1. 日志过滤的规则，设置完成后，支持预览效果。如果您无法在日志过滤和日志统计处选择到想要的日志字段，请您先在采集配置中配置好，详细操作请参考[设置云端结构化解析日志](#)。
  - 支持“或”“且”两种方式交互式过滤日志。
  - 不同字段支持的过滤规则不同。
  - 日志过滤和日志统计字段的类型仅支持string、float、long，不支持json。
  - 日志过滤是保留符合条件的日志，不符合条件的日志将被丢弃。
  - group by分组字段只支持字符串和整数类型。
2. 设置日志统计的字段，选择统计类型、支持选择或者自定义输入被统计的字段和分组字段。数据迟到1分钟，将不参与统计。
  - 支持以下统计类型：  
Count: 统计日志条数，CountKeyword: 统计关键词出现的次数，Sum: 统计指定字段求和值，Avg: 统计指定字段平均值，Max: 统计指定字段最大值，Min: 统计指定字段最小值，P50: 统计指定字段50%的值，P75: 统计指定字段75%的值，P90: 统计指定字段90%的值，P95: 统计指定字段95%的值，P99: 统计指定字段99%的值。
  - P系列统计类型是将数字排序后取xx%位置的值作为统计结果。
  - 日志统计是针对符合条件的日志进行操作，执行过程中会在日志组system生成一个日志流，如果删除该日志流将导致所有日志生成指标的规则执行详情无法查看。
3. 选择频率，支持5秒钟、1分钟、5分钟。频率不仅代表上报数据间隔，也代表统计操作的时间窗口，例如：频率5分钟具体含义为到达5分钟间隔后，取当前时间前5分钟数据进行统计操作并上报。
4. 设置完成后，根据查询中选择的内容自动生成维度。最多支持15个维度，建议设置5个维度。
5. 选择单位，例如选择角度、带宽、频率等数据的单位。

**步骤10** （可选）单击“实时日志预览”查看实时日志。

**步骤11** 单击“确定”。

**步骤12** 创建成功后，在生成指标页签下方，新增一条规则记录。

- 指标可视化和告警：创建成功后，您可以前往应用运维管理AOM控制台的仪表盘配置指标可视化图表，或者前往AOM告警规则配置指标告警。
- 建议每个日志流配置规则数量≤ 5个。
- 在对应规则的操作列，支持复制、修改、删除规则。

----结束

## 10.6 使用消费组管理消费数据（邀测）

### 10.6.1 创建消费组

您可以在华为云LTS控制台为日志流创建消费组，使用集成云日志服务Java SDK通过消费组消费数据。您无需关注SDK的实现细节以及消费者之间的负载均衡等，只需要关注业务逻辑。

云日志服务通过消费组消费数据，使用消费组功能之前，您需要了解以下基本概念。

### 说明

目前此功能在邀测中，暂不支持申请开通。

表 10-9 基本概念

| 概念         | 说明                                                                                                                                                                    |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 消费组        | <ul style="list-style-type: none"><li>一个消费组只能对应一个日志流且消费这个日志流的数据。</li><li>一个日志流最多支持创建30个消费组。</li><li>一个消费组可以被多个消费者消费，每个消费者不会重复消费数据。</li><li>同一个日志流下，消费组名称唯一。</li></ul> |
| 消费者        | 消费组的基本构成单元，承担消费任务，可以根据您的服务自行决定消费者数量。                                                                                                                                  |
| Shard      | 云日志服务将数据必定保存在某一个Shard中，是数据存储的基本单元。                                                                                                                                    |
| CheckPoint | 消费点位，是消费者消费到的最新位置。程序重启后，SDK会自动通过CheckPoint恢复上次的消费进度。                                                                                                                  |

## 消费逻辑

一个日志流包含多个Shard，通过消费组消费数据就是将Shard分配给消费组的消费者，分配规则如下：

- 在消费过程中，一个Shard只会分配给一个消费者
- 在消费过程中，一个消费者可以消费多个Shard

当消费组新增/减少消费者，或者日志流的Shard数量增加时，这个时候消费组的Shard会重新分配给当前的消费者，以实现消费者的负载均衡。分配规则不变如上。

## 创建消费组

**步骤1** 登录[云日志服务控制台](#)，进入“日志管理”页面。

**步骤2** 单击日志组名称对应的  按钮。

**步骤3** 单击日志流名称进入日志流详情页面，单击右上角日志设置，在弹出页面中，选择“日志消费”页签。

**步骤4** 单击新建消费组，填写消费组名称，单击“确定”。

**步骤5** （可选）当某个消费组，消费到一定位置时，如果想重头或者指定一个时间继续进行消费，支持重置消费位点。

单击重置消费点位，支持设置最早位置和指定时间点，需要重启SDK后，重置消费位点才能生效。

- 最早位置：从头开始一直拉取到当前最新时间的日志。

- 指定时间点：从指定的时间点一直拉取到当前最新时间的日志。
- 消费期间重置消费位点，会导致消费日志紊乱，建议停止消费后再重置消费位点。

----结束

## 10.6.2 使用 Java SDK 管理消费组

消费组创建完成后，您可以使用集成云日志服务Java SDK通过消费组消费数据。

### 说明

目前此功能在邀测中，暂不支持申请开通。

### 前提条件

- 使用云日志SDK前，您需要注册华为云账号，并开通云日志服务。
- 确认云日志服务的区域，请用户根据所在区域，获取regionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“[我的凭证 > API凭证](#)”。
- 获取需要上报到LTS的日志组ID、日志流ID、消费组名称。
- 云日志服务SDK仅支持在华为云ECS主机上使用。
- 当用户修改权限后，权限信息在一天后生效。

### 安装 SDK

在Maven项目中加入依赖项。

1. maven构建时，settings.xml文件的mirrors节点需要增加[华为外部开源仓](#)。如下：

```
<mirror>
 <id>huaweicloud</id>
 <mirrorOf>*</mirrorOf>
 <url>https://repo.huaweicloud.com/repository/maven/</url>
</mirror>
```

2. 您可以在[华为外部开源仓](#)中获取日志服务Java SDK依赖的最新版本。
3. 在Maven工程中使用日志服务Java SDK，只需在pom.xml中加入相应依赖即可，Maven项目管理工具会自动下载相关JAR包。

推荐使用最新版本version：1.1.3，第2步中查看SDK版本。

```
<dependency>
 <groupId>io.github.huaweicloud</groupId>
 <artifactId>lts-sdk-common</artifactId>
 <version>version</version>
</dependency>
<dependency>
 <groupId>io.github.huaweicloud</groupId>
 <artifactId>lts-sdk-java</artifactId>
 <version>version</version>
</dependency>
```

### 配置参数说明

LogConsumerConfig中配置参数说明：

表 10-10 配置参数说明

| 参数名称              | 描述                                                                                                                                                                             | 类型                                                         | 是否需要填写 | 默认值   |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|--------|-------|
| regionName        | 云日志服务的区域                                                                                                                                                                       | String                                                     | 必填     | -     |
| projectId         | 华为云账号的项目ID（project id）                                                                                                                                                         | String                                                     | 必填     | -     |
| logGroupId        | LTS的日志组ID                                                                                                                                                                      | String                                                     | 必填     | -     |
| logStreamId       | LTS的日志流ID                                                                                                                                                                      | String                                                     | 必填     | -     |
| accessKeyId       | 华为云账号的AK                                                                                                                                                                       | String                                                     | 必填     | -     |
| accessKeySecret   | 华为云账号的SK                                                                                                                                                                       | String                                                     | 必填     | -     |
| consumerGroupName | LTS日志流对应的消费组名称                                                                                                                                                                 | String                                                     | 必填     | -     |
| startTimeNs       | 消费开始时间，纳秒值                                                                                                                                                                     | Long                                                       | 必填     | -     |
| endTimeNs         | 消费结束时间，纳秒值                                                                                                                                                                     | Long                                                       | 选填     | -     |
| batchSize         | 每批次消费数量，可选范围[10, 1000]                                                                                                                                                         | Integer                                                    | 选填     | 1000  |
| consumerPosition  | BEGIN_CURSOR: 消费组从头开始消费日志，起始消费位点为logStream中的第一条日志。<br>END_CURSOR: 此消费位点记录logStream日志的最后一条日志之后。<br>(SDK版本最低必须为1.0.7支持此功能)                                                       | ConsumerPosition<br>枚举:<br>BEGIN_CURSOR<br>、<br>END_CURSOR | 选填     | -     |
| enableLocalTest   | 是否开启跨云消费日志（SDK版本最低必须为1.1.0支持此功能）。此功能仅适用于开发调测，对于消费性能不做保障。 <ul style="list-style-type: none"><li>选true时，开启跨云后用户能通过公网（华东-上海一）访问。</li><li>选false时，关闭跨云后用户是通过华为云当前区域主机访问。</li></ul> | boolean                                                    | 选填     | false |
| historyQuery      | 是否消费历史数据（SDK版本最低必须为1.1.0支持此功能）。 <ul style="list-style-type: none"><li>选true时，可以消费开启日志流引擎之前的数据。</li><li>选false时，只能消费当开启了日志流引擎之后的数据。</li></ul>                                   | boolean                                                    | 选填     | false |

| 参数名称    | 描述             | 类型     | 是否需要填写 | 默认值 |
|---------|----------------|--------|--------|-----|
| proxylp | 使用自定义ip端口上报日志。 | String | 选填     | -   |

