

云日志服务

# 最佳实践

文档版本 01  
发布日期 2024-09-29



版权所有 © 华为云计算技术有限公司 2024。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

<b>1 LTS 最佳实践总览</b>	<b>1</b>
<b>2 日志接入</b>	<b>3</b>
2.1 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	3
2.2 采集第三方云厂商、互联网数据中心、华为云其他 Region 的 Kubernetes 日志到 LTS	9
2.3 采集 Syslog 汇聚服务器日志到 LTS	15
2.4 将自建 ELK 日志导入云日志服务 LTS	17
2.5 基于 VPN 打通网络实现 API 或 SDK 跨云上报日志到 LTS	20
2.6 使用 Flume 采集器上报日志到 LTS	23
2.7 通过 ECS 接入 LTS 采集 Zabbix 数据	32
<b>3 日志搜索与分析</b>	<b>34</b>
3.1 在 LTS 页面分析华为云 ELB 日志	34
3.2 通过 LTS 仪表盘可视化 ELB 日志分析结果	35
3.3 在 LTS 页面分析华为云 WAF 日志	37
3.4 在 LTS 页面分析 Log4j 格式的应用运行日志	38
3.5 将 LTS 日志查询页面嵌入用户自建系统	39
<b>4 日志转储</b>	<b>51</b>
4.1 批量修改 LTS 日志文件转储时区	51
<b>5 日志计费</b>	<b>56</b>
5.1 通过日志流标签统计不同部门在 LTS 的费用开销	56
<b>6 日志加工（邀测）</b>	<b>59</b>
6.1 使用 DSL 加工函数清洗 LTS 日志数据	59
6.1.1 场景一：过滤 LTS 日志	59
6.1.2 场景二：使用 e_set 函数为日志空缺字段赋值	60
6.1.3 场景三：删除和重命名字段（e_drop_fields 函数和 e_rename 函数）	61
6.1.4 场景四：转换日志参数类型（v 函数、cn_int 函数和 dt_totimestamp 函数）	61
6.1.5 场景五：使用 default 传参为日志不存在的字段填充默认值	62
6.1.6 场景六：使用 e_if 函数或 e_switch 函数判断日志并增加字段	62
6.1.7 场景七：数据转化纳秒级的 Unix 时间戳	63
6.1.8 场景八：数据转化微秒级标准 ISO8601 时间戳	63
6.2 使用 DSL 加工函数进行事件判断	64
6.3 使用 DSL 加工函数处理日期时间	67

6.3.1 概念解释.....	67
6.3.2 数据类型转换和转换函数.....	68
6.3.3 日期时间对象和 Unix 时间戳的相互转换.....	70
6.3.4 日期时间字符串和 Unix 时间戳的相互转换.....	70
6.3.5 日期时间对象和日期时间字符串的相互转换.....	71
6.3.6 日期时间偏移.....	72
6.4 使用 DSL 加工函数对 LTS 日志数据脱敏.....	73
6.5 文本解析.....	76
6.5.1 解析 Nginx 日志.....	76
6.5.2 提取字符串动态键值对.....	78
6.6 数据富化.....	82
6.6.1 从 OBS 获取 csv 文件进行数据富化.....	82
6.6.2 使用 e_dict_map、e_search_dict_map 函数进行数据富化.....	83
6.6.3 构建字典与表格进行数据富化.....	85

# 1 LTS 最佳实践总览

以下罗列了云日志服务（LTS）相关的最佳实践。

表 1-1 最佳实践总览

分类	最佳实践	场景说明
日志接入	<a href="#">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</a>	本实践主要介绍将阿里云主机日志采集到华为云LTS的操作步骤，互联网数据中心和华为云上跨Region采集日志的操作方式与采集阿里云主机日志的方式类似。
日志接入	<a href="#">采集第三方云厂商、互联网数据中心、华为云其他Region的Kubernetes日志到LTS</a>	本实践主要介绍将阿里云Kubernetes日志采集到华为云LTS的操作步骤，互联网数据中心和华为云上跨Region采集日志的操作方式与采集阿里云主机日志的方式类似。
日志接入	<a href="#">采集Syslog汇聚服务器日志到LTS</a>	本实践主要介绍通过Syslog协议将日志上传到日志服务的操作步骤。您需要购买ECS作为Syslog汇聚服务器，Linux服务器默认自带Syslog，目前华为云主机默认未配置接收远程Syslog写入，需要手动开启。
日志接入	<a href="#">将自建ELK日志导入云日志服务LTS</a>	本实践主要介绍使用自定义Python脚本和LTS采集器ICAgent，协助用户将日志从Elasticsearch（简称ES）迁移到LTS中。
日志接入	<a href="#">基于VPN打通网络实现API或SDK跨云上报日志到LTS</a>	本实践主要介绍基于VPN快速打通网络，实现API或SDK跨云上报日志。
日志接入	<a href="#">使用Flume采集器上报日志到LTS</a>	本实践主要介绍使用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。

分类	最佳实践	场景说明
日志搜索与分析	<a href="#">在LTS页面分析华为云ELB日志</a>	本实践主要介绍将ELB日志接入LTS后，配置日志结构化后，即可进行日志搜索分析。
日志搜索与分析	<a href="#">通过LTS仪表盘可视化ELB日志分析结果</a>	本实践主要介绍将ELB日志接入LTS后，配置日志结构化后，支持使用仪表盘将日志可视化，可以更直观的分析日志数据。
日志搜索与分析	<a href="#">在LTS页面分析华为云WAF日志</a>	本实践主要介绍将WAF日志接入LTS后，配置日志结构化后，即可进行日志搜索分析。
日志搜索与分析	<a href="#">在LTS页面分析Log4j格式的应用运行日志</a>	Log4j是Apache的一个开源项目，通过使用Log4j工具，我们可以将日志输出并保存到日志文件中，开发或运维人员会基于该日志统计日志级别的数量和占比，或者通过运行日志统计业务数据。
日志搜索与分析	<a href="#">将LTS日志查询页面嵌入用户自建系统</a>	本实践主要介绍通过统一身份认证服务IAM的联邦代理机制实现用户自定义身份代理，再将LTS登录链接嵌入至客户自建系统实现无需在华为云官网登录就可在自建系统界面查询LTS日志。
日志转储	<a href="#">批量修改LTS日志文件转储时区</a>	本实践主要介绍通过Python脚本结合LTS API接口实现自定义的批量操作。
日志计费	<a href="#">通过日志流标签统计不同部门在LTS的费用开销</a>	本实践主要介绍为了统计企业内部不同部门在LTS的费用开销情况，您可以在LTS的日志流上添加标签用于识别不同的业务部门，LTS在上传话单给费用中心时会带上这些标签信息。

# 2 日志接入

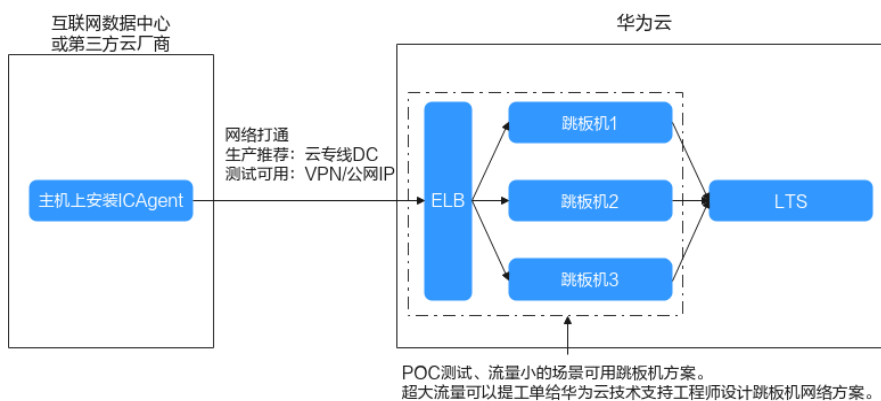
## 2.1 采集第三方云厂商、互联网数据中心、华为云其他 Region 云主机日志到 LTS

### 方案概述

云上用户经常会遇到多云或者跨Region采集日志的场景，典型场景有两种，分别是：

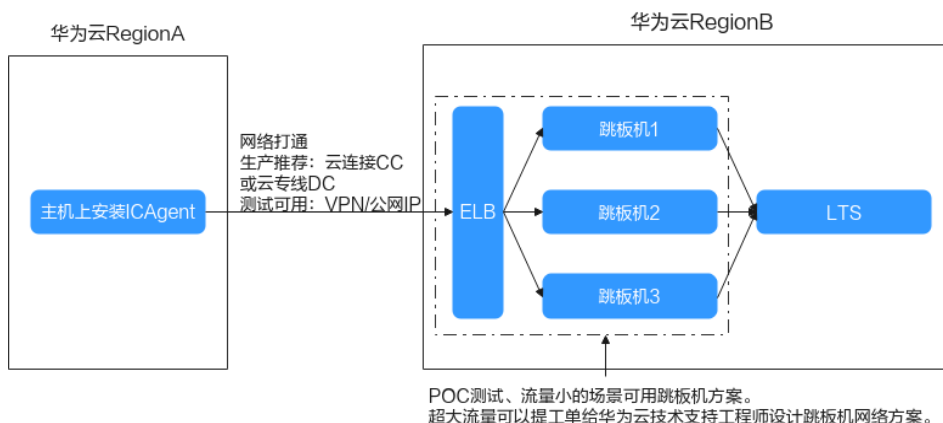
- 场景一：将互联网数据中心（Internet Data Center，以下简称IDC）或者第三方云厂商的日志采集到华为云LTS。

图 2-1 第三方云厂商日志采集



- 场景二：将华为云RegionA的日志采集到华为云RegionB的LTS。

图 2-2 跨 Region 日志采集



对于场景一和场景二，您需要先打通网络，再安装ICAgent，最后按照日志接入向导即将日志采集到LTS。

- **ICAgent:** ICAgent是华为云日志服务的日志采集器，通过在主机上安装ICAgent，您可以将日志采集到LTS。

#### 📖 说明

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。

- **网络互通:**
  - 场景一：自建IDC或者第三方云厂商与华为云之间网络互通的典型方式是云专线DC，如果没有专线，您可以尝试VPN/公网IP方式。
  - 场景二：华为云不同Region之间网络互通典型的方式是云连接CC/云专线DC，您也可以使用VPN/公网IP方式。
- **跳板机**
  - 安装在自建IDC/第三方云厂商/跨华为云Region的ICAgent无法直接访问华为云管理面上报日志的网段，需要配置跳板机进行数据转发；当您在进行POC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。对于大流量的日志场景，如果您希望在生产环境中去掉跳板机，请[提交工单](#)给华为云网络技术支持工程师帮您设计网络直通的方案。
  - 典型的跳板机配置是2vCPUs | 4GB，每台跳板机可以支持约30MB/s的流量转发，您可以根据自身的日志流量配置合理数量的跳板机，多台跳板机前面配置ELB进行流量分发。

本文将详细介绍将阿里云主机日志采集到华为云LTS的操作步骤，客户自建IDC和为云上跨Region采集日志的操作方式与采集阿里云主机日志的方式类似。

以下将阿里云-华北二-北京局点的日志采集到华为云华东-上海一局点的LTS服务，云主机的操作系统为Linux环境。



## 资源规划

表 2-1 资源规划

区域	资源	资源说明
华东-上海一	弹性云服务器 ECS	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。
	弹性负载均衡 ELB	<ul style="list-style-type: none"><li>• 购买弹性负载均衡 ELB时，选择与弹性云服务器 ECS相同的VPC。</li><li>• 新建弹性公网IP作为跳板机连接的IP。</li><li>• 根据业务量购买带宽，并进行适配。</li></ul>

## 在华为云华东-上海一购买 ELB 和多台云主机作为跳板机

**步骤1** 登录云服务控制台，购买弹性云服务器 ECS。

非华为云上的服务器安装ICAgent，需要先在华为云上购买一台弹性云服务器作为跳板机。

**步骤2** 购买弹性负载均衡 ELB，并添加TCP监听器和后端服务器组。

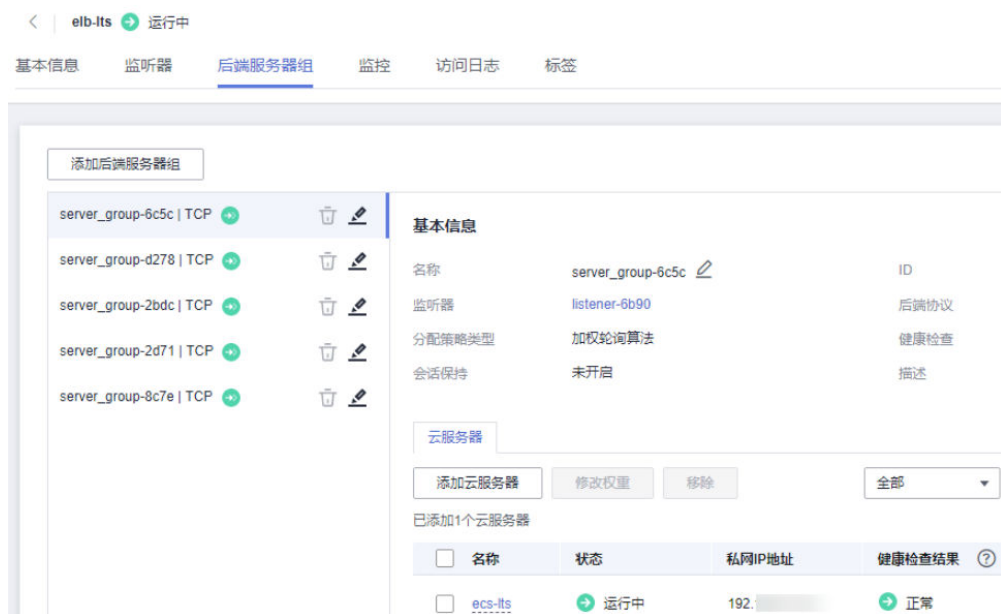
1. 添加监听器。分别为端口30200、30201、8149、8923、8102添加监听器，具体请参见[添加TCP监听器](#)。