1. startTimeNs（消费开始时间）和endTimeNs（消费结束时间），这里的时间指的是服务端时间。
2. SDK消费保证最终一致性，即您可以获取到这条日志流的全部内容，但由于时间为服务端时间，所以在获取日志流的过程中，可能导致获取到的日志数量跟LTS页面查询的日志数量不一致。

通过startTimeNs/endTimeNs可以在Shard中定位生命周期内的日志，假设LogStream日志流的生命周期为[begin\_time,end\_time)，startTimeNs/endTimeNs=from\_time，因此：

- 当from\_time ≤ begin\_time or from\_time = "begin"时：返回时间点为begin\_time对应的Cursor位置。
- 当from\_time ≥ end\_time or from\_time = "end"时：返回当前时间点下一条将被写入的Cursor位置（当前该Cursor位置上无数据）。
- 当from\_time > begin\_time and from\_time < end\_time时：返回第一个服务端接收时间大于等于from\_time的数据包对应的Cursor。

## 消费数据示例代码

- 创建消费者逻辑代码DemoLogConsumerProcessor.java

```
public class DemoLogConsumerProcessor implements ILogConsumerProcessor {
 @Override
 // 这个方法给您回调返回的ShardId, 是告诉您当前这个shard-consumer在消费那个shard
 public void initialize(String shardId) {
 }

 @Override
 // 数据处理方法, logGroups为拉取到的日志
 public String process(List<LogData> logGroups, ILogConsumerCheckPointTracker
 checkPointTracker) {
 for (LogData logData : logGroups) {
 // logData为您的一条日志, 日志内容在Labels属性中。
 // Labels为一个JSON, 存放您的这个条日志的内容, 比如: "log_content": "日志内容"
 System.out.println("这条日志内容: " + logData.getLabels());
 }
 // 方法的返回值为一个checkPoint
 // 如果您在处理这批数据的时候, 遇到什么异常或者需要重新获取这一次的数据, 那么 return
 checkPointTracker.getCurrentCursor();
 return null;
 }

 @Override
 // 当调用ClientConsumerWorker的shutdown方法, 会调用此函数, 您可以在此处写一些关闭流程
 public void shutdown(ILogConsumerCheckPointTracker checkPointTracker) {
 try {
 // 关闭前, 立即保存checkPoint
 checkPointTracker.saveCheckPoint(true);
 } catch (LogConsumerCheckPointException e) {
 e.printStackTrace();
 }
 }
}
```

- 创建消费者实体DemoLogConsumerProcessorFactory.java

```
import com.huaweicloud.lts.consumer.interfaces.ILogConsumerProcessor;
import com.huaweicloud.lts.consumer.interfaces.ILogConsumerProcessorFactory;

public class DemoLogConsumerProcessorFactory implements ILogConsumerProcessorFactory {
 @Override
 public ILogConsumerProcessor generatorProcessor() {
 // 通过这个方法会为每个Shard生成一个处理数据的DemoLogConsumerProcessor.java
 // 您也可以将返回值变成一个单例, 这样处理的话单个消费者所消费的所有Shard数据都会走这个单例
 // 流程
 return new DemoLogConsumerProcessor();
 }
}
```

- 创建TestDemo.java文件。创建消费者并启动消费者线程, 该消费者会从指定的logStream中消费数据。

```
import com.huaweicloud.lts.consumer.ClientConsumerWorker;
import com.huaweicloud.lts.consumer.config.LogConsumerConfig;
import com.huaweicloud.lts.producer.exception.LogException;

public class TestDemo {
 // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险, 建议在配置文件或者环境变量中密
 // 文存放, 使用时解密, 确保安全;
 // 本示例以ak和sk保存在环境变量中为例, 运行本示例前请先在本地环境中设置环境变量
 // HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK.
 String ak = System.getenv("HUAWEICLOUD_SDK_AK");

 String sk = System.getenv("HUAWEICLOUD_SDK_SK");

 // 云日志服务的区域
 private static final String TEST_REGION_NAME = "TEST_REGION_NAME";

 // 华为云账号的项目ID (project id)
 private static final String TEST_PROJECT = "TEST_PROJECT";

 // LTS的日志组ID
 private static final String TEST_LOG_GROUP_ID = "TEST_LOG_GROUP_ID";

 // LTS的日志流ID
 private static final String TEST_LOG_STREAM_ID = "TEST_LOG_STREAM_ID";

 // 华为云账号的AK
 private static final String ACCESS_KEY_ID = "ACCESS_KEY_ID";

 // 华为云账号的SK
 private static final String ACCESS_KEY_SECRET = "ACCESS_KEY_SECRET";

 // LTS日志流对应的消费组名称, 请您从云日志服务LTS页面创建日志流的消费组
 private static final String CONSUMER_GROUP_NAME = "CONSUMER_GROUP_NAME";

 // 启动消费者数量, 由客户根据自身资源状态决定启动多少消费者
 private static final Integer CONSUMER_COUNT = 5;

 // 消费开始时间
 private static final Long START_TIME = 1685444710000000000L;

 public static void main(String[] args) throws LogException, InterruptedException {
 // 多个消费者可以在多台机器上启动, 会均衡消费一个消费组
 // 如下案例, 启动5个消费者开始消费.
 Thread[] threads = new Thread[CONSUMER_COUNT];
 ClientConsumerWorker[] workers = new ClientConsumerWorker[CONSUMER_COUNT];
 for (int i = 0; i < CONSUMER_COUNT; i++) {
 // 构建消费者配置, 参数有必填的: regionName, projectId, logGroupId, logStreamId, ak, sk,
 // consumerGroupName, startTime
 // 当消费组名称不存在, SDK会自动为日志流创建消费组。
 LogConsumerConfig config = new LogConsumerConfig(TEST_REGION_NAME, TEST_PROJECT,
 TEST_LOG_GROUP_ID,
 TEST_LOG_STREAM_ID, ACCESS_KEY_ID, ACCESS_KEY_SECRET, CONSUMER_GROUP_NAME,
 START_TIME);
 workers[i] = new ClientConsumerWorker(config);
 threads[i] = new Thread(workers[i]);
 threads[i].start();
 }
 }
}
```

```
// 构建消费者的工作类
ClientConsumerWorker worker = new ClientConsumerWorker(new
DemoLogConsumerProcessorFactory(), config);
threads[i] = new Thread(worker);
workers[i] = worker;
}
for (int i = 0; i < CONSUMER_COUNT; i++) {
// 启动消费者, ClientConsumerWorker实现了Runnable接口, Thread启动后, 内置的消费任务会自动运行
threads[i].start();
}
Thread.sleep(60 * 60 * 1000);
for (int i = 0; i < CONSUMER_COUNT; i++) {
// 调用ClientConsumerWorker的shutdown方法, 安全的关闭消费者, 消费者中启动的内置线程也会自动停止
workers[i].shutdown();
}
// 调用ClientConsumerWorker的shutdown方法后, 由于消费者内置多个异步任务, 建议停止1分钟再关闭整个服务, 目的就是让消费者完成后台的异步任务, 安全的退出
// 如果消费者突然关闭, 没有调用shutdown方法; 或者调用shutdown方法之后, 没有等待一定的时间. 那么可能造成下次消费时, 会有一定的重复数据, 因为消费者后台的异步任务没有保存checkPoint点
Thread.sleep(60 * 1000);
}
}
```

## 按照不同的消费需求, 创建不同的 LogConsumerConfig

1. 指定消费开始时间, 不指定结束时间。

```
// 设置batchSize = 500, 即每次拉取500条。
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT","TEST_LOG_GROUP_ID","TEST_LOG_STREAM_ID","ACCESS_KEY_ID","ACCESS_KEY_SECRET","CONSUMER_GROUP_NAME", 1685444710000000000L, 500);

// 如果需要按照日志组/日志流名称消费
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT","TEST_LOG_GROUP_NAME","TEST_LOG_STREAM_NAME",
true,"ACCESS_KEY_ID","ACCESS_KEY_SECRET","CONSUMER_GROUP_NAME",
1685444710000000000L, 500);
```

2. 指定消费开始时间, 且指定结束时间。

```
// 设置batchSize = 500, 即每次拉取500条
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT","TEST_LOG_GROUP_ID","TEST_LOG_STREAM_ID","ACCESS_KEY_ID","ACCESS_KEY_SECRET","CONSUMER_GROUP_NAME",1685444710000000000L, 1685445470192043318L, 500);

// 如果需要按照日志组/日志流名称消费
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT","TEST_LOG_GROUP_NAME","TEST_LOG_STREAM_NAME",
true,"ACCESS_KEY_ID","ACCESS_KEY_SECRET","CONSUMER_GROUP_NAME",1685444710000000000L,
1685445470192043318L, 500);
```

3. 指定ConsumerPosition ( BEGIN\_CURSOR、END\_CURSOR )

```
// 指定 BEGIN_CURSOR 开始消费
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT","TEST_LOG_GROUP_ID","TEST_LOG_STREAM_ID",
"ACCESS_KEY_ID","ACCESS_KEY_SECRET","CONSUMER_GROUP_NAME",
ConsumerPosition.BEGIN_CURSOR);

// 指定 END_CURSOR 开始消费
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT","TEST_LOG_GROUP_ID","TEST_LOG_STREAM_ID",
"ACCESS_KEY_ID","ACCESS_KEY_SECRET","CONSUMER_GROUP_NAME",
ConsumerPosition.END_CURSOR);
```