图 2-3 添加监听器



2. 添加云服务器。给后端服务器组添加所有跳板机，具体请参见[添加后端服务器](#)。

图 2-4 添加云服务器



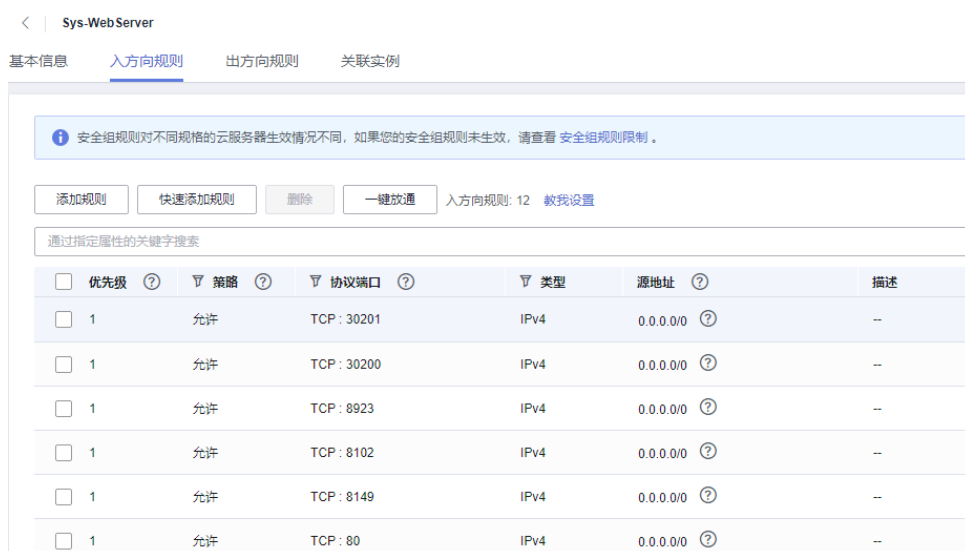
### 步骤3 配置跳板机安全组规则，并开通转发端口。

1. 修改跳板机ECS使用的安全组规则。

#### 说明

1. 在ECS详情页，单击安全组页签，进入安全组列表页。
2. 单击具体的安全组名，单击“更改安全组规则”，进入安全组详情页。
3. 在该安全组详情页，单击“入方向规则 > 添加规则”。将安全组的入方向端口8149、8102、8923、30200、30201、80开启，保证非华为云的VM到跳板机ECS的数据连通性。

图 2-5 修改安全组规则



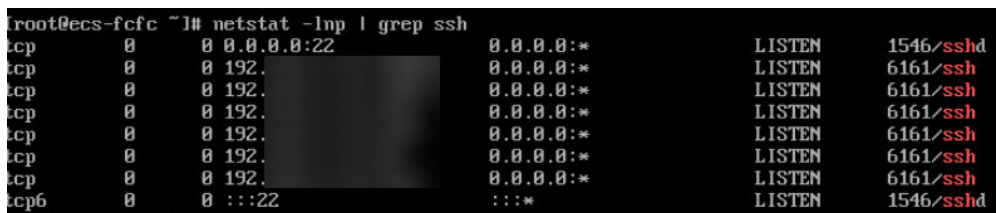
2. 在云日志服务控制台，选择“主机管理”，单击“安装ICAgent”，进入安装ICAgent详情页面。按下图配置，填入ECS私有IP，生成安装口令。

图 2-6 安装 ICAgent



- 复制命令以root用户登录跳板机，执行SSH Tunnel转发命令，根据命令提示输入root用户密码即可。
- 执行netstat -lnp | grep ssh命令查看对应端口是否被侦听，如果返回结果如下图所示，说明TCP端口已开通。

图 2-7 查看端口



📖 说明

- 在浏览器地址栏里输入“http://跳板机ECS的IP地址”。如果访问成功，说明安全组规则已经生效。
- 如果跳板机ECS掉电重启，请重新执行安装ICAgent页面生成的安装命令。如果您在生产环境中使用，请将上述端口转发命令配置为开机启动任务。

----结束

## 在阿里云主机上安装 ICAgent

**步骤1** 获取AK/SK，请参考[获取华为账号的AK/SK](#)。

**步骤2** 输入跳板机连接IP（注意需要填入ECS弹性公网IP），生成ICAgent安装命令。

### 📖 说明

- 请确保替换正确的AK/SK，否则将无法安装ICAgent。
- 跳板机连接IP：使用EIP方式连接，为跳板机弹性公网IP，使用云专线VPC对等连接方式，为跳板机VPC内网IP。

**步骤3** 复制命令root用户登录阿里云ECS，执行ICAgent安装命令进行安装，当显示“ICAgent install success”时，表示安装成功。

图 2-8 查看 ICAgent 安装状态

```
root@i22zefw10m3j17h0lcu1a2c-0 com] http://[AGENT_IP]:8080/obs-ak-sk-3-mhuuaw1cloud.com/ICAgent-11m5/aw_agent_install.sh && #610W-57
root@i22zefw10m3j17h0lcu1a2c-0 com]
#####
% Total % Received % Xferd Average Speed Time Time Time Current
           Dload Upload Total Spent Left Speed
100 7400 100 7400 0 0 55922 0 0 55922 0 0 55922
start to install ICAGENT.
begin to download install package from icag.....myhuaweicloud.com.....
Download success.
start install package.
start install ICAGENT...
daemon
start
#####
Starting ICAGENT...
ICAGENT install success.
```

### 📖 说明

如果是华为云跨Region采集日志，例如：将华为云华东-上海一局点采集日志到华为云华南-广州局点，需要在华南-广州局点购买ELB和跳板机ECS，然后将ICAgent的安装命令在华东-上海一局点的跳板机ECS上执行。

**步骤4** 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状态为“运行”。

----结束

## 将日志接入 LTS

**步骤1** 登录华为云云日志服务控制台，在左侧导航栏中，选择“主机管理>新建主机组”，填写“主机组名称”并勾选主机，即可创建完成主机组。

**步骤2** 配置日志接入规则。具体操作请参考[ECS接入](#)。

----结束

## 查看日志流

在云日志服务（LTS）的日志管理页面，单击目标日志流名称进入详情页面，查看有日志即可证明阿里云主机日志已成功上报到LTS。

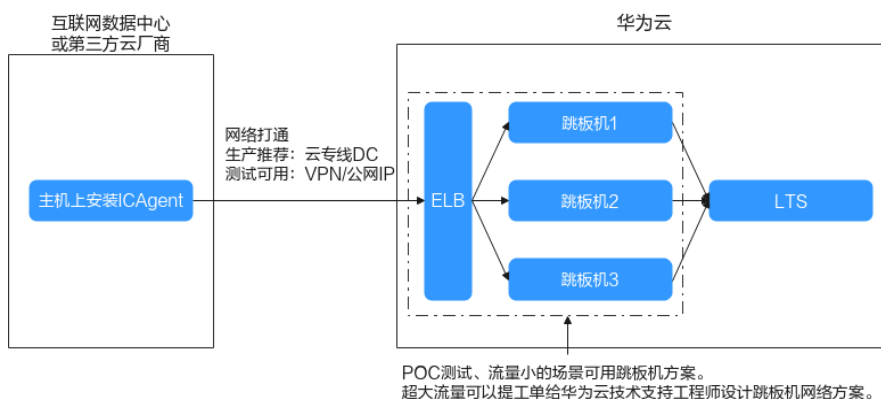
## 2.2 采集第三方云厂商、互联网数据中心、华为云其他 Region 的 Kubernetes 日志到 LTS

### 方案概述

云上用户经常会遇到多云或者跨Region采集Kubernetes日志场景，典型场景有两种，分别是：

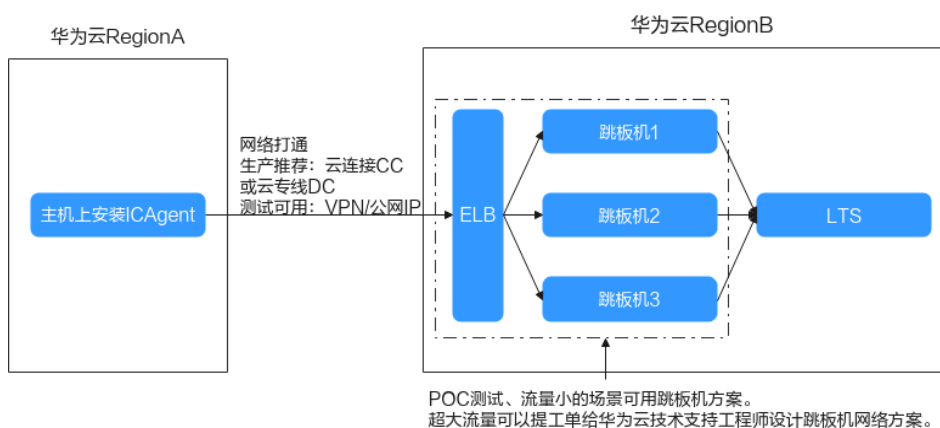
- 场景一：将互联网数据中心（Internet Data Center，以下简称IDC）或者第三方云厂商的日志采集到华为云LTS。

图 2-9 第三方云厂商日志采集



- 场景二：将华为云RegionA的日志采集到华为云RegionB的LTS。

图 2-10 跨 Region 日志采集



对于场景一和场景二，您需要先打通网络，再安装ICAgent，最后按照日志接入向导即可将日志采集到LTS。

- **ICAgent**：ICAgent是华为云日志服务的日志采集器，通过在主机上安装ICAgent，您可以将日志采集到LTS。

## 📖 说明

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。

- **网络互通：**
  - 场景一：自建IDC或者第三方云厂商与华为云之间网络互通的典型方式是云专线DC，如果没有专线，您可以尝试VPN/公网IP方式。
  - 场景二：华为云不同Region之间网络互通典型的方式是云连接CC/云专线DC，您也可以使用VPN/公网IP方式。
- **跳板机**
  - 安装在自建IDC/第三方云厂商/跨华为云Region的ICAgent无法直接访问华为云管理面上报日志的网段，需要配置跳板机进行数据转发；当您在进行POC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。对于大流量的日志场景，如果您希望在生产环境中去掉跳板机，请[提交工单](#)给华为云网络技术支持工程师帮您设计网络直通的方案。
  - 典型的跳板机配置是2vCPUs | 4GB，每台跳板机可以支持约30MB/s的流量转发，您可以根据自身的日志流量配置合理数量的跳板机，多台跳板机前面配置ELB进行流量分发。

本文将详细介绍将阿里云主机日志采集到华为云日志服务（LTS）的操作步骤，客户自建IDC和华为云上跨region采集日志的操作方式与采集阿里云主机日志的方式类似。

以下将阿里云-华北二北京局点的日志采集到华为云华东-上海一局点的LTS服务，云主机的操作系统为Linux环境。

## 资源规划

表 2-2 资源规划

区域	资源	资源说明
华东-上海一	弹性云服务器 ECS	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。
	弹性负载均衡 ELB	<ul style="list-style-type: none"><li>● 购买弹性负载均衡 ELB时，选择与弹性云服务器 ECS相同的VPC。</li><li>● 新建弹性公网IP作为跳板机连接的IP。</li><li>● 根据业务量购买带宽，并进行适配。</li></ul>