4. 支持使用临时AK、SK、STS-Token消费数据 ( 若使用临时STS-Token, 则AK、SK需要填写与STS-Token配套的临时AK、SK )。

- a. 创建STS-Token更新类DemoLogConsumerSTSToken.java ( 由于临时凭证会存在过期, 为了不影响您的消费业务, 需要实时更新临时凭证 )

```
import com.huaweicloud.lts.consumer.interfaces.ILogConsumerSTSToken;
import com.huaweicloud.lts.producer.STSTokenConfig;

public class DemoLogConsumerSTSToken implements ILogConsumerSTSToken {

 @Override
 // SDK会定期从此方法中获取临时AK, 临时SK, 临时securityToken. 如果临时认证信息有变化, 在此方法中实现即可
 public STSTokenConfig getSTSTokenConfig() {
 String accessKeyId = "";
 String accessKeySecret = "";
 String securityToken = "";
 return new STSTokenConfig(accessKeyId, accessKeySecret, securityToken);
 }
}
```

b. 创建LogConsumerConfig

```
// 指定消费开始时间, 不指定结束时间
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT", "TEST_LOG_GROUP_ID", "TEST_LOG_STREAM_ID", new
DemoLogConsumerSTSToken("ACCESS_KEY_ID", "ACCESS_KEY_SECRET", "STS-TOKEN"),
"CONSUMER_GROUP_NAME", 1685444710000000000L);

// 指定消费开始时间, 且指定结束时间
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT", "TEST_LOG_GROUP_ID", "TEST_LOG_STREAM_ID", new
DemoLogConsumerSTSToken("ACCESS_KEY_ID", "ACCESS_KEY_SECRET", "STS-TOKEN"),
"CONSUMER_GROUP_NAME", 1685444710000000000L, 1685445470192043318L);

// 指定 BEGIN_CURSOR
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT", "TEST_LOG_GROUP_ID", "TEST_LOG_STREAM_ID", new
DemoLogConsumerSTSToken("ACCESS_KEY_ID", "ACCESS_KEY_SECRET", "STS-TOKEN"),
"CONSUMER_GROUP_NAME", ConsumerPosition.BEGIN_CURSOR);

// 指定 END_CURSOR
LogConsumerConfig config = new LogConsumerConfig("TEST_REGION_NAME",
"TEST_PROJECT", "TEST_LOG_GROUP_ID", "TEST_LOG_STREAM_ID", new
DemoLogConsumerSTSToken("ACCESS_KEY_ID", "ACCESS_KEY_SECRET", "STS-TOKEN"),
"CONSUMER_GROUP_NAME", ConsumerPosition.END_CURSOR);
```

5. 开启跨云消费日志。此功能仅适用于开发调测，对于消费性能不做保障

```
LogConsumerConfig config = new LogConsumerConfig (....);
config.setEnableLocalTest(true);
```

6. 开启消费历史数据。

```
LogConsumerConfig config = new LogConsumerConfig (....);
config.setHistoryQuery(true);
```

7. 使用自定义IP消费日志

```
LogConsumerConfig config = new LogConsumerConfig (....);
config.setProxyIp("127.0.0.1:8102");
```

## 获取日志组和日志流 ID

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬停在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬停在日志流名称上，可查看日志流名称和日志流ID。

## 10.6.3 使用 Go SDK 管理消费组

消费组创建完成后，您可以使用集成云日志服务Go SDK通过消费组消费数据。



 说明

目前此功能在邀测中，暂不支持申请开通。

约束限制

消费历史日志时，只支持开通公测白名单时间点后的日志。

前提条件

- 使用云日志SDK前，您需要注册华为云账号，并开通云日志服务。
- 确认云日志服务的区域，请用户根据所在区域，获取regionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（ project id ），步骤参考：请参见“我的凭证 > [API凭证](#)”。
- 获取需要上报到LTS的日志组ID、日志流ID、消费组名称。
- 云日志服务SDK仅支持在华为云ECS主机上使用。
- 当用户修改权限后，权限信息在一天后生效。

操作步骤

您可以通过“在Maven项目中加入依赖项”的方式安装云日志服务Go SDK。

1. 获取云日志服务的Go SDK包：

go get github.com/huaweicloud/huaweicloud-lts-sdk-go
2. 引用云日志服务的Go SDK包：

import github.com/huaweicloud/huaweicloud-lts-sdk-go

配置参数说明

LogConsumerConfig中配置参数说明：

表 10-11 配置参数说明

| 参数名称              | 描述                        | 类型     | 是否需要填写 | 默认值 |
|-------------------|---------------------------|--------|--------|-----|
| RegionName        | 云日志服务的区域。                 | string | 必填     | -   |
| ProjectId         | 华为云账号的项目ID（ project id ）。 | string | 必填     | -   |
| LogGroupId        | LTS的日志组ID。                | string | 必填     | -   |
| LogStreamId       | LTS的日志流ID。                | string | 必填     | -   |
| AccessKeyId       | 华为云账号的AK。                 | string | 必填     | -   |
| AccessKeySecret   | 华为云账号的SK。                 | string | 必填     | -   |
| ConsumerGroupName | LTS日志流对应的消费组名称。           | string | 必填     | -   |

| 参数名称          | 描述                      | 类型        | 是否需要填写 | 默认值  |
|---------------|-------------------------|-----------|--------|------|
| StartTimeNs   | 消费开始时间，单位是毫秒。           | time.Time | 必填     | -    |
| EndTimeNs     | 消费结束时间，单位是毫秒。           | time.Time | 选填     | -    |
| BatchSize     | 每批次消费数量，可选范围[10, 1000]。 | int       | 选填     | 1000 |
| consumerCount | 消费者的数量。                 | int       | 选填     | 1    |

1. startTimeNs（消费开始时间）和endTimeNs（消费结束时间），这里的时间指的是服务端时间。
2. SDK消费保证最终一致性，即您可以获取到这条日志流的全部内容，但由于时间为服务端时间，所以在获取日志流的过程中，可能导致获取到的日志数量跟LTS页面查询的日志数量不一致。

通过startTimeNs/endTimeNs可以在Shard中定位生命周期内的日志，假设LogStream日志流的生命周期为[begin\_time,end\_time)，startTimeNs/endTimeNs=from\_time，因此：

- 当from\_time ≤ begin\_time or from\_time = "begin"时：返回时间点为begin\_time对应的Cursor位置。
- 当from\_time ≥ end\_time or from\_time = "end"时：返回当前时间点下一条将被写入的Cursor位置（当前该Cursor位置上无数据）。
- 当from\_time > begin\_time and from\_time < end\_time时：返回第一个服务端接收时间大于等于from\_time的数据包对应的Cursor。

## 消费数据示例代码

```
package main

import (
)

var allShardLogCount int64

func init() {
 allShardLogCount = 0
}

"github.com/huaweicloud/huaweicloud-lts-sdk-go/consumer"
"github.com/huaweicloud/huaweicloud-lts-sdk-go/producer"
"github.com/sirupsen/logrus"
"log/slog"
"sync/atomic"
"time"

func ConsumeLog(regionName, projectId, logGroupId, logStreamId, ak, sk, consumerGroupName, logLevel,
logDest string, consumeCount, batchSize int, startTime, endTime int64) {
 // 消费开始时间，括号中填毫秒值
 var StartTime time.Time
 if startTime != 0 {
 StartTime = time.UnixMilli(startTime)
 }
 // 消费结束时间
```

```
var EndTime time.Time
if endTime != 0 {
 EndTime = time.UnixMilli(endTime)
}
var logConfig producer.LogConf
if logDest == "file" {
 logConfig = producer.LogConf{
 Dir: "/opt/clouds",
 Name: "lts-go-sdk.log",
 Level: logLevel,
 MaxSize: 100,
 }
 producer.InitLoggerFile(logConfig)
} else {
 logConfig = producer.LogConf{
 Dir: "",
 Name: "",
 Level: logLevel,
 MaxSize: 100,
 }
 producer.InitLoggerStd(logConfig)
}
slog.Info("region is: ", "region", regionName)
slog.Info("projectId is: ", "region", projectId)
slog.Info("logGroupId is: ", "logGroupId", logGroupId)
slog.Info("ak is: ", "ak", ak)
slog.Info("sk is: ", "sk", sk)
slog.Info("consumerGroupName is: ", "consumerGroupName", consumerGroupName)
slog.Info("consumerCount is: ", "consumerCount", consumeCount)
slog.Info("batchSize is: ", "batchSize", batchSize)
slog.Info("start time is: ", "startTime", StartTime)
slog.Info("end time:", "endTime", EndTime, "endTime is Zero", EndTime.IsZero())
workers := make([]*consumer.ClientConsumerWorker, 0)
for i := 0; i < consumeCount; i++ {
 config := consumer.GetConsumerConfig()
 // 构建消费者配置, 参数有必填的: regionName, projectId, logGroupId, logStreamId, ak, sk,
 consumerGroupName, startTime
 config.ProjectId = projectId
 config.LogGroupId = logGroupId
 config.LogStreamId = logStreamId
 config.AccessKeyId = ak
 config.AccessKeySecret = sk
 config.BatchSize = batchSize //BatchSize默认值1000
 config.StartTimeNs = StartTime
 config.EndTimeNs = EndTime
 config.ConsumerGroupName = consumerGroupName
 config.RegionName = regionName
 // 构建消费者的工作者
 worker := consumer.GetClientConsumerWorker(new(DemoLogConsumerProcessorFactory), config)
 workers = append(workers, worker)
}
for _, work := range workers {
 // 启动消费者, ClientConsumerWorker启动后, 内置的消费任务会自动运行
 work.Run()
}
time.Sleep(30 * time.Minute)
for _, work := range workers {
 // 调用ClientConsumerWorker的shutdown方法, 安全的关闭消费者, 消费者中启动的内置线程也会自动停止
 work.Shutdown()
}
// 调用ClientConsumerWorker的shutdown方法后, 由于消费者内置多个异步任务, 建议停止1分钟在关闭整个
// 服务, 目的就是让消费者完成后台的异步任务, 安全的退出
// 如果消费者突然关闭, 没有调用shutdown方法; 或者调用shutdown方法之后, 没有等待一定的时间. 那么可能
// 造成下次消费时, 会有一定的重复数据, 因为消费者后台的异步任务没有保存checkPoint点
time.Sleep(time.Minute)
}
type DemoLogConsumerProcessor struct {
 LogCount int
}
```

```
// Initialize这个方法给您回调返回的ShardId, 是告诉您当前这个shard-consumer在消费那个shard
func (processor *DemoLogConsumerProcessor) Initialize(shardId string) {
}
// Process数据处理方法, logGroups为拉取到的日志
func (processor *DemoLogConsumerProcessor) Process(logGroups []consumer.LogData, checkPointTracker
consumer.ILogConsumerCheckPointTracker) string {
 atomic.AddInt64(&allShardLogCount, int64(len(logGroups)))
 processor.LogCount = processor.LogCount + len(logGroups)
 slog.Info("this time process log", "consume log", len(logGroups), "total log num", processor.LogCount)
 slog.Info("after this consume", "consume log", len(logGroups), "all shard consume total log num",
allShardLogCount)
 logrus.WithField("consume log", len(logGroups)).WithField("total log num",
processor.LogCount).Info("this time process log")
 logrus.WithField("consume log", len(logGroups)).WithField("all shard consume total log num",
allShardLogCount).Info("after this consume")
 //for _, logData := range logGroups {
 // // logData为您的一条日志, 日志内容在Labels属性中。
 // // Labels为一个JSON, 存放您的这个条日志的内容, 比如: "log_content": "日志内容"
 // fmt.Println(fmt.Sprintf("日志内容: %v", logData.Labels))
 //}
 // 方法的返回值为一个checkPoint
 // 如果您在处理这批数据的时候, 遇到什么异常或者说想重新获取这一次的数据, 那么 return
checkPointTracker.GetCurrentCursor();
 return ""
}
// Shutdown 当调用ClientConsumerWorker的shutdown方法, 会调用此函数, 您可以在此处写一些关闭流程
func (processor *DemoLogConsumerProcessor) Shutdown(checkPointTracker
consumer.ILogConsumerCheckPointTracker) error {
 // 关闭前, 立即保存checkPoint
 return checkPointTracker.SaveCheckPoint(true)
}
type DemoLogConsumerProcessorFactory struct {
}
func (processor *DemoLogConsumerProcessorFactory) GeneratorProcessor()
consumer.ILogConsumerProcessor {
 demoProcessor := new(DemoLogConsumerProcessor)
 demoProcessor.LogCount = 0
 return demoProcessor
}
```

## 获取日志组和日志流 ID

- 日志组ID: 登录云日志服务页面, 选择“日志管理”, 鼠标悬浮在日志组名称上, 可查看日志组名称和日志组ID。
- 日志流ID: 单击日志组名称对应的  按钮, 鼠标悬浮在日志流名称上, 可查看日志流名称和日志流ID。

## 10.6.4 使用 Python SDK 管理消费组

消费组创建完成后, 您可以使用集成云日志服务Python SDK通过消费组消费数据。

### 说明

目前此功能在邀测中, 暂不支持申请开通。

## 约束限制

- Python SDK适用于python 3.10.1及以上版本。
- 消费历史日志时, 只支持开通公测白名单时间点后的日志。

## 前提条件

- 使用云日志SDK前，您需要注册华为云账号，并开通云日志服务。
- 确认云日志服务的区域，请用户根据所在区域，获取regionName。
- [获取华为账号的AK/SK](#)。
- 获取华为云账号的项目ID（ project id ），步骤参考：请参见“我的凭证 > [API凭证](#)”。
- 获取需要上报到LTS的日志组ID、日志流ID、消费组名称。
- 云日志服务SDK仅支持在华为云ECS主机上使用。
- 当用户修改权限后，权限信息在一天后生效。

## 操作步骤

**步骤1** 获取LTS Python SDK包源码。

```
git clone https://gitee.com/lordstar-habile/huaweicloud-lts-python-sdk.git
```

**步骤2** 安装相关依赖。

```
pip3 install requests
pip3 install loguru
pip3 install six
```

----结束

## 配置参数说明

LogConsumerConfig中配置参数说明：

表 10-12 配置参数说明

| 参数名称                | 描述                        | 类型        | 是否需要填写 | 默认值 |
|---------------------|---------------------------|-----------|--------|-----|
| region              | 云日志服务的区域。                 | string    | 必填     | -   |
| projectId           | 华为云账号的项目ID（ project id ）。 | string    | 必填     | -   |
| logGroup            | LTS的日志组ID。                | string    | 必填     | -   |
| logStream           | LTS的日志流ID。                | string    | 必填     | -   |
| ak                  | 华为云账号的AK。                 | string    | 必填     | -   |
| sk                  | 华为云账号的SK。                 | string    | 必填     | -   |
| consumer_group_name | LTS日志流对应的消费组名称。           | string    | 必填     | -   |
| start_time          | 消费开始时间，单位是纳秒。             | time.Time | 必填     | -   |
| end_time            | 消费结束时间，单位是纳秒。             | time.Time | 选填     | -   |

| 参数名称               | 描述       | 类型  | 是否需要填写 | 默认值  |
|--------------------|----------|-----|--------|------|
| consume_batch_size | 每批次消费数量。 | int | 选填     | 1000 |
| consumer_count     | 消费者的数量。  | int | 选填     | 1    |

1. startTimeNs（消费开始时间）和endTimeNs（消费结束时间），这里的时间指的是服务端时间。
2. SDK消费保证最终一致性，即您可以获取到这条日志流的全部内容，但由于时间为服务端时间，所以在获取日志流的过程中，可能导致获取到的日志数量跟LTS页面查询的日志数量不一致。

通过startTimeNs/endTimeNs可以在Shard中定位生命周期内的日志，假设LogStream日志流的生命周期为[begin\_time,end\_time)，startTimeNs/endTimeNs=from\_time，因此：

- 当from\_time ≤ begin\_time or from\_time = "begin"时：返回时间点为begin\_time对应的Cursor位置。
- 当from\_time ≥ end\_time or from\_time = "end"时：返回当前时间点下一条将被写入的Cursor位置（当前该Cursor位置上无数据）。
- 当from\_time > begin\_time and from\_time < end\_time时：返回第一个服务端接收时间大于等于from\_time的数据包对应的Cursor。

## 消费数据示例代码

```
import argparse
import sys
import time
from typing import List
from loguru import logger
from consumer.core.client_consumer_worker import get_client_consumer_worker
from consumer.interface.consumer_check_point_tracker import ILogConsumerCheckPointTracker
from consumer.interface.consumer_processor import ILogConsumerProcessor
from consumer.interface.consumer_processor_factory import ILogConsumerProcessorFactory
from consumer.model.consumer_config import ConsumerConfig
from consumer.model.log_data import LogData
class DemoLogConsumerProcessor(ILogConsumerProcessor):
 def __init__(self):
 self.log_count = 0
 def initialize(self, shard_id):
 pass
 def process(self, log_group: List[LogData], tracker: ILogConsumerCheckPointTracker) -> str:
 self.log_count += len(log_group)
 logger.info("this time consume log {} total log num {}", len(log_group), self.log_count)
 def shutdown(self, tracker: ILogConsumerCheckPointTracker):
 return tracker.save_check_point(True)
class DemoLogConsumerProcessorFactory(ILogConsumerProcessorFactory):
 def generate_processor(self):
 return DemoLogConsumerProcessor()
def start_consume(endpoint, region_name, project_id, log_group_id, log_stream_id, access_key, access_secret,
 consumer_group_name, start_time, end_time, log_level, is_test, consumer_count,
 consume_batch_size):
 logger.remove()
 logger.add(sys.stderr, level=log_level)
 if is_test:
 logger.add("/fanxin/log.log", level="DEBUG")
 logger.info("start time {}, end time {}", start_time, end_time)
```

```
workers = []
for i in range(consumer_count):
 config = ConsumerConfig(endpoint, region_name, project_id, log_group_id, log_stream_id, access_key,
 access_secret, consumer_group_name, start_time, end_time, consume_batch_size)
 work = get_client_consumer_worker(DemoLogConsumerProcessorFactory(), config)
 if work is None:
 logger.error("get work error, work is none")
 else:
 workers.append(work)
for work in workers:
 work.start()
time.sleep(30000 * 60)
for work in workers:
 work.shutdown()
time.sleep(60)
if __name__ == '__main__':
 parser = argparse.ArgumentParser(description='lts python sdk consume log help tester information')
 parser.add_argument('--endpoint', dest='endpoint', type=str, help='lts endpoint', default="https://")
 parser.add_argument('--project_id', dest='project_id', type=str, help='huaweicloud project_id',
 default="projectId")
 parser.add_argument('--region', dest='region', type=str, help='huaweicloud region', default="region")
 parser.add_argument('--group_id', dest='group_id', type=str, help='send log group_id', default="group_id")
 parser.add_argument('--stream_id', dest='stream_id', type=str, help='send log stream_id',
 default="stream_id")
 parser.add_argument('--ak', dest='ak', type=str, help='huaweicloud ak', default="ak")
 parser.add_argument('--sk', dest='sk', type=str, help='huaweicloud sk', default="sk")
 parser.add_argument('--consumer_count', dest='consumer_count', type=int, help='consumer count',
 default=1)
 parser.add_argument('--consume_batch_size', dest='consume_batch_size', type=int, help='each fetch max
log nums',
 default=1000)
 parser.add_argument('--consumer_group_name', dest='consumer_group_name', type=str, help='consumer
group name',
 default="consumer_group_name")
 parser.add_argument('--start_time', dest='start_time', type=int, help='consume log start time ms',
 default=213123123123)
 parser.add_argument('--end_time', dest='end_time', type=int, help='consume log end time ms', default=0)
 parser.add_argument('--log_level', dest='log_level', type=str, help='consume log end time ms',
 default="INFO")
 parser.add_argument('--is_test', dest='is_test', type=bool, help='test consumer will write log to node',
 default=True)
 args = parser.parse_args()
 start_consume(args.endpoint, args.region, args.project_id, args.group_id, args.stream_id, args.ak,
 args.sk, args.consumer_group_name, args.start_time, args.end_time, args.log_level, args.is_test,
 args.consumer_count, args.consume_batch_size)
```