## 在华为云华东-上海一购买 ELB 和多台云主机作为跳板机

**步骤1** 登录云服务控制台，购买弹性云服务器 ECS。

非华为云上的服务器安装ICAgent，需要先在华为云上购买一台弹性云服务器作为跳板机。

**步骤2** 购买弹性负载均衡 ELB，并添加TCP监听器和后端服务器组。

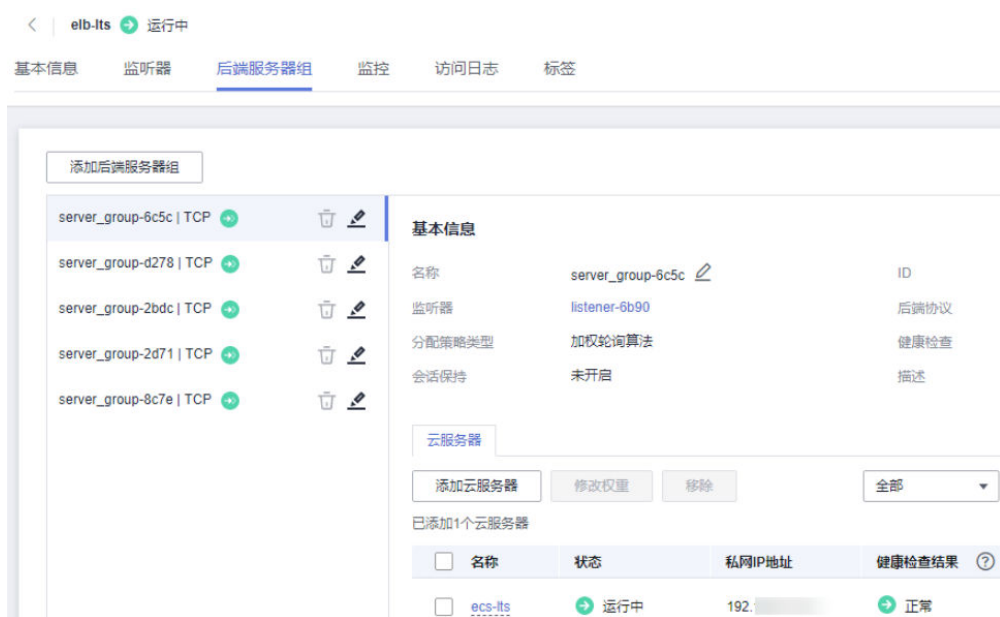
1. 添加监听器。分别为端口30200、30201、8149、8923、8102添加监听器，具体请参见[添加TCP监听器](#)。

图 2-11 添加监听器



2. 添加云服务器。给后端服务器组添加所有跳板机，具体请参见[添加后端服务器](#)。

图 2-12 添加云服务器



### 步骤3 配置跳板机安全组规则，并开通转发端口。

1. 修改跳板机ECS使用的安全组规则。

#### 📖 说明

1. 在ECS详情页，单击安全组页签，进入安全组列表页。
2. 单击具体的安全组名，单击“更改安全组规则”，进入安全组详情页。
3. 在该安全组详情页，单击“入方向规则 > 添加规则”。将安全组的入方向端口8149、8102、8923、30200、30201、80开启，保证非华为云的VM到跳板机ECS的数据连通性。

图 2-13 修改安全组规则



- 在云日志服务控制台, 选择“主机管理”, 单击“安装ICAgent”, 进入安装ICAgent详情页面。按下图配置, 填入ECS私有IP, 生成安装口令。

图 2-14 安装 ICAgent



- 复制命令以root用户登录跳板机, 执行SSH Tunnel转发命令, 根据命令提示输入root用户密码即可。



4. 执行 `netstat -lnp | grep ssh` 命令查看对应端口是否被侦听，如果返回结果如下图所示，说明TCP端口已开通。

图 2-15 查看端口

```
root@ecs-fcfc ~]# netstat -lnp | grep ssh
tcp        0      0 0.0.0.0:22          0.0.0.0:*          LISTEN     1546/sshd
tcp        0      0 0.0.0.0:22          0.0.0.0:*          LISTEN     1546/sshd
tcp        0      0 0.0.0.0:22          0.0.0.0:*          LISTEN     1546/sshd
tcp        0      0 0.0.0.0:22          0.0.0.0:*          LISTEN     1546/sshd
tcp        0      0 0.0.0.0:22          0.0.0.0:*          LISTEN     1546/sshd
tcp        0      0 0.0.0.0:22          0.0.0.0:*          LISTEN     1546/sshd
tcp6       0      0 :::22              :::*                LISTEN     1546/sshd
```

### 说明

- 在浏览器地址栏里输入“`http://跳板机ECS的IP地址`”。如果访问成功，说明安全组规则已经生效。
- 如果跳板机ECS掉电重启，请重新执行安装ICAgent页面生成的安装命令。如果您在生产环境中使用，请将上述端口转发命令配置为开机启动任务。

---结束

## 配置日志接入

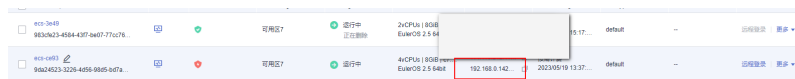
Kubernetes集群在一个节点上安装成功即可，不需要所有节点重复操作。

请提前获取AK/SK，请参考[获取华为账号的AK/SK](#)。

### 步骤1 配置跳板机。

1. 在弹性云服务器管理控制台找到已创建的机器，获取私有IP。

图 2-16 获取私有 IP



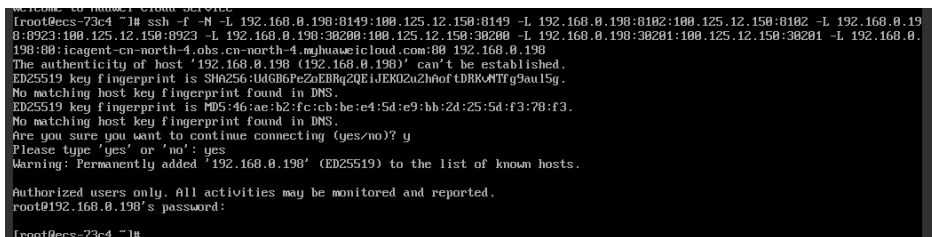
2. 在云日志服务LTS（LTS）控制台，选择“主机管理”，单击“安装ICAgent”，进入安装ICAgent详情页面。按下图配置，填入ECS私有IP，生成安装口令并复制。

图 2-17 安装 ICAgent



- 3. 登录跳板机 ECS，执行上一步的命令并根据提示输入节点密码，若无报错，表示安装成功。

图 2-18 执行生成的安装命令



- 4. 如上图所示，在文本框中输入跳板机的公网ip，关闭历史命令记录，复制命令并在跳板机ecs上执行，根据提示输入当前账号的AK、SK，当显示“ICAgent install success”时，表示安装成功。

**步骤2** 配置日志接入规则。具体操作请参考[自建K8s应用日志接入LTS](#)。

----结束

## 查看日志流

在云日志服务（LTS）的日志管理页面中，单击目标日志流名称进入详情页面，查看有日志即可证明阿里云 Kubernetes 日志已成功上报到 LTS。

## 2.3 采集 Syslog 汇聚服务器日志到 LTS

### 背景信息

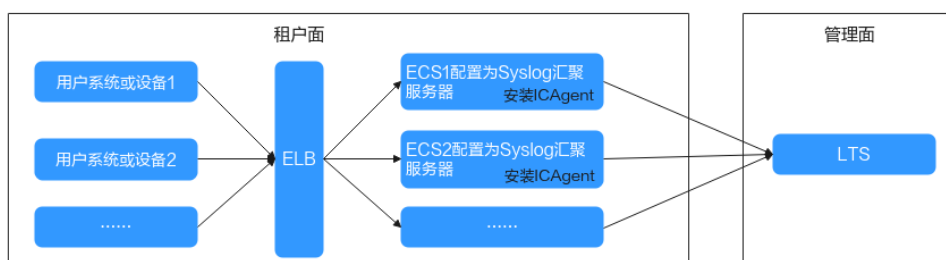
Syslog是网络上各种设备将日志收集到日志服务器的一种数据协议，它几乎被所有的网络设备支持，并且能够记录多种事件类型的日志消息，支持syslog的设备常见的有路由器、交换机、打印机等，甚至unix-like的服务器也可以支持产生syslog消息，用以记录用户的登录、防火墙事件、apache或者nginx access日志等。

Syslog主要是基于RFC5424和RFC3164定义相关格式规范，RFC3164协议是2001年发布的，RFC5424协议是2009年发布的升级版本。因为新版兼容旧版，且新版本解决了很多问题，因此推荐使用RFC5424协议。更多信息请参见[RFC5424](#)和[RFC3164](#)。

本文介绍通过Syslog协议将日志上传到日志服务的操作步骤。您需要购买ECS作为Syslog汇聚服务器，Linux服务器默认自带Rsyslog，目前华为云主机默认未配置接收远程Syslog写入，需要手动开启。

### 方案概述

图 2-19 方案流程图



- 你可以购买Linux云主机，配置为Syslog汇聚服务器，用于接收其他设备发送的日志数据；Syslog服务器默认接收日志大小为1024字节，超过会截断。
- 单台Syslog服务器处理日志能力为10MB/s，如果您的日志量较大，或者希望可靠性更高，可以购买多台ECS配置为Syslog服务器，并配置ELB负载均衡分发流量。
- 您需要在Syslog服务器上安装ICAgent，并配置日志采集规则，就可以将日志采集到LTS。

### 资源规划

购买两台ECS机器，一台ECS作为Syslog汇聚服务器，另一台用为业务ECS模拟客户系统或者设备发送日志。

### 购买弹性云服务器

**步骤1** 登录管理控制台，选择“计算 > 弹性云服务器 ECS”。

**步骤2** 购买一台弹性云服务器作为Syslog汇聚服务器。

#### 📖 说明

推荐CentOS 6.5 64bit及其以上版本的镜像，推荐规格为2vCPUs | 4GB。

**步骤3** 使用root用户登录Syslog服务器，安装ICAgent。

1. syslog服务器安全组出方向打开TCP协议的30200、30201、8149、8923、8102等端口，入方向需打开UDP的514端口作为syslog服务器默认监听端口。
2. 在云日志服务 LTS管理控制台，左侧导航栏选择主机管理，单击“安装ICAgent”，安装系统选择“Linux”，主机类型选择“区域内主机”，更改命令中的ak/sk字段为获取到的ak/sk。
3. 复制命令root用户登录syslog服务器，执行ICAgent安装命令进行安装，当显示“ICAgent install success”时，表示安装成功。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。

**步骤4** 打开Rsyslog服务器监听接收功能。

目前华为云主机rsyslog服务器默认未配置接收远程syslog写入，需要手动开启。

1. 已登录云主机。
2. 修改rsyslog配置文件。  
vi /etc/rsyslog.conf
3. 配置文件中添加以下内容，开启tcp udp远程接收。  
Provides UDP syslog reception  
\$ModLoad imudp  
\$UDPServerRun 514  
  
Provides TCP syslog reception  
\$ModLoad imtcp  
\$InputTCPServerRun 514
4. 保存后重启rsyslog服务器，单击云服务器“更多 > 重启”。
5. 执行以下任意一条命令显示正常代表服务运行正常。  
执行“service rsyslog status”命令检查rsyslog运行状态为running。

**图 2-20** 检查 rsyslog 运行状态

```
root@ecs-syslog-huod270223 ~# service rsyslog status
Redirecting to /bin/systemctl status rsyslog.service
● rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2023-09-08 15:55:17 CST; 2 days ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
   Main PID: 4162 (rsyslogd)
   CGroup: /system.slice/rsyslog.service
           └─4162 /usr/sbin/rsyslogd -n
```

执行“systemctl status rsyslog”命令检查rsyslog运行状态为running。

**图 2-21** rsyslog 运行状态

```
root@ecs-syslog-huod270223 ~# systemctl status rsyslog
● rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2023-09-08 15:55:17 CST; 2 days ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
   Main PID: 4162 (rsyslogd)
   CGroup: /system.slice/rsyslog.service
           └─4162 /usr/sbin/rsyslogd -n
```

执行“netstat -anp | grep 514”命令查看是否打开监听。

**图 2-22** 查看是否打开监听

```
root@ecs-e859 ~# netstat -anp | grep 514
tcp        0      0 0.0.0.0:514          0.0.0.0:*           LISTEN    988/rsyslogd
tcp6      0      0 0.0.0.0:514        :::*                 LISTEN    988/rsyslogd
udp        0      0 0.0.0.0:514        0.0.0.0:*           988/rsyslogd
udp6      0      0 :::514              :::*                 988/rsyslogd
```

**步骤5** 配置采集syslog日志。

1. 创建主机组，在云日志服务 LTS控制台左侧导航栏中选择“主机管理 > 新建主机组”，设计“主机组名称”，勾选主机即可。
2. 左侧导航栏中选择“日志接入 > 云主机ECS-文本日志”。
3. 选择日志流。
4. 选择主机组。
5. 采集配置，配置采集路径为/var/log/messages。

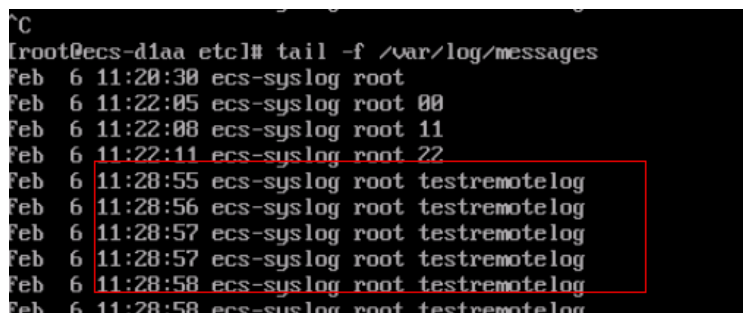
**步骤6** 登录业务ecs验证。

当您的业务系统或者设备输出日志后，即可在LTS界面查看。登录业务ecs使用logger -n x.x.x.x -P 514 testremotelog命令发送syslog至汇聚服务器，x.x.x.x为syslog服务器ip（公网ip/私有ip）地址，testremotelog为发送日志内容，可以自定义。

执行成功后，配置的日志组日志流里可以查看到该条日志。

或登录syslog汇聚服务器，查看/var/log/messages中是否存在testremotelog日志。

```
tail -f /var/log/messages
```