## 获取日志组和日志流 ID

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的  按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

## 10.6.5 Flink 消费

云日志服务LTS提供的LTS-Flink-Connector工具用于对接flink，包括消费者(Consumer)和生产者(Producer)两部分。

消费者用于从云日志服务中读取数据，支持exactly once语义，支持shard负载均衡，生产者用于将数据写入云日志服务，使用LTS-Flink-Connector工具时，需要在项目中添加maven依赖：

```
<dependency>
 <groupId>io.github.huaweicloud</groupId>
```

```
<artifactId>lts-flink-connector</artifactId>
<version>1.1.3</version>
</dependency>
```

目前此功能在邀测中，暂不支持申请开通。

## 前提条件

- 已创建日志流，详细操作请参考[管理日志流](#)。
- 已创建消费组，详细操作请参考[创建消费组](#)。
- 已创建通用队列，详细操作请参考[创建弹性资源池并添加队列](#)。
- 已创建Flink作业，详细操作请参考[创建Flink OpenSource SQL作业](#)。
- 已从[maven仓库](#)获取lts-flink-connector-1.1.3.jar包，创建Flink Jar作业时需要上传lts-flink-connector-1.1.3.jar的程序包。在选择程序包之前需要将对应的Jar包上传至OBS桶中，并在“数据管理>程序包管理”中创建程序包，具体操作请参考[创建程序包](#)。

## FlinkLtsLogConsumer

FlinkLtsLogConsumer提供了订阅云日志服务中某一个日志组中单个或者多个日志流的能力，支持Exactly Once语义，在使用时，用户无需关心日志流中shard数量的变化，FlinkLtsLogConsumer会自动感知。

Flink中每一个subtask启动一个LTS SDK的consumer worker负责消费日志流中部分shard，如果日志流中shard发生split或者merge，consumer worker消费的shard也会随之改变。

### 1. 本地调试配置启动参数。

以下是一个简单的消费示例，使用java.util.Properties作为配置工具，所有Consumer的配置都可以在ConfigConstants中找到。

```
Properties configProps = new Properties();
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文
// 存放，使用时解密，确保安全
// LTS 日志服务所属region
configProps.put(ConfigConstants.LOG_REGION_NAME, REGION_NAME);
// LTS 日志租户项目ID
configProps.put(ConfigConstants.LOG_PROJECT_ID, LOG_PROJECT_ID);
// LTS 日志组ID
configProps.put(ConfigConstants.LOG_GROUP_ID, LOG_GROUP_ID);
// LTS 日志流ID
configProps.put(ConfigConstants.LOG_STREAM_ID, LOG_STREAM_ID);
// LTS 租户AK
configProps.put(ConfigConstants.ACCESS_KEY_ID, ACCESS_KEY_ID);
// LTS 租户SK
configProps.put(ConfigConstants.ACCESS_KEY_SECRET, ACCESS_KEY_SECRET);
// LTS 日志流的消费组
configProps.put(ConfigConstants.CONSUMER_GROUP_NAME, CONSUMER_GROUP);
configProps.put(ConfigConstants.LOG_MAX_NUMBER_PER_FETCH, "10");
configProps.put(ConfigConstants.LOG_COMMIT_INTERVAL_MILLIS, "10000");
// LTS 日志消费结束位置
configProps.put(ConfigConstants.STOP_TIME, "");
// LTS 日志消费起始位置
configProps.put(ConfigConstants.START_TIME, 1689836602157000000L);
LogDataDeserializer deserializer = new LogDataDeserializer();
DataStream<LogDataList> stream = env.addSource(
 new FlinkLtsLogConsumer<>(LOG_STREAM_ID, deserializer, configProps));
```

### 2. 监控消费进度(可选)



FlinkLtsLogConsumer支持设置消费进度监控，所谓消费进度就是获取每一个shard实时的消费位置，这个位置使用时间戳表示，详细操作可以参考文档[创建消费组](#)。

## FlinkLtsLogProducer

FlinkLtsLogProducer用于将数据写到云日志服务中。producer只支持Flink at-least-once语义，在发生作业失败的情况下，写入云日志服务中的数据有可能会重复，但是绝对不会丢失。

### 1. 用法示例如下，将模拟产生的字符串写入云日志服务：

```
// 将数据序列化成日志服务的数据格式
return row -> {
 Map<String, Object> contentMap = new HashMap<>();
 // 把flink SQL指定的列数据放到一个map
 for (int i = 0; i < rowArity; i++) {
 contentMap.put(fieldNames[i], fieldConverters[i].convert(row, i));
 }
 LOGGER.debug("row data convert to lts log, data is {}", contentMap);
 // logTimeNs是必填字段，获取当前时间
 long logTimeNs = System.currentTimeMillis() * 1000000L + System.nanoTime() % 1000000L;
 LogContent logContent = new LogContent();
 logContent.setLog(contentMap.toString());
 logContent.setLogTimeNs(logTimeNs);
 return logContent;
};

public class FlinkProducerTest {
 private static final String REGION_NAME = "region name";
 private static final String LOG_PROJECT_ID = "*****";
 private static final String LOG_GROUP_ID = "*****";
 private static final String LOG_STREAM_ID = "*****";
 private static final String ACCESS_KEY_ID = "*****";
 private static final String ACCESS_KEY_SECRET = "*****";
 public static void main(String[] args) throws Exception {
 final ParameterTool params = ParameterTool.fromArgs(args);
 final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
 env.getConfig().setGlobalJobParameters(params);
 env.setParallelism(1);
 DataStream<String> stream = env.addSource(new EventsGenerator());
 Properties configProps = new Properties();
 // LTS 日志服务所属region
 configProps.put(ConfigConstants.LOG_REGION_NAME, REGION_NAME);
 // LTS 日志租户项目ID
 configProps.put(ConfigConstants.LOG_PROJECT_ID, LOG_PROJECT_ID);
 // LTS 日志组ID
 configProps.put(ConfigConstants.LOG_GROUP_ID, LOG_GROUP_ID);
 // LTS 日志流ID
 configProps.put(ConfigConstants.LOG_STREAM_ID, LOG_STREAM_ID);
 // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中
 // 密文存放，使用时解密，确保安全
 // LTS 租户AK
 configProps.put(ConfigConstants.ACCESS_KEY_ID, ACCESS_KEY_ID);
 // LTS 租户SK
 configProps.put(ConfigConstants.ACCESS_KEY_SECRET, ACCESS_KEY_SECRET);
 LogDataDeserializer deserializer = new LogDataDeserializer();
 FlinkLtsLogProducer<String> logProducer = new FlinkLtsLogProducer<>(new
SimpleLogSerializer(), configProps);
 stream.addSink(logProducer);
 env.execute("flink log producer");
 }

 public static class EventsGenerator implements SourceFunction<String> {
 private volatile boolean running = true;
 @Override
 public void run(SourceContext<String> ctx) throws Exception {
 long seq = 0;
 while (running) {

```

```
 Thread.sleep(10);
 ctx.collect((seq++) + "-" + RandomStringUtils.randomAlphabetic(12));
 }
}
@Override
public void cancel() {
 running = false;
}
}
```

2. 初始化，Producer初始化主要是初始化配置参数Properties，这一步和Consumer类似，Producer有一些定制的参数，一般情况下使用默认值即可。

## DynamicLtsTableFactory 支持 SQL 作业

LtsDynamicSource和LtsDynamicSink table作业，支持LTS日志直接接入flink，支持SQL语法，示例如下：

```
public static void main(String[] args) throws Exception {
 StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
 EnvironmentSettings environmentSettings =
 EnvironmentSettings.newInstance().useBlinkPlanner().inStreamingMode().build();
 StreamTableEnvironment tableEnv = StreamTableEnvironment.create(env, environmentSettings);
 // enable checkpointing
 Configuration configuration = tableEnv.getConfig().getConfiguration();
 configuration.set(
 ExecutionCheckpointingOptions.CHECKPOINTING_MODE, CheckpointingMode.EXACTLY_ONCE);
 configuration.set(
 ExecutionCheckpointingOptions.CHECKPOINTING_INTERVAL, Duration.ofSeconds(10));

 tableEnv.executeSql("CREATE TABLE source (" +
 " collectTime varchar, " +
 " lineNum varchar, " +
 " podName varchar, " +
 " pathFile varchar, " +
 " category varchar " +
 ") " +
 " with (" +
 // connector 表类型 固定值lts
 "'connector' = 'lts', " +
 // LTS 日志服务所属region
 "'regionName' = 'cn-north-7', " +
 // LTS 日志租户项目ID
 "'projectId' = '*****', " +
 // 注意：认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险, 建议在配置文件或者环境变量中密文存放, 使用时解密, 确保安全
 // LTS 租户AK
 "'accessKey' = '*****', " +
 // LTS 租户SK
 "'accessSecret' = '*****', " +
 // LTS 日志组ID
 "'logGroupId' = '*****', " +
 // LTS 日志流ID
 "'logStreamId' = '*****', " +
 // LTS 日志流的消费组
 "'consumerGroup' = '*****', " +
 // LTS 日志消费起始位置
 "'startTime' = '1689836602157000000', " +
 // LTS 原始日志是否做JSON解析
 "'jsonParse' = 'false' " +
 ")");
 tableEnv.executeSql("CREATE TABLE print_sink (" +
 " collectTime varchar, " +
 " lineNum varchar, " +
 " podName varchar, " +
 " pathFile varchar, " +
 " category varchar " +
 ")");
}
```

```
 ") " +
 " with (" +
 // connector 表类型 固定值lts
 "'connector' = 'lts', " +
 // LTS 日志服务所属region
 "'regionName' = 'cn-north-7', " +
 // LTS 日志租户项目ID
 "'projectId' = '2a473356cca5487f8373be891bffc1cf', " +
 // LTS 租户AK
 "'accessKey' = 'DADYWPUP8JMUUV3UGPEI9', " +
 // LTS 租户SK
 "'accessSecret' = 'jUtvcc0oIcGZGoAUvtlSi8Oz6sZdFI2ZqFKBGUZ', " +
 // LTS 日志组ID
 "'logGroupId' = 'e83e94db-2e29-49c9-ae15-d3a9f4c3ea1b', " +
 // LTS 日志流ID
 "'logStreamId' = '0a423cfc-dbf8-4cf3-8fb9-f5cf95fa1298' " +
 ")");
 tableEnv.executeSql("insert into print_sink select * from source ");
}
```

### 10.6.6 LTS-Flink-Connector 参数说明

云日志服务可以帮助您快捷地完成数据采集、消费、投递以及查询分析，提升运维和运营效率，建立海量日志处理能力。

LTS-Flink-Connector工具支持以下[表10-13](#)。

表 10-13 功能说明

| 功能             | 说明                     |
|----------------|------------------------|
| 支持类型           | 源表和结果表。                |
| 运行模式           | 仅支持流模式。                |
| API种类          | SQL。                   |
| 是否支持更新或删除结果表数据 | 不支持更新和删除结果表数据，只支持插入数据。 |

### 使用限制

- 云日志服务提供的lts-flink-connector工具版本在1.0.4及之后支持flink1.15版本，1.0.4之前的版本支持flink1.12版本。
- LTS服务不支持作为维表。
- lts-flink-connector仅保证At-Least-Once语义。
- lts-flink-connector source和sink日志不能为同一个日志流。
- flink消费保证最终一致性，即您可以获取到这条日志流的全部内容，但由于时间为服务端时间，所以在获取日志流的过程中，可能导致获取到的日志数量跟LTS页面查询的日志数量不一致。

### 语法结构

认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
CREATE TABLE source (
 field1 INT,
```

```
field2 INT,
field3 VARCHAR

)with (
 'connector' = 'lts',
 'regionName' = '<yourRegionName>',
 'projectId' = '<yourProjectId>',
 'accessKey' = '<yourAk>',
 'accessSecret' = '<yourSk>',
 'logGroupId' = '<yourLogGroupId>',
 'logStreamId' = '<yourLogStreamId>',
 'consumerGroup' = '<yourConsumerGroup>',
 'startTime' = '<consumerStartTime>'
);
```

WITH 参数

- 通用参数

表 10-14 通用参数说明

| 参数名称          | 描述                           | 类型     | 是否必填 | 默认值             |
|---------------|------------------------------|--------|------|-----------------|
| connector     | 表类型。                         | String | 必填   | lts             |
| regionName    | 云日志服务的区域                     | String | 必填   | -               |
| projectId     | 华为云账号的项目ID<br>( project id ) | String | 必填   | -               |
| logGroupId    | LTS的日志组ID                    | String | 必填   | -               |
| logStreamId   | LTS的日志流ID                    | String | 必填   | -               |
| logGroupName  | LTS的日志组名称                    | String | 选填   | 日志组ID和名称不能同时为空。 |
| logStreamName | LTS的日志流名称                    | String | 选填   | 日志流ID和名称不能同时为空。 |
| accessKey     | 华为云账号的AK                     | String | 必填   | -               |
| accessSecret  | 华为云账号的SK                     | String | 必填   | -               |
| consumerGroup | LTS日志流对应的消费组名称               | String | 必填   | -               |

| 参数名称            | 描述                                                                                                                                                                                         | 类型      | 是否必填 | 默认值   |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|------|-------|
| startTime       | 消费开始时间：ns时间戳（指定时间消费）、begin_cursor（从头开始消费）、end_cursor（从最新数据开始消费）                                                                                                                            | String  | 必填   | -     |
| stopTime        | ns时间戳（消费结束时间）                                                                                                                                                                              | Long    | 选填   | -     |
| jsonParse       | 是否对原始日志做json解析                                                                                                                                                                             | Boolean | 选填   | false |
| enableLocalTest | 是否开启跨云消费日志（flink-connector版本最低必须为1.0.7支持此功能）。此功能仅适用于开发调测，对于消费性能不做保障。 <ul style="list-style-type: none"><li>选true时，开启跨云后用户能通过公网（华东-上海一）访问。</li><li>选false时，关闭跨云后用户是通过华为云当前区域主机访问。</li></ul> | boolean | 选填   | false |
| historyQuery    | 是否消费历史数据（flink-connector版本最低必须为1.0.7支持此功能）。 <ul style="list-style-type: none"><li>选true时，可以消费开启日志流引擎之前的数据。</li><li>选false时，只能消费当开启了日志流引擎之后的数据。</li></ul>                                   | boolean | 选填   | false |
| proxyp          | 使用自定义ip端口上报日志（flink-connector版本最低必须为1.1.0支持此功能）。                                                                                                                                           | String  | 选填   | -     |

| 参数名称      | 描述                                                                             | 类型     | 是否必填 | 默认值                                                |
|-----------|--------------------------------------------------------------------------------|--------|------|----------------------------------------------------|
| timeField | 指定字段名，该字段的值会覆盖__time__属性字段的值，表示日志写入时间。<br>( flink-connector版本最低必须为1.1.3支持此功能 ) | String | 选填   | 该参数值是表中已存在的字段之一，且字段必须为ms时间戳的INT类型。如果未指定，则默认填充当前时间。 |

类型映射

表 10-15 类型映射

| Flink字段类型 | LTS字段类型 |
|-----------|---------|
| VARCHAR   | STRING  |

代码示例

认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

```
CREATE TABLE source (
 filed1 varchar,
 filed2 varchar,
 filed3 varchar
)with (
 'connector' = 'lts',
 'regionName' = 'cn-north-4',
 'projectId' = '{projectId}',
 'accessKey' = '{ak}',
 'accessSecret' = '{sk}',
 'logGroupId' = '{groupId}',
 'logStreamId' = '{streamId}',
```

```
'consumerGroup' = '{consumerGroup}',
'startTime' = '1689836602157000000',
'jsonParse' = 'false'
);