**图 2-23** 查看是否存在 testremotelog 日志**步骤7** 通过多台syslog服务器和elb实现负载均衡。

目前采集单台syslog服务器处理日志能力为10MB/s，如果客户日志业务量较大，可以使用多台syslog服务器和elb实现扩容和负载均衡。

1. 创建syslog汇聚服务器和安装ICAgent。
2. 请参考[创建ELB及配置](#)创建负载均衡器。
3. 分别为TCP/UDP的端口和514端口添加监听器，请参考[添加监听器](#)。
4. 为后端添加服务器组，请参考[添加syslog汇聚服务器](#)。

----结束

## 2.4 将自建 ELK 日志导入云日志服务 LTS

### 方案概述

ELK是Elasticsearch、Logstash和Kibana的简称，它们组合起来提供了业界最常用的日志分析和可视化工具。

- Elasticsearch是一个基于Lucene的开源、分布式、RESTful搜索和分析引擎。

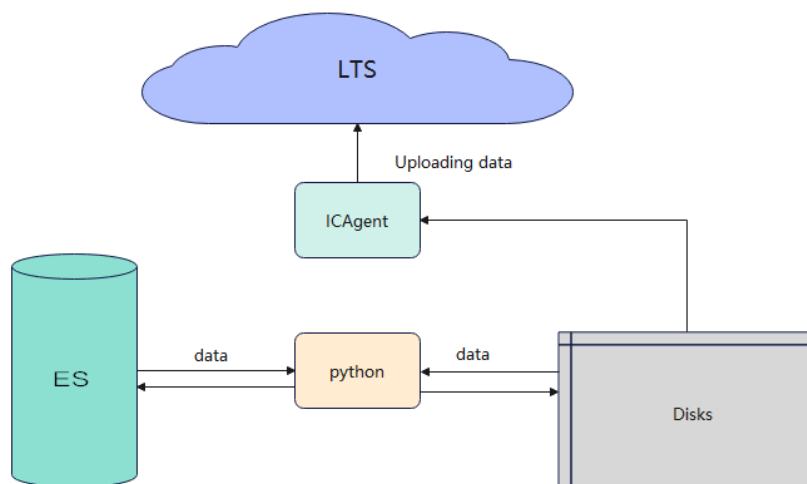
- Logstash是一个开源的、服务器端的数据处理管道，能够同时从多个来源实时接收、转换并将数据发送到用户选择的“存储库”。通常用于日志的收集、过滤和转发。
- Kibana是一个开源的分析和可视化平台，用于数据的可视化、仪表盘创建和搜索查询。通常与Elasticsearch一起使用。

华为云日志服务LTS在功能丰富度、成本、性能方面优于开源ELK方案，具体对比可以参考[云日志服务LTS对比自建ELK Stack有什么优势?](#)。本文提供最佳实践，使用自定义Python脚本和LTS采集器ICAgent，协助用户将日志从Elasticsearch（简称ES）迁移到LTS中。

当前华为云支持ECS机器通过安装ICAgent来采集日志文件，因此可以基于该功能实现Elasticsearch日志导入云日志服务。

Elasticsearch数据先通过python脚本将数据落盘到ECS，然后通过LTS服务的日志接入功能，将落盘的日志文件采集到LTS服务。

图 2-24 方案流程图



## 将自建 ELK 日志导入云日志服务 LTS

**步骤1** 登录[云日志服务控制台](#)。

**步骤2** 请参考[安装ICAgent](#)在ECS主机安装ICAgent。

**步骤3** 配置ECS日志接入云日志服务，请参考[ECS接入](#)。

**步骤4** 脚本执行前期准备。以下示例仅供参考，请以实际信息填写为准。

- 首次使用python，需要安装python环境。
- 首次使用ES时需要安装对应ES版本的python数据包，本次方案测试使用的elasticsearch为7.10.1版本。  

```
pip install elasticsearch==7.10.1
```
- 方案测试使用的ES为华为云CSS服务创建的ES。

**步骤5** 执行用来构造索引数据的python脚本，如果索引已经有数据，忽略这一步，直接执行5。

python脚本需执行在ECS机器，脚本命名为xxx.py格式，构造数据请参考如下示例：

### 📖 说明

以下斜体字段需按照实际情况进行修改，参考示例是插入1000条数据，内容为：This is a test log,Hello world!!!\n;

- **index**：要创建的索引名称，参考示例为: *test*;
- **链接ES**：ES的访问url，参考示例为: *http://127.0.0.1:9200*;

```
from elasticsearch import Elasticsearch
def creadIndex(index):
    mappings = {
        "properties": {
            "content": {
                "type": "text"
            }
        }
    }
    es.indices.create(index=index, mappings=mappings)
def reportLog(index):
    i = 0
    while i < 1000:
        i = i + 1
        body = {"content": "This is a test log,Hello world!!!\n"}
        es.index(index=index,body=body)
if __name__ == '__main__':
    #索引名称
    index = 'test'
    #链接ES
    es = Elasticsearch("http://127.0.0.1:9200")
    creadIndex(index)
    reportLog(index)
```

**步骤6** 构建python读写脚本，用来将ES数据写入磁盘，输出文件路径需与配置日志接入规则一致。

脚本需执行在ecs机器，命名xxx.py格式，写入磁盘数据脚本请参考如下示例：

### 📖 说明

以下斜体字段需按照实际情况进行修改。

- **index**：字段为索引名，参考示例为: *test*;
- **pathFile**：为数据写入磁盘绝对路径，参考示例为: */tmp/test.log*;
- **scroll\_size**：为索引滚动查询大小，参考示例为: *100*;
- **链接ES**：ES的访问url，参考示例为: *http://127.0.0.1:9200*;

```
from elasticsearch import Elasticsearch
def writeLog(res, pathFile):
    data = res.get('hits').get('hits')
    i = 0
    while i < len(data):
        log = data[i].get('_source').get('content')
        file = open(pathFile, 'a', encoding='UTF-8')
        file.writelines(log)
        i = i + 1
    file.flush()
    file.close()
if __name__ == '__main__':
    #索引名称
    index = 'test'
    #输出文件路径
    pathFile = '/tmp/' + index + '.log'
    #滚动查询一次滚动大小，默认为100
    scroll_size = 100
    #链接ES
    es = Elasticsearch("http://127.0.0.1:9200")
    init = True
    while 1:
```



```
if (init == True):
    res = es.search(index=index, scroll="1m", body={"size": scroll_size})
    init = False
else:
    scroll_id = res.get("_scroll_id")
    res = es.scroll(scroll="1m", scroll_id=scroll_id)
if not res.get('hits').get('hits'):
    break
writeLog(res, pathFile)
```

**步骤7** 执行命令前，请确保python已安装成功，在ECS执行如下命令，将ES索引数据写入磁盘。

```
python xxx.py
```

**步骤8** 查看数据是否成功查询及写入磁盘。

参考示例demo写入磁盘路径为：`/tmp/test.log`，操作时需要填写实际使用的路径，执行如下命令可以查看数据写入磁盘情况。

```
tail -f /tmp/test.log
```

**步骤9** 登录云日志服务控制台，在“日志管理”页面，单击日志流名称进入详情页面，在“日志搜索”页签查看到日志数据，即代表采集成功。

----结束

## 2.5 基于VPN打通网络实现API或SDK跨云上报日志到LTS

### 方案概述

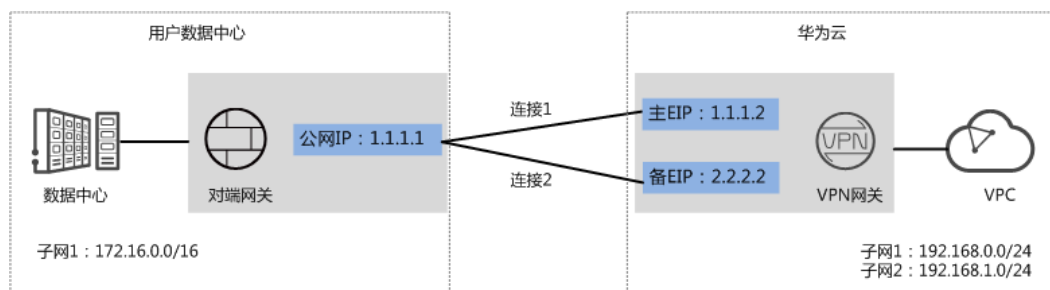
云上用户经常会遇到多云采集日志场景，需要将自建IDC、第三方云厂商、华为云其他region的日志采集到华为云LTS。支持基于VPN快速打通网络，实现API或SDK跨云上报日志。

本方案依赖云上VPN服务，可通过在数据中心建立主备两条VPN连接，提高网络连通的可靠性。

#### 📖 说明

当前仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一。

图 2-25 方案架构



### 约束与限制

- VPN网关的本端子网与对端子网不能相同，即VPC和用户数据中心待互通的子网不能相同。



- VPN网关和对端网关的IKE策略、IPsec策略、预共享密钥需要相同。
- VPN网关和对端网关上配置的本端接口地址和对端接口地址需要互为镜像。
- VPC内弹性云服务器安全组允许访问对端和被对端访问。

## 基于 VPN 打通网络实现 API 或 SDK 跨云上报日志到 LTS

**步骤1** 登录管理控制台。

**步骤2** 在左侧导航栏中，选择“网络 > 虚拟专用网络”。

**步骤3** 创建VPN网关。详细请参考[创建VPN](#)。

选择“虚拟专用网络 > 企业版-VPN网关”，单击“创建VPN网关”。根据界面提示选择相关参数，完成创建。

图 2-26 创建 VPN 网关

The screenshot shows the 'Create VPN Gateway' (创建VPN网关) configuration page. The page is divided into several sections:

- 计费模式 (Billing Mode):** 包年/包月 (Annual/Monthly) and 按需计费 (Pay-as-you-go).
- 区域 (Region):** 华南-广州 (South China - Guangzhou).
- 名称 (Name):** vpngw-222f.
- 网络类型 (Network Type):** 公网 (Public) selected, 私网 (Private) unselected.
- 关联模式 (Association Mode):** 虚拟私有云 (Virtual Private Cloud) selected, 企业路由器 (Enterprise Router) unselected.
- 虚拟私有云 (Virtual Private Cloud):** vpc-8e7e(192.168.0.0/16).
- 互联网子网 (Internet Subnet):** subnet-8e8e (192.168.0.0/24).
- 本地子网 (Local Subnet):** 选择子网 (Select Subnet) selected, 输入网段 (Enter CIDR) unselected.
- BGP ASN:** 64512.
- 规格 (Instance Type):** 基础型 (Basic), 专业型1 (Professional 1) selected, 专业型2 (Professional 2) unselected.
- 可用区 (Availability Zone):** 可用区6 (AZ6), 可用区3 (AZ3).
- 弹性公网IP组 (Elastic Public IP Group):**
  - 主EIP (Main EIP):** 现在创建 (Create Now) selected, 使用已有 (Use Existing) unselected.
  - 公网带宽 (Public Bandwidth):** 按带宽计费 (Pay by Bandwidth) selected, 按流量计费 (Pay by Traffic) unselected.
  - 带宽大小 (Bandwidth Size):** 5, 10, 20, 50, 100, 200, 300.
  - 带宽名称 (Bandwidth Name):** vpngw-bandwidth-224f.
  - 备用EIP (Backup EIP):** 现在创建 (Create Now) selected, 使用已有 (Use Existing) unselected.
  - 公网带宽 (Public Bandwidth):** 按带宽计费 (Pay by Bandwidth) selected, 按流量计费 (Pay by Traffic) unselected.
  - 带宽大小 (Bandwidth Size):** 5, 10, 20, 50, 100, 200, 300.
  - 带宽名称 (Bandwidth Name):** vpngw-bandwidth-225d.

**步骤4** 配置对端网关。

选择“虚拟专用网络 > 企业版-VPN连接”，单击“创建VPN连接”。

**图 2-27** 创建 VPN 连接**说明**

此处网关IP为数据中心的网关IP地址且需要放通UDP端口4500。

**步骤5** 配置VPN连接。

1. 选择“虚拟专用网络 > 企业版-VPN连接”，单击“创建VPN连接”。

2. 配置连接参数如下：

**连接模式**选择策略模式，**对端子网**填写数据中心的需要和华为云VPC通信的子网。

**预共享密钥**、**确认密钥**和对端网关的预共享密钥需要保持一致。

策略规则用于定义本端子网到对端子网之间具体进入VPN连接加密隧道的数据流信息，由源网段与目的网段来定义。源网段必须包含部分本端子网；目的网段必须完全包含对端子网。

**步骤6** 配置完成后，在“企业版-VPN连接”页面，VPN连接显示连接成功。也可通过数据中心ping对端虚拟机的私有ip进行验证。

**图 2-28** 查看 VPN 连接状态

**步骤7** VPN网络连通后，可通过API或SDK上报日志到云日志服务LTS。

**说明**

上报日志前，用户需要提供账号ID给云日志服务的技术支持工程师。

- 通过调用API上报日志。详细API接口请参考[上报日志](#)和[上报精度日志](#)。例如通过curl命令模拟调用API接口实现日志上报。

```
curl -ki -X POST 'https://lts-access.{region_id}.myhuaweicloud.com/v2/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/tenant/contents' -H "X-Auth-Token: $token" -H 'Content-Type: application/json;charset=UTF-8' -d '{
  "log_time_ns": "1686622249244000000",
  "contents": [
```



```
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}改为你虚拟机的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
```

```
# 填入你的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接受到的只有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。

### 📖 说明

- 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;
    }

    @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;
    }

    @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，如果你需要丰富日志内容，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() {
}
}
```

## 2.7 通过 ECS 接入 LTS 采集 Zabbix 数据

Zabbix作为常用的开源监控系统，提供了丰富的告警规则用于系统监控。云日志服务 LTS支持将Zabbix中的监控数据采集到日志流中。本文介绍通过ECS接入将Zabbix数据采集到云日志服务的操作步骤。

### 前提条件

- 准备好需要采集日志的ECS主机，详细请参考[购买弹性云服务器](#)。如果您已有可用的ECS主机，可重复使用，不需要再次创建。
- 已下载及在ECS机器上安装Zabbix，详细请参考[下载与安装Zabbix](#)。

### 步骤一：配置监控数据存储路径

Zabbix会将监控数据保存在其所在的机器上，您可以根据如下步骤设置监控数据的存储路径。

1. 登录Zabbix所在服务器。
2. 打开zabbix\_server.conf文件。  
`vim /etc/zabbix/zabbix_server.conf`
3. 在zabbix\_server.conf文件中，设置数据存储路径。  
`ExportDir=/tmp/`
4. 重启Zabbix服务，使配置生效。  
`systemctl restart zabbix-server`

配置生效后，Zabbix会在/tmp目录下生产文件（文件名后缀为.ndjson），用于保存监控数据。

## 步骤二：配置 ECS 接入 LTS

**步骤1** 在左侧导航栏中，选择“接入 > 接入中心”，单击“云主机 ECS-文本日志”进行主机接入配置。

**步骤2** 进入选择日志流页面。

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（例如lts-group-ECS）。
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（例如lts-topic-ECS）。
3. 单击“下一步：选择主机组（可选）”。

**步骤3** 选择主机组，单击“下一步：采集配置”。

**步骤4** 采集配置，路径配置为`/tem/**/*.*ndjson`，其余参数按照界面默认即可。更多设置请参考[云主机ECS文本日志接入LTS](#)。

图 2-30 采集配置

基本配置 ▾

\* 采集配置名称  [导入其他配置](#)

\* 路径配置  [添加自定义采集规则](#)

+ 添加采集路径

[了解更多Linux和Windows环境的采集路径规则。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

采集Windows事件日志

**步骤5** 单击“下一步：索引配置”，进入索引配置页面，按照界面默认参数配置即可，通过配置索引后，可对日志进行查询和分析操作。更多信息请参考[索引配置](#)。

**步骤6** 单击“提交”，日志接入成功，可以单击“返回接入配置列表”查看日志接入，在接入管理页签，则会生成一条接入配置信息。

**步骤7** 完成日志接入配置后，可以在云日志服务控制台实时查看上报的日志。

单击目标日志接入任务“所属日志流”列的日志流名称，即可进入日志流详情页。

**步骤8** 单击“实时日志”页签，查看实时日志。

日志大约每隔5秒钟上报一次，在日志消息区域，您最多需要等待5秒钟左右，即可查看实时上报的日志。

---结束

# 3 日志搜索与分析

## 3.1 在 LTS 页面分析华为云 ELB 日志

### 方案概述

ELB在分发外部流量时，详细的记录HTTP(S)的访问日志，如URI请求、客户端IP和端口、状态码。

ELB日志可用于审计，也可用于通过时间和日志中的关键词信息搜索日志，同时也可以通过各种SQL聚合函数来分析某段时间内的外部请求统计数据，比如统计1天内所有URI请求404的错误条数；分析1周内的UV（用户实际单击网站次数）或PV（网站的业务访问量），掌握真实用户的网站使用频率等。

### 资源规划


购买并使用华为云ELB实例。

### 限制条件

LTS ELB日志当前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。

## 在 LTS 页面分析华为云 ELB 日志

**步骤1** 将ELB访问日志对接至云日志服务详细操作请参见[访问日志](#)。

**步骤2** 在系统首页左上角单击 ，选择“管理与监管 > 云日志服务 LTS”。


**步骤3** 在日志管理页面，单击日志流名称进入日志流详情页面，单击右上角的  按钮，进入设置页面，在“云端结构化解析”页签，“自动配置索引和快速分析”默认开启，选择“结构化模板”提取方式，勾选ELB系统模板，单击“保存”，关于快速分析的更多详情请参见[快速分析](#)。

图 3-1 配置结构化模板



**步骤4** 在日志流详情页面，单击“日志分析”页签，进行SQL查询与分析，如需要多样化呈现查询结果，请参考[设置云端结构化解析日志](#)进行配置。

- 统计1周内的PV，具体SQL查询分析语句如下所示：

```
select count(*) as pv
```
- 统计1周内的UV，具体SQL查询分析语句如下所示：

```
select count(distinct remote_port) as uv
```
- 统计1天所有URI返回请求2xx/3xx/4xx/5xx（返回码），了解业务的执行结果。具体的SQL查询分析语句如下所示：

```
select host, router_request_uri as url, count(*) as pv,
sum(case when status >= 200 and status < 300 then 1 else 0 end) as "2xx times",
sum(case when status >= 300 and status < 400 then 1 else 0 end) as "3xx times",
sum(case when status >= 400 and status < 500 then 1 else 0 end) as "4xx times",
sum(case when status >= 500 and status < 600 then 1 else 0 end) as "5xx times"
group by host, router_request_uri
order by pv desc
limit 100
```

查询结果有表格、柱状图、折线图、饼图和数字图等呈现形式。更多信息请参考[使用统计图表将日志可视化](#)。

----结束

## 3.2 通过 LTS 仪表盘可视化 ELB 日志分析结果

当ELB日志接入云日志服务后，您可以通过SQL语句查询分析日志，将日志结果保存为多种图表，并将图表保存至仪表盘，从而使用仪表盘实时分析ELB日志数据。

### 前提条件

- 已采集ELB日志，具体操作，请参见[ELB接入](#)。
- 已对日志内容完成结构化配置，具体操作请参考[结构化配置](#)。

### 限制条件

- 一个日志流最多可创建100个图表。
- 一个仪表盘最多可创建50个图表。


- 一个仪表盘最多可添加10个过滤器。
- 一个华为账号最多可创建100个仪表盘。
- 一个华为账号最多可创建100个仪表盘模板。
- 一个华为账号最多可创建200个仪表盘分组。
- 一个华为账号最多可创建200个仪表盘模板分组。

## 操作流程



## 新建可视化图表

### 步骤1 SQL查询与分析。

1. 登录云日志服务控制台，在左侧导航栏中，选择“日志管理”。
2. 在日志列表中，单击日志组名称前对应的 ，选择目标日志流，进入日志详情页面。
3. 选择“日志分析”，在SQL查询条件框中，选择对应时间并输入SQL语句，单击“查询”进行[日志搜索](#)。

#### 说明

- 当设置时间范围内日志量超过10亿行时会触发迭代查询，可以通过迭代查询分多次完成全部日志的查询，界面会显示“查询状态：结果精确”。
- 根据SQL查询返回的数据，依照您的业务需求选择不同图表类型，呈现查询结果。
- 关于更多SQL查询的说明，请参见[SQL查询语法](#)。

### 步骤2 新建图表。

1. 单击“新建”，新建可视化图表。  
或单击“保存”，将当前查询结果新建为可视化图表。当选中某个可视化图表时，单击“保存”，可对当前图表修改结果进行保存。
2. 在创建图表页面中，配置相关参数。

#### 说明

如果开启“同时添加到仪表盘”按钮，新建图表可以直接添加到仪表盘中。

3. 完成后，单击“确定”。

### 步骤3 查看可视化图表。


单击“展开图表”，查看可视化图表。

----结束

## 将图表添加到仪表盘

将图表添加到仪表盘有两种方式：

方式一：

- 步骤1 鼠标悬浮可视化图表名称，单击  选择“添加到仪表盘”。



**步骤2** 在弹出的移动图表页面中，选择待存放的仪表盘。

**步骤3** 完成后，单击“确定”。

----结束

方式二：

**步骤1** 创建仪表盘。

1. 在左侧导航栏中，选择“仪表盘”。
2. 在仪表盘下方，选择仪表盘分组。
3. 单击“添加仪表盘”，在创建仪表盘页面，配置相关参数。

#### 说明

关于仪表盘参数的说明，请参见[仪表盘](#)。

**步骤2** 将图表添加到仪表盘。

1. 在创建仪表盘页面，单击“开始添加图表”，进入添加可视化图表界面，选择目标日志新建的[可视化图表](#)。
2. 完成后，单击“确定”。

----结束


## 添加过滤器

根据设置的变量添加过滤器的操作步骤如下：

**步骤1** 在左侧导航栏中，选择“仪表盘”。

**步骤2** 在仪表盘下方，选择仪表盘分组。

**步骤3** 单击待操作的仪表盘名称，进入仪表盘详情页面。

**步骤4** 单击，在过滤器页面中，配置相关参数，单击“确定”。

#### 说明

关于过滤器参数的说明，请参见[过滤器](#)。

**步骤5** 调整页面布局，单击“保存设计”。

**步骤6** 验证过滤结果。

----结束

## 3.3 在 LTS 页面分析华为云 WAF 日志

### 方案概述

WAF（Web应用防火墙）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/代码注入等攻击，所有请求流量经过WAF时，WAF会记录攻击和访问的日志，可实时决策分析、对设备进行运维管理以及业务趋势分析。

## 通过 LTS 分析 WAF 日志

WAF接入LTS后，支持[通过LTS分析WAF日志](#)。

# 3.4 在 LTS 页面分析 Log4j 格式的应用运行日志

## 背景信息

Log4j是Apache的一个开源项目，通过使用Log4j工具，我们可以将日志输出并保存到日志文件中，开发或运维人员会基于该日志统计日志级别的数量和占比，或者通过运行日志统计业务数据。

如统计今天某商品的交易量，示例日志如下：

```
2020-12-28_21:10:48.081 [http-nio-8083-exec-6] INFO discounted shoes - num is :9
```

## 在 LTS 页面分析 Log4j 格式的应用运行日志

**步骤1** 在云日志服务管理控制台，单击“日志接入”，进入日志接入页面。

**步骤2** 选择“云主机 ECS - 文本日志”，进行接入日志配置。

**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：选择主机组（可选）”。

**步骤4** 选择主机组。

1. 在主机组列表中选择一个或多个需要采集日志的主机组，若没有所需的主机组，单击列表左上方“新建”，在弹出的新建主机组页面创建新的主机组，具体可参考[创建主机组](#)。

### 说明


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。

- 在接入规则页面，单击操作列的编辑，进入接入配置页面对主机组和接入配置进行关联。
- 在“主机管理 > 主机组”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5** 采集配置。

1. 对主机日志采集设置具体的采集规则，具体可参考[采集配置](#)。
2. 单击“下一步：索引配置”，使用默认设置。
3. 单击“提交”，接入成功。

**步骤6** 在日志流详情页面，单击 ，在“云端结构化解析”页面，选择“正则分析”提取方式，根据业务需求选择日志，将日志正文拆分成四个部分：Time1/ThreadName/Level/Message。

**步骤7** 在日志流详情页面，单击“日志分析”页签，进行SQL查询与分析，如需要多样化呈现查询结果，请参考[日志结构化](#)进行配置。

- 统计最近7天内错误类型的分布，具体SQL查询分析语句如下所示：

```
SELECT Level, count(*) as Number group by Level
```

- 统计近5分钟内运行中的线程，具体SQL查询分析语句如下所示，输入查询语句并将时间选择为“近5分钟”。

```
SELECT distinct(ThreadName)
```

- 统计某型号商品的总交易量，具体SQL查询分析语句如下所示：

```
SELECT sum(cast(regex_extract(Message, 'num is\s:(?<Total>[\d]+)', 1) as double)) as Total WHERE Message like '%shoes%'
```

----结束

## 3.5 将 LTS 日志查询页面嵌入用户自建系统

LTS支持将日志查询界面嵌入到客户自建系统。通过统一身份认证服务IAM的联邦代理机制实现用户自定义身份代理，再将登录链接嵌入至客户自建系统实现无需在华为云官网登录就可在自建系统界面查询LTS日志。

### 应用场景

- 该功能主要用于用户可以在自建系统免密登录LTS的场景，但是登录华为云LTS控制台还是需要账号密码。
- 用户在外部系统中（例如公司内部运维或运营系统）快速集成云日志服务的查询、分析能力。
- 无需管理众多华为子账户，方便将日志数据进行分享查看。