CREATE TABLE print_sink (
 filed1 varchar,
 filed2 varchar,
 filed3 varchar
)with (
 'connector' = 'lts',
 'regionName' = 'cn-north-4',
 'projectId' = '{projectId}',
 'accessKey' = '{ak}',
 'accessSecret' = '{sk}',
 'logGroupId' = '{groupId}',
 'logStreamId' = '{streamId}'
);

insert into print_sink select * from source;
```

# 11 LTS 配置中心管理

## 11.1 设置 LTS 日志采集配额和使用量预警

### 配置超额采集

当日志超过每月免费赠送的额度（500M）时，超过的部分将按需收费。如果每月免费赠送的额度已经可以满足您的使用需求，超过后希望暂停日志收集，可以在配置中心进行设置。

- 该开关默认为开启状态，开启后表示当日志超过免费赠送的额度（500M）时，继续采集日志，超过的部分按需收费。
- 云日志服务的计费依据为日志使用量，包括日志读写、日志索引和日志存储。超过免费额度后，如果关闭日志采集开关，将无法再进行日志读写和索引，同时也不再产生日志读写和索引费用。关闭日志采集开关后，超额部分的日志继续存储在LTS，LTS收取日志存储费用，系统将根据配置的日志存储时间老化日志，日志老化后将不再产生任何费用。
- 云日志服务与应用运维管理的日志采集开关为同步状态，即如果您在应用运维管理服务关闭了“超额继续采集日志”开关，则云日志服务的开关也同步关闭。

**步骤1** 登录[云日志服务控制台](#)。

**步骤2** 左侧导航选择“配置中心”，进入“配额设置”页面。

**步骤3** 选择关闭“超额继续采集日志”。

关闭后表示当日志超过每月免费赠送的额度(500M)时，将暂停采集日志。支持在日志总览页查看当前资源使用情况，详情请参考[查看日志管理](#)。

**步骤4** 开启“可观测For LTS”，推荐开启，开启后LTS的日志资源统计数据将投递到lts-system/lts-resource-traffic，您可以查看分钟级用量明细、配置自定义告警策略。

#### 说明

“可观测For LTS”功能仅支持华北-北京四、华东-上海一、华南-广州、亚太-新加坡区域的白名单用户使用，如有需要，请[提交工单](#)，其他区域暂不支持申请开通。

----结束



配置日志资源使用量预警

开启自定义日志资源使用量预警开关后，系统将自动为您创建一条告警规则（日志资源使用量预警）。当日志使用量超过当前配置的自定义日志资源使用量额度时，系统会发送告警通知。日志使用量包括日志读写流量、索引流量和标准存储量。

**步骤1** 在“配额设置”页签，单击“自定义日志资源使用量预警”开关。

开启后，云日志服务首页的[资源统计](#)数据将投递到自动创建的日志流下（lts-system/lts-resource-statistics），可以通过日志告警功能实现自定义日志资源使用量预警。

- 参数为“或”关系，当任意一个参数满足条件时，告警列表会产生一条告警。
- 该开关默认为关闭状态，当开启后表示当日志资源使用量超过1024GB时，会触发告警规则。
- 开启该开关后，默认会自动创建日志组/日志流（lts-system/lts-resource-statistics）和告警规则（log-statistics-alarm）；关闭该开关，则会自动删除告警规则（log-statistics-alarm）。
- 日志资源使用量会每小时统计一次。
- 系统自动创建一条告警规则（日志资源使用量预警），SQL告警语句为：select write\_traffic,index\_traffic,storage，条件表达为：write\_traffic > 1024E12 || index\_traffic > 1024E12 || storage > 1024E13，如果您希望产生的告警以短信、邮件等方式通知您，您可以修改告警规则配置发送通知。
- 如果告警规则不存在，重新开启自定义日志资源使用量预警开关，系统会默认创建规则。



表 11-1 参数配置说明表

| 参数          | 说明                                                                                                                          |
|-------------|-----------------------------------------------------------------------------------------------------------------------------|
| 最近1小时日志读写流量 | 设置最近1小时内云日志服务的日志读写流量额度。默认值为1024GB，单位为GB，只能输入数字和小数点，保留4位小数，且不能以小数点开头和结尾，最小值为0，最大值输入长度不能超过10个字符。<br>当日志读写流量超过设置额度时，会触发日志告警功能。 |
| 最近1小时日志索引流量 | 设置最近1小时内云日志服务的日志索引流量额度。默认值为1024GB，单位为GB，只能输入数字和小数点，保留4位小数，且不能以小数点开头和结尾，最小值为0，最大值输入长度不能超过10个字符。<br>当日志索引流量超过设置额度时，会触发日志告警功能。 |

| 参数        | 说明                                                                                                                          |
|-----------|-----------------------------------------------------------------------------------------------------------------------------|
| 日志最新标准存储量 | 设置最近1小时日志的最新标准存储量额度。默认值为1024GB，单位为GB，只能输入数字和小数点，保留4位小数，且不能以小数点开头和结尾，最小值为0，最大值输入长度不能超过10个字符。<br>当日志最新标准存储量超过设置额度时，会触发日志告警功能。 |

----结束

## 11.2 设置 LTS 日志内容分词

通过配置分词可将日志内容按照分词符切分为多个单词，在日志搜索时可使用切分后的单词进行搜索。初次使用时，LTS已默认进行了分词配置，默认配置的分词符为：

```
, ";=() []{}@&<>/:\?\\n\\t\\r
```

若默认分词符不能满足您的需求时，可按照如下操作进行自定义配置。

### 注意事项

- 分词配置只会对配置时间点以后生成的日志生效，之前的日志按照之前配置的分词符进行处理。
- 在“分词配置”页签设置分词符后会对当前Region区域内的所有日志流生效，若需要单独配置单个日志流的分词符，请参考[创建LTS日志索引](#)。

### 配置分词

**步骤1** 登录[云日志服务控制台](#)。

**步骤2** 在左侧导航栏中选择“配置中心”，选择“分词配置”页签。

**步骤3** 配置分词。

LTS提供了如下两种配置分词的方法。若同时使用了这两种配置方法，则分词符取并集。

- 自定义分词符：单击“编辑”，在文本框中自定义输入需要的分词符。
- 特殊分词符：单击“编辑 > 添加特殊分词符”，参考[ASCII码对照表](#)输入ASCII值。

**步骤4** 预览分词效果。

在文本框中输入待预览的日志内容，单击“预览”。

**步骤5** 预览确认配置无误后，单击“保存”。

单击“重置”，可恢复到系统默认分词配置。

----结束

ASCII 码对照表

表 11-2 ASCII 码对照表

| AS<br>CII<br>值 | 控制字符                   | ASC<br>II值 | 控制字符 | AS<br>CII<br>值 | 控制字符 | AS<br>CII<br>值 | 控制字符 |
|----------------|------------------------|------------|------|----------------|------|----------------|------|
| 0              | NUL（空字<br>符）           | 32         | 空格   | 64             | @    | 96             | `    |
| 1              | SOH（标题开<br>始）          | 33         | !    | 65             | A    | 97             | a    |
| 2              | STX（正文开<br>始）          | 34         | "    | 66             | B    | 98             | b    |
| 3              | ETX（正文结<br>束）          | 35         | #    | 67             | C    | 99             | c    |
| 4              | EOT（传输结<br>束）          | 36         | \$   | 68             | D    | 100            | d    |
| 5              | ENQ（询问字<br>符）          | 37         | %    | 69             | E    | 101            | e    |
| 6              | ACK（确认回<br>应）          | 38         | &    | 70             | F    | 102            | f    |
| 7              | BEL（响铃）                | 39         | '    | 71             | G    | 103            | g    |
| 8              | BS（退格）                 | 40         | (    | 72             | H    | 104            | h    |
| 9              | HT（水平定位<br>符号，制表<br>符） | 41         | )    | 73             | I    | 105            | i    |
| 10             | LF（换行）                 | 42         | *    | 74             | J    | 106            | j    |
| 11             | VT（垂直定位<br>符号）         | 43         | +    | 75             | K    | 107            | k    |
| 12             | FF（换页键）                | 44         | ,    | 76             | L    | 108            | l    |
| 13             | CR（归位键）                | 45         | -    | 77             | M    | 109            | m    |
| 14             | SO（取消变<br>换）           | 46         | .    | 78             | N    | 110            | n    |
| 15             | SI（启用变<br>换）           | 47         | /    | 79             | O    | 111            | o    |
| 16             | DLE（跳出数<br>据通讯）        | 48         | 0    | 80             | P    | 112            | p    |
| 17             | DC1（设备控<br>制1）         | 49         | 1    | 81             | Q    | 113            | q    |

| AS CII 值 | 控制字符         | ASC II 值 | 控制字符 | AS CII 值 | 控制字符 | AS CII 值 | 控制字符     |
|----------|--------------|----------|------|----------|------|----------|----------|
| 18       | DC2 (设备控制2)  | 50       | 2    | 82       | R    | 114      | r        |
| 19       | DC3 (设备控制3)  | 51       | 3    | 83       | S    | 115      | s        |
| 20       | DC4 (设备控制4)  | 52       | 4    | 84       | T    | 116      | t        |
| 21       | NAK (确认失败回应) | 53       | 5    | 85       | U    | 117      | u        |
| 22       | SYN (同步用暂停)  | 54       | 6    | 86       | V    | 118      | v        |
| 23       | ETB (区块传输结束) | 55       | 7    | 87       | W    | 119      | w        |
| 24       | CAN (取消)     | 56       | 8    | 88       | X    | 120      | x        |
| 25       | EM (连接介质中断)  | 57       | 9    | 89       | Y    | 121      | y        |
| 26       | SUB (替换)     | 58       | :    | 90       | Z    | 122      | z        |
| 27       | ESC (跳出)     | 59       | ;    | 91       | [    | 123      | {        |
| 28       | FS (文件分隔符)   | 60       | <    | 92       | \    | 124      |          |
| 29       | GS (组群分隔符)   | 61       | =    | 93       | ]    | 125      | }        |
| 30       | RS (记录分隔符)   | 62       | >    | 94       | ^    | 126      | ~        |
| 31       | US (单元分隔符)   | 63       | ?    | 95       | _    | 127      | DEL (删除) |

### 11.3 设置 ICAgent 日志采集开关

在“ICAgent采集开关”页签，支持设置ICAgent采集开关（控制ICAgent是否对日志数据进行采集）、采集Syslog日志到AOM、采集容器标准输出到AOM、ICAgent诊断开关。

#### 设置采集开关

- 步骤1 登录[云日志服务控制台](#)。
- 步骤2 左侧导航选择“配置中心”，单击“ICAgent采集开关”。

**步骤3** ICAgent采集开关是用来控制ICAgent是否对日志数据进行采集。

- “ICAgent采集开关”默认打开，当您不需要采集日志时，通过关闭“ICAgent采集开关”停止日志采集，以减少资源占用。
- “ICAgent采集开关”关闭后，ICAgent会停止采集日志，且在应用运维管理AOM控制台的“日志采集开关”也会同步关闭。

**步骤4** 采集Syslog日志到AOM用来控制ICAgent是否采集Syslog日志到AOM1.0。开关关闭后，ICAgent将不会采集Syslog日志到AOM1.0，此功能仅支持5.12.182以上版本的ICAgent。

**步骤5** 采集容器标准输出到AOM。选择CCE集群，开启或关闭应用到该集群。开关关闭后，ICAgent将不会采集标准输出日志到AOM，此功能仅支持5.12.133以上版本的ICAgent。建议使用[云容器引擎CCE应用日志接入LTS](#)直接采集容器标准输出到LTS，不推荐采集到AOM。

**步骤6** ICAgent诊断开关用来控制是否开启采集诊断功能。开关开启后ICAgent运行日志将上报到日志组/日志流lts-system/lts-icagent-statistics，然后您可以使用采集诊断功能查看ICAgent运行状态，及时发现异常情况，此功能仅支持5.12.196及以上版本的ICAgent。详细请参考[采集诊断仪表盘模板](#)。

----结束

# 12 查看 LTS 审计事件

## 概述

云审计服务（Cloud Trace Service，CTS）可以记录LTS相关的操作事件，便于日后的查询、审计和回溯。

开通了云审计服务后，系统开始记录LTS资源的操作。云审计服务管理控制台保存最近7天的操作记录。

## 开通云审计服务

云审计服务的开通请参见[开通云审计服务](#)。

开通云审计服务后，如果需要查看LTS相关操作事件，请参见[查询审计事件](#)。

## 云审计支持的 LTS 操作列表

表 12-1 云审计服务支持的 LTS 操作列表

| 操作名称       | 资源类型           | 事件名称                |
|------------|----------------|---------------------|
| 删除消费组      | consumerGroup  | deleteConsumerGroup |
| 创建消费组      | consumerGroup  | createConsumerGroup |
| 创建日志组      | group          | createLogGroup      |
| 修改日志组      | group          | updateLogGroup      |
| 删除日志组      | group          | deleteLogGroup      |
| 创建日志流      | topic          | createLogStream     |
| 修改日志流      | topic          | updateLogStream     |
| 删除日志流      | topic          | deleteLogStream     |
| 删除日志收藏     | filter         | deleteFavorite      |
| 删除日志桶      | logPailSetting | deleteLogPail       |
| 添加日志转储至OBS | als            | addLogPailToOBS     |

| 操作名称          | 资源类型                 | 事件名称                        |
|---------------|----------------------|-----------------------------|
| 修改日志转储至OBS    | als                  | updateLogPailToOBS          |
| 删除日志转储至OBS    | als                  | deleteLogPailToOBS          |
| 批量启动暂停周期性日志转储 | als                  | batchActionLogPailToOBS     |
| 创建转储          | transfer             | createLogTransfer           |
| 修改转储          | transfer             | updateLogTransfer           |
| 删除转储          | transfer             | deleteLogTransfer           |
| 创建快速查询        | searchCriteria       | createLogSearchCriteria     |
| 修改快速查询        | searchCriteria       | updateLogSearchCriteria     |
| 删除快速查询        | searchCriteria       | deleteLogSearchCriteria     |
| 新增采集路径        | LogAgent             | createLogAgent              |
| 修改采集路径        | LogAgent             | updateLogAgent              |
| 删除采集路径        | LogAgent             | deleteLogAgent              |
| 创建跨账号日志接入     | ltsLogAccessConfig   | CreateLtsLogAccessConfig    |
| 创建日志接入        | ltsLogAccessConfig   | createAccessConfig          |
| 修改日志接入        | ltsLogAccessConfig   | updateAccessConfig          |
| 删除日志接入        | ltsLogAccessConfig   | deleteAccessConfig          |
| 创建通知模板        | NotificationTemplate | createNotificationTemplate  |
| 修改通知模板        | NotificationTemplate | updateNotificationTemplate  |
| 删除通知模板        | NotificationTemplate | deleteNotificationTemplate  |
| 创建结构化模板       | structLogConfig      | createLogStreamStructConfig |
| 修改结构化模板       | structLogConfig      | updateLogStreamStructConfig |
| 删除结构化模板       | structLogConfig      | deleteLogStreamStructConfig |
| 添加或更新索引配置     | filedIndex           | createOrUpdateFiledIndex    |
| 创建dli转储       | dliDump              | createDliDump               |
| 删除dli转储       | dliDump              | deleteDliDump               |

| 操作名称      | 资源类型               | 事件名称                     |
|-----------|--------------------|--------------------------|
| 修改主机组     | hostGroup          | updateHostGroup          |
| 创建主机组     | hostGroup          | createHostGroup          |
| 删除主机组     | hostGroup          | deleteHostGroup          |
| 创建快速分析    | wordFreqConfig     | updateWordFreqConfig     |
| 版面刷新配置    | wordFreqConfig     | updateLayoutConfig       |
| 创建预汇聚算子   | aggrMeta           | createAggrMeta           |
| 删除快速分析聚合器 | aggrMeta           | deleteAggrMeta           |
| 存储日志路径配置  | path               | addLogPath               |
| 添加统计规则    | rule               | addRuleStatistics        |
| 修改统计规则    | rule               | updateRuleStatistics     |
| 删除统计规则    | rule               | deleteRuleStatistics     |
| 创建结构化     | structurization    | addStructurization       |
| 删除结构化     | structurization    | deleteStructurization    |
| 添加kafka实例 | dmsKafka           | registerKafkaInfo        |
| 修改kafka实例 | dmsKafka           | updateKafkaInfo          |
| 删除kafka实例 | dmsKafka           | deleteKafkaInfo          |
| 创建dis转储   | transfer           | createDisTransfer        |
| 修改dis转储   | transfer           | updateDisTransfer        |
| 删除dis转储   | transfer           | deleteDisTransfer        |
| 创建kafka转储 | kafkaTransfer      | createKafkaTransfer      |
| 修改kafka转储 | kafkaTransfer      | updateKafkaTransfer      |
| 删除kafka转储 | kafkaTransfer      | deleteKafkaTransfer      |
| 创建仪表盘过滤器  | logDashboardFilter | createLogDashboardFilter |
| 修改仪表盘过滤器  | logDashboardFilter | updateLogDashboardFilter |
| 删除仪表盘过滤器  | logDashboardFilter | deleteLogDashboardFilter |
| 创建历史查询    | historicalQuery    | createHistoricalQuery    |
| 创建图表      | logChart           | createLogChart           |
| 修改图表      | logChart           | updateLogChart           |



| 操作名称       | 资源类型                         | 事件名称                          |
|------------|------------------------------|-------------------------------|
| 删除图表       | logChart                     | deleteLogChart                |
| 创建仪表盘      | logDashboard                 | createLogDashboard            |
| 修改仪表盘      | logDashboard                 | updateLogDashboard            |
| 删除仪表盘      | logDashboard                 | deleteLogDashboard            |
| 创建仪表盘分组    | logDashboard                 | createLogDashboardGroup       |
| 开启日志超额采集开关 | LogCollectionSwitchOperation | LogCollectionSwitchOperation  |
| 关闭日志超额采集开关 | LogCollectionSwitchOperation | LogCollectionSwitchOperation  |
| 批量查询告警规则   | alarm                        | batchQueryAlarmList           |
| 修改告警规则状态   | KeywordsAlarm                | updateKeywordsAlarmRuleStatus |
| 创建关键词告警    | KeywordsAlarm                | createKeywordsAlarmRule       |
| 删除关键词告警    | KeywordsAlarm                | deleteKeywordsAlarmRule       |
| 修改关键词告警    | KeywordsAlarm                | updateKeywordsAlarmRule       |
| 创建ELB日志桶   | elbPailType                  | createElbPail                 |
| 修改ELB日志桶   | elbPailType                  | updateElbPail                 |
| 删除ELB日志桶   | elbPailType                  | deleteElbPail                 |
| 添加日志路径采集规则 | logPathCollectionType        | createLogPathCollection       |
| 修改日志路径采集规则 | logPathCollectionType        | updateLogPathCollection       |
| 删除日志路径采集规则 | logPathCollectionType        | deleteLogPathCollection       |
| 清理Redis缓存  | cleanTenantResourceType      | deleteCleanTenantResource     |
| 获取obs预览    | ltsLogAccessConfig           | PreviewObsAccessConfig        |
| 修改主机组      | ltsHostGroup                 | UpdateHostGroup               |
| 删除主机组      | ltsHostGroup                 | DeleteHostGroup               |
| 主机组删除主机    | ltsHostGroup                 | HostGroupDeleteHosts          |
| 主机组添加主机    | ltsHostGroup                 | HostGroupAddHosts             |
| 创建主机组      | ltsHostGroup                 | CreateHostGroup               |