### 将 LTS 日志查询页面嵌入用户自建系统

您需要先在IAM服务为用户自定义创建身份代理并创建委托，然后再将LTS日志查询页面嵌入用户自建系统。

**步骤1** 使用DomainA（该账号仅供参考，请以实际账号为准）登录统一身份认证服务控制台。

**步骤2** 在用户组页面创建IAM用户组（用户组名以GroupC为例）并授予全局服务中的Agent Operator权限，该权限仅能切换角色至委托方账号中，访问授权的服务，具体方法请参见：[创建用户组并授权](#)。

**步骤3** 在用户页面创建IAM用户（用户名以UserB为例），并加入GroupC用户组中，具体方法请参见：[用户组添加用户](#)。

#### 说明

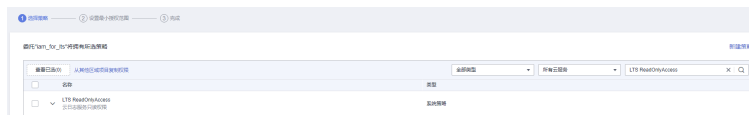
请确认该IAM用户支持[编程访问](#)和[管理控制台访问](#)云日志服务。如需修改IAM用户访问方式，请参考：[修改IAM用户信息](#)。

**步骤4** 在左侧导航栏选择“委托”，单击右上方的“创建委托”。

**步骤5** 在创建委托页面，设置委托参数。

1. “委托名称”以“iam\_for\_lts”为例，“委托类型”必须选择“普通账号”，“委托的账号”填写“DomainA”，“持续时间”选择“永久”，单击“下一步”。
2. 设置最小授权范围，选择“LTS ReadOnlyAccess”权限（该权限为LTS云日志服务的只读权限，只能查询LTS服务的数据，不能对LTS服务做设置修改），单击“下一步”。

**图 3-2** 选择策略



3. 选择授权范围方案，勾选“指定区域项目资源”，根据需要勾选对应区域，单击“确定”。

**步骤6** 使用postman等工具获取X-Subject-LoginToken参数。（以下示例截图仅供参考，请以实际获取的参数为准）

1. 通过账号密码获取UserB用户的X-Subject-Token。

接口类型：POST

接口url: `https://Endpoint/v3/auth/tokens`，参数选择自定义格式，并输入如下参数，其中name从上而下依次为租户名称、用户名称、租户名称，password为用户密码。

终端节点（Endpoint）即调用API的**请求地址**，不同服务在不同区域的终端节点不同，您可以从**地区和终端节点**中查询IAM服务的终端节点。

```
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "domain": {
            "name": "xxxxxxx"
          },
          "name": "xxxxxxx",
          "password": "xxxxxxx"
        }
      }
    },
    "scope": {
      "domain": {
        "name": "xxxxxxx"
      }
    }
  }
}
```

在返回结果中获取响应头中X-Subject-Token字段。

图 3-3 返回结果

```

General
Request URL: https://iam.myhuaweicloud.com/v3/auth/tokens
Request Method: POST
Status Code: 201

Response Headers:
cache-control: no-cache, no-store, must-revalidate
connection: keep-alive
content-length: 5482
content-type: application/json; charset=UTF-8
date: Tue, 26 Sep 2023 07:29:37 GMT
expires: Thu, 01 Jan 1970 00:00:00 GMT
pragma: no-cache
server: CloudWAF
strict-transport-security: max-age=31536000; includeSubdomains;
x-content-type-options: nosniff
x-download-options: noopen
x-frame-options: SAMEORIGIN
x-iam-trace-id: token_cn-north-4_null_f8530fd2e48e21cc953d48988219b639
x-request-id: f8530fd2e48e21cc953d48988219b639
X-Subject-Token
MIIRSQYJKoZihvcNAQcCoIIROJCCETYCAQEXDTALBgIghkgBZQMEAgEwg9bBgkqhkiG9w0BBwGggg9MBIPSHsidG9r:
a3jvRmc3TudvocOQBq+4-QLhbpckgY1M3LS7pFv0vW2rGJEPAYrk9V+tb5zBaH5RwE1rfMI99PxmSGSFhLh9EUH6WMr9:
+Zk1Y26HpaQrrTKKOG9+PYPRw02ktSvgPaDJoeWIMiyF-5T0Ng3BT3srVfVWZb3uPWjhm0Ls2r6w==
x-xss-protection: 1; mode=block;
    
```

2. 根据1获取的用户X-Subject-Token获取临时访问密钥。  
在请求header添加X-Auth-Token字段，value为1中获取到的X-Subject-Token。

图 3-4 获取临时访问密钥

Key	Value
X-Auth-Token	MIIRSQYJKoZihvcNAQcCoIIROJCCETYCAQEXDTALBgIghkgBZQMEAgEwg9bBgkqhkiG9w0BBwGggg9MBIPSHsidG9r: a3jvRmc3TudvocOQBq+4-QLhbpckgY1M3LS7pFv0vW2rGJEPAYrk9V+tb5zBaH5RwE1rfMI99PxmSGSFhLh9EUH6WMr9: +Zk1Y26HpaQrrTKKOG9+PYPRw02ktSvgPaDJoeWIMiyF-5T0Ng3BT3srVfVWZb3uPWjhm0Ls2r6w==

接口类型：POST

接口url: https://Endpoint/v3.0/OS-CREDENTIAL/securitytokens, 参数选择自定义格式, 并输入如下参数, 其中具体参数含义分别为: agency\_name为委托名称、domain\_name为租户名称、duration\_seconds为token过期时间(单位秒)、name为用户名。

```

{
  "auth": {
    "identity": {
      "methods": [
        "assume_role"
      ],
      "assume_role": {
        "agency_name": "iam_for_its",
        "domain_name": "xxxxxx",
        "duration_seconds": 86400,
        "session_user": {
          "name": "xxxxxx"
        }
      }
    }
  }
}
    
```

在返回结果中获取响应体中获取临时访问密钥。

图 3-5 获取临时访问密钥

```
{
  - credential: {
    access: "ZMC5PD5C5IE5V10X4JCE",
    expires_at: "2023-09-27T07:33:18.912000Z",
    secret: "lOA5hKWDuxLYN3uJLUOGqB9g2RDvOFdkRty32h7X",
    securitytoken: "gQpjbi1ub3J0aC00iZMdAa3kx9GOlg0zTTob5wvpFPee-hVQjagvQfE_f
XhCXSmJw79obJuQVHeLA0SGiPTey_4OBI-5OmBwDuYXgLiXMcTIS4XoXBAXqo4hYR
QGvI4heEj3X834BlpfOApOBLA1433er9ViO6Gz_qio48jXSSyPBQ2i993320D3IBWUA0r
XEIJtk5OpIOYWWU56DmPHNDvaX1AwxwTzsXGg29dLW27L-RVvp6wN9WGVgbgWKJ
iQkAjAMnx6_ajfmcptquc7ibB1JsoF8vB5baQ8eOKpsSypCqLiSY7vhWgicykmmKUcW_)
uNqz24LzPaxaUZEv9sMeJK9Mlq7dfccachmDw5wXGGwQzIV8bT2GZr15xd0qipVbM
RdefvTQWYon1Qzc3pL5pkw7Qn491FN9rJqpG6IkXiSjihyMY6smZEmBVPrQd75CHUI
1E6YRCvEkQxtCtmqolLuRDzd6-lpEjEKEutLR_fHLPGeOvCmkAklytgkCag-_zFneRlvht
U19ttPcyVRxsbbpkfbox2jVGWyrIH4GvvfEfZbOYAQ0jilPgGctfxwGaUm8slQCyyBPjP
XK8UDV8uioCv5QNMkjXLCXiAaW7bshSITqn66b9LCOp36q_CvqfCn2XgWmMzHP2vtv
  }
}
```

3. 根据2获取的临时访问密钥获取登录X-Subject-LoginToken。

接口类型：POST

接口url: <https://Endpoint/v3.0/OS-AUTH/securitytoken/logintokens>, 参数选择自定义格式, 并输入如下参数, 其中access、secret、id对应的值分别为2返回的access、secret、securitytoken, duration\_seconds为token过期时间, 单位为秒。

```
{
  "auth": {
    "securitytoken": {
      "access": "xxxxxx",
      "secret": "xxxxxx",
      "id": "xxxxxx",
      "duration_seconds": 43200
    }
  }
}
```

在返回结果中获取响应头中的X-Subject-LoginToken字段。

图 3-6 获取 X-Subject-LoginToken

```
General:
Request URL: https://iam.myhuaweicloud.com/v3.0/OS-AUTH/securitytoken/logintokens
Request Method: POST
Status Code: 201

Response Headers:
cache-control: no-cache, no-store, must-revalidate
connection: keep-alive
content-length: 529
content-type: application/json; charset=UTF-8
date: Tue, 26 Sep 2023 07:34:56 GMT
expires: Thu, 01 Jan 1970 00:00:00 GMT
pragma: no-cache
server: CloudWAF
strict-transport-security: max-age=31536000; includeSubdomains;
x-content-type-options: nosniff
x-download-options: noopen
x-frame-options: SAMEORIGIN
x-iam-trace-id: token_cn-north-4_null_dfa3dffde609d11e6f9f5d2bdc669f7e
x-request-id: dfa3dffde609d11e6f9f5d2bdc669f7e
x-subject-logintoken: MIIEEgYJKoZIhvcNAQcCoIIeAZCCA-
8CAQExDTALBgIghkgBZQMEAgEwggIkBgkqhkiG9w0BBwGgggIVBIIcEXsibG9naW50b2tlibiI6eyJkb21haW5l
mDmgm7xaRF7MPveGMBMj8worNmn8r+NckfKGYUpxgHbCFIdnaFbl9YGZWCBBNyul1zTcdIXjK-YZrB5iLs
WcdOcoAQWFEVtju9iGnCN6ve3ESULb5+61FQGtkoQ7dxITJobYmL5rjnmHssnKmvbll5eJpsFGddV1nTFC
WDq8ZzItpZRe8B5NTvOwXvCq5KBBBeup+e6EXGZ2S6uT7THuXYFRuQBIGCJLRsHsC4oww54yAKNozvTr
x-xss-protection: 1; mode=block;
```

**步骤7** 根据3获取的X-Subject-LoginToken构建代理URL，完成免密登录。

代理登录地址的构建规则为：

```
https://auth.huaweicloud.com/authui/federation/login?  
service={target_console_url}&logintoken={logintoken}&idp_login_url={enterprise_s  
ystem_loginURL}
```

**表 3-1** URL 参数说明

参数名称	说明
{target_console_url}	云日志服务地址说明的urlencode编码结果。详细请参考 <a href="#">云日志服务地址说明</a> 。
{logintoken}	为3获取的X-Subject-LoginToken的urlencode编码结果。
{enterprise_system_login URL}	选填参数，为客户的页面地址的urlencode编码结果，当loginToken验证失效时会跳转到该页面。

#### 说明

- 以上三个参数都需要进行urlencode编码，否则可能导致免密登录失败。
- urlencode编码操作方式：打开浏览器按F12进入开发者模式，选择console（控制台），输入“encodeURIComponent('\*')”，\*为需要编码的信息，单击“Enter”，查看返回的urlencode编码值。

{target\_console\_url}的值为LTS前台服务URL地址的urlencode编码，编码前URL如下，具体参考[表3-2](#)。

```
https://console.huaweicloud.com/lts/?  
region={regionId}&cfModuleHide=header_sidebar_floatlayer#/lts/  
logEventsLeftMenu/events?  
groupId={groupId}&topicId={topicId}&epsId={epsId}&condition={condition}
```

**表 3-2** 参数说明

参数名称	说明
{regionId}	区域ID，登录console页面后，在浏览器地址栏中获取。
{groupId}	日志组ID。
{topicId}	日志流ID。
{epsId}	日志流所属的企业项目ID，若无企业项目，值为0。

参数名称	说明
{condition}	日志查询条件，如name:a and age:12 and addr:xx。 <ul style="list-style-type: none"> <li>非必填</li> <li>单个关键词形式为key:value</li> <li>多个关键词用 and 隔开</li> <li>关键词中不能包含英文分号 (;)、英文冒号 (:)</li> <li>关键词中含有其他特殊字符如 (+、=、?、#、%、&amp;) 需转换为十六进制，即%加字符的ASCII码 (%2B、%3D、%3F、%23、%25、%26)</li> </ul>

**步骤8** 完成以上步骤，即可实现在自建系统免密登录云日志服务LTS。

使用如下iframe嵌套，其中src的值为7中获得的代理URL。

```
<body>
  <iframe src="target_url" width="100%" height="96%" id="ltsiframePage"></iframe>
</body>
```

----结束

## 云日志服务地址说明

1. 云日志服务首页，基础URL为：

```
https://console.huaweicloud.com/lts/?
region={regionId}&cfModuleHide=header_sidebar_floatlayer#/cts/manager/groups
```

表 3-3 参数说明表

参数名称	必填	类型	描述
regionId	是	String	区域ID，登录console页面后在浏览器的地址栏中获取。

2. 日志搜索界面，基础URL为：

```
https://console.huaweicloud.com/lts/?
region={regionId}&cfModuleHide=header_sidebar_floatlayer#/cts/logEventsLeftMenu/events?
groupId={groupId}&topicId={topicId}&epsId={epsId}&hideHeader={hideHeader}&fastAnalysisCollapsed=
{fastAnalysisCollapsed}&hideDashboard={hideDashboard}&hideFeedback={hideFeedback}&isFoldLabel=
{isFoldLabel}&hideStreamName={hideStreamName}&showK8sFilter={showK8sFilter}&clusterId={clus
terId}&hideBarChart={hideBarChart}&hideTabs={hideTabs}&condition={condition}
```

表 3-4 参数说明表 c

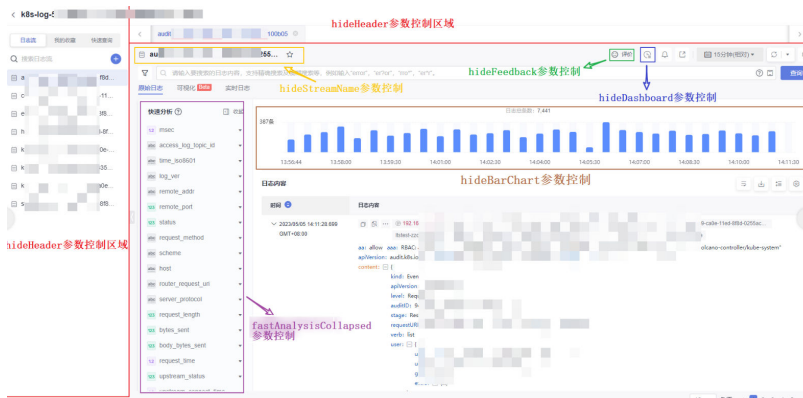
参数名称	必填	类型	默认值	描述
regionId	是	String	无	区域ID，登录console页面后在浏览器的地址栏中获取。
groupId	是	String	无	日志组ID。
topicId	是	String	无	日志流ID。



参数名称	必填	类型	默认值	描述
epsId	否	String	无	日志流所属的企业项目ID，若无企业项目，值为0。
hideHeader	否	Boolean	false	是否隐藏左侧列表及顶部横向日志流列表，如需隐藏，该参数值为true。 <b>说明</b> 使用iframe内嵌场景才会生效，不使用iframe内嵌场景则不生效。
fastAnalysisCollapsed	否	Boolean	false	是否收起快速分析，如需默认收起，该参数值为true。
hideDashboard	否	Boolean	false	是否隐藏创建仪表盘图标，如需隐藏，该参数值为true。
hideFeedback	否	Boolean	false	是否隐藏评价按钮，如需隐藏，该参数值为true。
isFoldLabel	否	Boolean	true	控制日志表格中的label字段是否换行，如需换行展示，该参数值为true。
hideStreamName	否	Boolean	false	是否隐藏日志流名称，如需隐藏，该参数值为true。
showK8sFilter	否	Boolean	false	是否展示容器日志筛选，容器日志搜索场景下，可选择该参数为true，控制是否展示容器日志筛选条件。
clusterId	否	String	无	集群ID，showK8sFilter参数为true时，该参数必填。
hideBarChart	否	Boolean	false	是否默认收起日志条数统计图，如需默认收起，该参数值为true。
hideTabs	否	Boolean	false	是否隐藏“日志搜索、日志分析、实时日志”标签tabs，默认不隐藏。如需隐藏，该参数值为true。
hideShare	否	Boolean	false	是否隐藏“分享”按钮，默认不隐藏。如需隐藏，该参数值为true。（当前仅华北-北京四局点支持该参数）

参数名称	必填	类型	默认值	描述
condition	否	String	无	日志查询条件，如name:a and age:12 and addr:xx。 <ul style="list-style-type: none"> <li>• 非必填</li> <li>• 单个关键词形式为key:value</li> <li>• 多个关键词用 and 隔开</li> <li>• 关键词中不能包含英文分号 (;)、英文冒号 (:)</li> <li>• 关键词中含有其他特殊字符如 (+、=、?、#、%、&amp;) 需转换为十六进</li> </ul>

图 3-7 参数与界面对应关系图（仅供参考）



3. 可视化日志搜索界面，基础URL为：

[https://console.huaweicloud.com/lts/?region={regionId}&cfModuleHide=header\\_sidebar\\_floatlayer#/cts/logEventsLeftMenu/events?visualization=true&groupId={groupId}&topicId={topicId}&epsId={epsId}&sql={sql}](https://console.huaweicloud.com/lts/?region={regionId}&cfModuleHide=header_sidebar_floatlayer#/cts/logEventsLeftMenu/events?visualization=true&groupId={groupId}&topicId={topicId}&epsId={epsId}&sql={sql})

表 3-5 参数说明表

参数名称	必填	类型	默认值	描述
regionId	是	String	无	区域ID，登录console页面后在浏览器的地址栏中获取。
groupId	是	String	无	日志组ID。
topicId	是	String	无	日志流ID。
epsId	否	String	无	日志流所属的企业项目ID，若无企业项目，值为0。

参数名称	必填	类型	默认值	描述
hideHeader	否	Boolean	false	是否隐藏左侧列表及顶部横向日志流列表，如需隐藏，该参数值为true。
sql	否	String	无	SQL查询语句，如SELECT count (*)。

图 3-8 参数与界面对应关系图（仅供参考）



4. 仪表盘界面，基础URL为：

[https://console.huaweicloud.com/lts/?region={regionId}&cfModuleHide=header\\_sidebar\\_floatlayer#/cts/manager/dashboard?dashboardId={dashboardId}&hideDashboardList={hideDashboardList}&showCurrentdashboardGroup={showCurrentdashboardGroup}&streamId={streamId}&streamDisabled={streamDisabled}&readonly={readonly}&filter=key1:value1,value2;key2:value3,value4&autoFresh={autoFresh}](https://console.huaweicloud.com/lts/?region={regionId}&cfModuleHide=header_sidebar_floatlayer#/cts/manager/dashboard?dashboardId={dashboardId}&hideDashboardList={hideDashboardList}&showCurrentdashboardGroup={showCurrentdashboardGroup}&streamId={streamId}&streamDisabled={streamDisabled}&readonly={readonly}&filter=key1:value1,value2;key2:value3,value4&autoFresh={autoFresh})

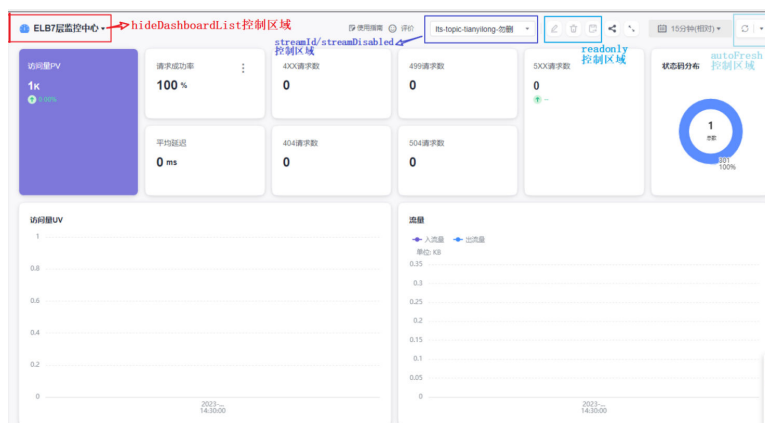
表 3-6 参数说明表

参数名称	必填	类型	默认值	描述	示例
regionId	是	String	无	区域ID，登录console页面后在浏览器的地址栏中获取。	region=xx-xx-xx
dashboardId	否	String	无	需要展示的仪表盘ID，默认值为""。 使用场景：当用户需要默认展示某仪表盘时，可添加此参数。	dashboardId=xxxxxxx

参数名称	必填	类型	默认值	描述	示例
hideDashboardList	否	Boolean	false	<p>是否隐藏仪表盘选择下拉列表：默认不隐藏，true表示隐藏。</p> <p><b>使用场景：</b>当用户需要隐藏仪表盘下拉列表时，可通过添加该参数且值为true来实现。</p>	hideDashboardList=true
showCurrentDashboardGroup	否	Boolean	false	<p>是否只展示当前仪表盘所在分组/模板：默认值为false。</p> <p><b>使用场景：</b>当用户只需要展示当前仪表盘分组/模板的仪表盘时，可通过添加该参数且值为true来实现。</p> <p><b>注意：</b>若hideDashboardList参数值为true时，当前参数无效。</p>	showCurrentDashboardGroup=true
streamId	否	String	无	<p>日志流ID：默认值为""。</p> <p><b>使用场景：</b>只适用于仪表盘模板。当用户需要默认选中某日志流时，可添加该参数。</p>	streamId=xxxxxx
streamDisabled	否	Boolean	false	<p>日志流下拉框：默认可选择，true标识不可选择</p> <p><b>使用场景：</b>只适用于仪表盘模板。当用户需要置灰日志流下拉框时，可添加该参数。</p>	streamDisabled=true

参数名称	必填	类型	默认值	描述	示例
filter	否	String	无	<p>过滤器参数，值为要选中的过滤器的名称及选中项。</p> <p>key1、key2为过滤器名称，value1、value2为过滤器key1需要选中的值，value3、value4为过滤器key2需要选中的值。多个过滤器按照;分隔，多个选中项按照,分隔。</p> <p><b>使用场景：</b>当用户内嵌仪表盘界面需要默认选中某些过滤器的key、value时，可添加该参数。</p>	filter=key1:value1,value2;key2:value3,value4
readonly	否	Boolean	false	<p>是否是只读场景，只读场景下，操作类按钮会被隐藏。例如：新建过滤器、添加/修改/删除仪表盘等。</p> <p><b>使用场景：</b>当用户只需要展示仪表盘，不需要操作权限时，可添加该参数。</p>	readonly=true
autoFresh	否	String	无	<p>定时刷新时长，默认值为""。</p> <p><b>使用场景：</b>当用户需要指定默认定时刷新时长时，可添加此参数，当前支持的定时刷新时长参数取值为：0m, 1m, 5m, 15m之一，对应：不定时刷新、1分钟定时刷新、5分钟定时刷新、15分钟定时刷新。</p>	autoFresh=1m

图 3-9 参数与界面对应关系图（仅供参考）



# 4 日志转储

## 4.1 批量修改 LTS 日志文件转储时区

您在LTS上经常会执行日志接入、日志告警、日志转储等配置操作。有些操作是需要重复多次配置，但目前LTS还没有提供控制台批量操作功能，这时您可以通过Python脚本结合LTS API接口实现自定义的批量操作。

### 使用场景

当用户创建了1000条日志转储的OBS规则，但转储时文件时区全部选择(UTC) 协调世界时间，现在需要根据实际情况修改为(UTC+08:00) 北京。目前LTS控制台没有提供批量修改功能，如果手动修改每条转储规则，会导致人工耗时非常长。

### 前提条件

1. Linux系统的主机。
2. 查询API相关接口文档。
  - 通过查询日志转储API获取到所有转储任务的信息。
  - 通过更新日志转储API将转储任务配置的时区修改。
3. 在API Explorer中测试API功能，API Explorer提供API检索及平台调试能力。
4. 参考API Explorer示例代码，在主机上安装Python SDK。

- Python的[SDK依赖包地址](#)以及[SDK使用说明](#)。

```
pip install huaweicloudsklts
```

- API Explore提供Python调用API的示例代码，以下示例仅供参考：

```
# coding: utf-8
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsklts.v2.region.lts_region import LtsRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsklts.v2 import *
if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has
    # great security risks. It is recommended that the AK and SK be stored in ciphertext in
    # configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
    # running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the
    # local environment
    /* 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;
ak = os.environ["CLOUD_SDK_AK"]
sk = os.environ["CLOUD_SDK_SK"]
credentials = BasicCredentials(ak, sk)
client = LtsClient.new_builder() \
    .with_credentials(credentials) \
    .with_region(LtsRegion.value_of("xx")) \
    .build()
try:
    request = ListTransfersRequest()
    response = client.list_transfers(request)
    print(response)
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
    
```

## 批量修改 LTS 日志文件转储时区

步骤1 获取参数，并将实际值替换到代码中。

- [获取华为账号的AK/SK](#)
- 获取项目ID ( project id )，详细步骤请参考[API凭证](#)。

图 4-1 获取 Project\_ID



- 请参考[地区和终端节点](#)获取Region&iam\_Endpoint。

表 4-1 Endpoint 表

区域名称	区域	终端节点 ( Endpoint )	协议类型
华北-北京三	cn-north-3	lts.cn-north-3.myhuaweicloud.com	HTTPS
华北-北京四	cn-north-4	lts.cn-north-4.myhuaweicloud.com	HTTPS
华北-北京一	cn-north-1	lts.cn-north-1.myhuaweicloud.com	HTTPS
华东-上海二	cn-east-2	lts.cn-east-2.myhuaweicloud.com	HTTPS
华东-上海一	cn-east-3	lts.cn-east-3.myhuaweicloud.com	HTTPS



区域名称	区域	终端节点 ( Endpoint )	协议类型
华南-广州	cn-south-1	lts.cn-south-1.myhuaweicloud.com	HTTPS
华南-深圳	cn-south-2	lts.cn-south-2.myhuaweicloud.com	HTTPS
亚太-曼谷	ap-southeast-2	lts.ap-southeast-2.myhuaweicloud.com	HTTPS
亚太-新加坡	ap-southeast-3	lts.ap-southeast-3.myhuaweicloud.com	HTTPS
中国-香港	ap-southeast-1	lts.ap-southeast-1.myhuaweicloud.com	HTTPS

- 获取时区和时区ID。

表 4-2 常用时区表

时区	时区ID
UTC-12:00	Etc/GMT+12
UTC-11:00	Etc/GMT+11
UTC-10:00	Pacific/Honolulu
UTC-09:00	America/Anchorage
UTC-08:00	America/Santa_Isabel
UTC-07:00	America/Chihuahua
UTC-06:00	America/Chicago
UTC-05:00	America/New_York
UTC-04:00	America/Santiago
UTC-03:00	America/Sao_Paulo
UTC-02:00	Etc/GMT+2
UTC-01:00	Atlantic/Azoresjavik
UTC+00:00	Europe/London
UTC+01:00	Europe/Parist

时区	时区ID
UTC+02:00	Europe/Istanbul
UTC+03:00	Europe/Minsk
UTC+04:00	Europe/Moscow
UTC+05:00	Asia/Tashkent
UTC+06:00	Asia/Almaty
UTC+07:00	Asia/Bangkok
UTC+08:00	Asia/Shanghai
UTC+09:00	Asia/Tokyo
UTC+10:00	Asia/Yakutsk
UTC+11:00	Asia/Vladivostok
UTC+12:00	Pacific/Fiji
UTC+13:00	Pacific/Apia

**步骤2** 在主机上执行如下命令确认是否已安装huaweicloudsdkcore和huaweicloudsklts包。

```
pip list | grep huaweicloudsdk
```

- 如果已安装，执行结果会返回huaweicloudsdk相关信息。如果没有安装，则不会返回任何内容。
- 若未安装在主机上请执行如下操作进行安装：  

```
pip install huaweicloudsdkcore huaweicloudsklts
```

**步骤3** 在主机上执行“vi lts\_python.py”新建lts\_python.py文件，将如下代码复制到该文件中，用于实现批量修改OBS转储文件时区。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsklts.v2 import *
from huaweicloudsklts.v2.region.lts_region import LtsRegion
/* 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;
if __name__ == "__main__":
    AK = "your ak"
    SK = "your sk"
    PROJECT_ID = "your project id"
    REGION = "your region"
    IAM_ENDPOINT = "iam_endpoint"

    OBS_TIME_ZONE = "the time_zone you want to change"
    OBS_TIME_ZONE_ID = "time_zone_id"

    credentials = BasicCredentials(AK, SK, PROJECT_ID).with_iam_endpoint(IAM_ENDPOINT)

    client = LtsClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(LtsRegion.value_of(REGION)) \
        .build()

# 1.get obs transfer task
```

```
try:
    request = ListTransfersRequest()
    request.log_transfer_type = "OBS"
    response = client.list_transfers(request)
    obs_transfer_num = len(response.log_transfers)
    task_list = response.log_transfers
    print("#### get {} obs transfer task ####".format(obs_transfer_num))
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)

# 2.set obs transfer task obs_time_zone to UTC+08:00
CNT = 1
while len(task_list):
    transfer_task = task_list.pop()
    print("There are still {} progress: \n".format(len(task_list), transfer_task))
    try:
        if transfer_task.log_transfer_info.log_transfer_detail.obs_time_zone == OBS_TIME_ZONE:
            CNT += 1
            continue
        request = UpdateTransferRequest()
        transfer_task.log_transfer_info.log_transfer_detail.obs_time_zone = OBS_TIME_ZONE
        transfer_task.log_transfer_info.log_transfer_detail.obs_time_zone_id = OBS_TIME_ZONE_ID
        request.body = UpdateTransferRequestBody(
            log_transfer_info=transfer_task.log_transfer_info,
            log_transfer_id=transfer_task.log_transfer_id
        )
        response = client.update_transfer(request)
        CNT += 1
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
        task_list.append(transfer_task)
    except exceptions.ServerResponseException as e:
        print({
            "target": transfer_task.log_streams,
            "reason": e
        })
        task_list.append(transfer_task)
```

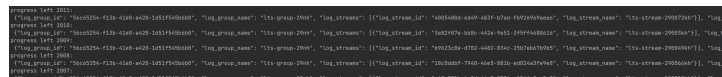
**步骤4** 在主机上执行Python脚本，批量修改OBS转储文件时区。

```
nohup python lts_python.py > lts_python.log &
```

**步骤5** 查看执行日志确认Python脚本运行成功，文件时区已修改。

```
tail -f lts_python.log
```

图 4-2 查看执行日志



```
INFO: [2024-09-29 10:00:00] [log_group_name: "log_group_2024", "log_streams": [{"log_stream_id": "2024092910000000000000000000000000", "log_stream_name": "log_stream_2024092910000000000000000000000000"}]]
INFO: [2024-09-29 10:00:00] [log_group_name: "log_group_2024", "log_streams": [{"log_stream_id": "2024092910000000000000000000000000", "log_stream_name": "log_stream_2024092910000000000000000000000000"}]]
INFO: [2024-09-29 10:00:00] [log_group_name: "log_group_2024", "log_streams": [{"log_stream_id": "2024092910000000000000000000000000", "log_stream_name": "log_stream_2024092910000000000000000000000000"}]]
INFO: [2024-09-29 10:00:00] [log_group_name: "log_group_2024", "log_streams": [{"log_stream_id": "2024092910000000000000000000000000", "log_stream_name": "log_stream_2024092910000000000000000000000000"}]]
INFO: [2024-09-29 10:00:00] [log_group_name: "log_group_2024", "log_streams": [{"log_stream_id": "2024092910000000000000000000000000", "log_stream_name": "log_stream_2024092910000000000000000000000000"}]]
```

---结束

# 5 日志计费

## 5.1 通过日志流标签统计不同部门在 LTS 的费用开销

为了统计企业内部不同部门在LTS的费用开销情况，您可以在LTS的日志流上添加标签用于识别不同的业务部门，LTS在上传话单给费用中心时会带上这些标签信息。您可以在“账单管理 > 流水和明细账单 > 明细账单”下载LTS的明细账单信息，然后基于资源标签来统计不同部门的费用开销，为企业内部的费用分摊提供依据。

### 前提条件

按日志流维度上报话单功能目前在友好用户内测中，若您需要使用日志流标签统计不同部门在LTS的费用开销，请[提交工单](#)申请开通。

### 方案介绍

日志流是通过日志组管理的，给日志组添加标签时，默认开启应用到日志流，这样日志流就自动添加标签。即可通过日志流统计不同部门在LTS的费用开销。


本实践以aa和bb部门为例子，首先在aa部门的日志组添加group=groupaa标签，bb部门的日志组添加group=groupbb标签，然后在费用明细中导出账单，通过Excel进行统计分析。

#### 说明

以下提到的价格仅为示例，实际计算请以[价格计算器](#)中的价格为准。

## 通过日志流标签统计不同部门在 LTS 的费用开销

**步骤1** 登录云日志服务控制台。

**步骤2** 在“日志管理”页面，将鼠标悬浮在aa部门创建的日志组“标签”列单击 。

**步骤3** 在弹出的编辑标签页面，单击“添加标签”，填写aa部门的标签键group和标签值groupaa，默认开启“应用到日志流”，将日志组标签同步到日志流，单击“确定”。

图 5-1 aa 部门添加标签



**步骤4** 将鼠标悬浮在bb部门创建的日志组“标签”列单击

**步骤5** 在弹出的编辑标签页面，单击“添加标签”，填写bb部门的标签键group和标签值groupbb，默认开启“应用到日志流”，将日志组标签同步到日志流，单击“确定”。

图 5-2 bb 部门添加标签



**步骤6** 标签添加成功后预计需要等待一个小时才能生成话单。

**步骤7** 在控制台顶部菜单栏中选择“费用 > 费用账单”，进入“账单概览”页面。

**步骤8** 左侧导航栏选择“流水和明细账单”，在“明细账单”页签，统计维度选择“按使用量”，统计周期选择“明细”，在筛选条件中选择“产品类型：云日志服务”。详细操作请参考[流水与明细账单](#)。

**步骤9** 单击“导出”，在导出页面自定义设置导出范围，将费用明细导出到本地查看，详细操作请参考[账单导出](#)。

**步骤10** 费用明细导出的Excel文件中，筛选“资源标签”，过滤标签名称，即可更直观的查看到aa和bb部门的消费明细详情。

#### 说明

实际计算请以[价格计算器](#)中的价格为准。



# 6 日志加工（邀测）

## 📖 说明

DSL加工的功能在邀测中，支持华北-北京四、华东-上海一、华南-广州局点，仅针对用户内测使用，后续将全网开放，敬请期待！

## 6.1 使用 DSL 加工函数清洗 LTS 日志数据

您可以通过日志加工函数清洗您所采集的海量日志数据，实现数据格式标准化。本文介绍调用函数清洗数据的常见场景和相关操作。

## 📖 说明

DSL加工的功能在邀测中，支持华北-北京四、华东-上海一、华南-广州局点，仅针对用户内测使用，后续将全网开放，敬请期待！

### 6.1.1 场景一：过滤 LTS 日志

您可以使用 `e_if` 函数与 `DROP` 参数、`e_if_else` 函数与 `DROP` 参数过滤日志，也可以使用 `e_drop` 函数或 `e_keep` 函数过滤日志。

常用规则如下所示：

- `e_keep(e_has(...))`：满足条件时保留，不满足条件时丢弃。
- `e_drop(e_has(...))`：满足条件时丢弃，不满足条件时保留。
- `e_if_else(e_has(...), KEEP, DROP)`：满足条件时保留，不满足条件时丢弃。
- `e_if(e_not_has(...), DROP)`：满足条件时丢弃，不满足条件时保留。

示例如下所示：

- 原始日志

```
[[ //日志1
  "source": "192.168.0.1",
  "client_ip": "192.168.0.2",
  "receive_time": 1587214851,
  "topic": "app",
  "class": "test_case",
  "id": 7892,
  "content": "this is a log test"
],
{ //日志2
```

```
"source": "192.168.0.1",  
"class": "produce_case",  
"id": 7890,  
"content": "this is a log test"  
}]
```

- 加工规则：丢弃没有topic字段和receive\_time字段的日志。

```
e_if(e_not_has("topic"),e_drop())  
e_if(e_not_has("receive_time"),e_drop())
```

- 加工结果

```
{  
source: 192.168.0.1  
client_ip: 192.168.0.2  
receive_time: 1587214851  
topic: app  
class: test_case  
id: 7892  
content: this is a log test  
}
```

## 6.1.2 场景二：使用 e\_set 函数为日志空缺字段赋值

您可以使用e\_set函数为日志空缺字段赋值。

1. 场景1：原字段不存在或者为空时，为字段赋值。

```
e_set("result", ".....value.....", mode="fill")
```

示例如下所示：

- 原始日志

```
{  
  "name": ""  
}
```

- 加工规则

```
e_set("name", "Apache 2.0", mode="fill")
```

- 加工结果

```
{  
  name: Apache 2.0  
}
```

2. 场景2：为多个字段赋值。

```
e_set("k1", "v1", "k2", "v2")
```

示例如下所示：

- 原始日志

```
{  
  "source": "192.168.0.1",  
  "topic": "",  
  "tag": "",  
  "id": 7990,  
  "content": "this is a log"  
}
```

- 加工规则

```
e_set("topic", "app", "tag", "stu")
```

- 加工结果

```
{  
  topic: app  
  source: 192.168.0.1  
  tag: stu  
  id: 7990  
  content: this is a log  
}
```



### 6.1.3 场景三：删除和重命名字段（e\_drop\_fields 函数和 e\_rename 函数）

一般情况下，推荐您使用e\_compose函数进行多个函数的组合操作。

示例如下所示：

- 原始日志

```
{
  "content": "123",
  "age": 23,
  "name": "twiss",
  "IdNumber": 1
}
```
- 加工规则：首先判断content字段值是否为123，如果是，则删除age和name字段，再将IdNumber字段重命名为Id。  
`e_if(e_search("content==123"), e_compose(e_drop_fields("age|name"), e_rename("IdNumber", "Id")))`
- 加工结果

```
{
  "Id": 1,
  "content": 123
}
```

### 6.1.4 场景四：转换日志参数类型（v 函数、cn\_int 函数和 dt\_totimestamp 函数）

1. 场景1：调用op\_add函数进行字符拼接和数据相加。

op\_add函数既可以接收字符串类型，也可以接受数值类型，因此不需要做参数类型转换。

示例如下所示：

- 原始日志

```
{
  "a": "1",
  "b": "2"
}
```

- 加工规则

```
e_set("d", op_add(v("a"), v("b")))
e_set("e", op_add(ct_int(v("a")), ct_int(v("b"))))
```

- 加工结果

```
{
  "a": 1,
  "b": 2,
  "d": 12,
  "e": 3
}
```

2. 场景2：运用字段操作函数和ct\_int函数进行类型转换并调用op\_mul函数进行数据相乘。

示例如下所示：

- 原始日志

```
{
  "a": "2",
  "b": "5"
}
```

- 加工规则

```
e_set("c", op_mul(ct_int(v("a")), ct_int(v("b"))))
e_set("d", op_mul(v("a"), ct_int(v("b"))))
```

- 加工结果

```
{
  "a": 2,
  "b": 5,
  "c": 10,
  "d": 22222
}
```

3. 子场景3：调用`dt_parse`函数和`dt_parsetimestamp`函数将字符串或日期时间转换为标准时间。

`dt_totimestamp`函数接收的参数类型为日期时间对象，不是字符串。因此需要调用`dt_parse`函数将`time1`的字符串值类型转化为日期时间对象类型。您也可以直接使用`dt_parsetimestamp`函数，它既能接收日期时间对象，也能接收字符串。

示例如下所示：

- 原始日志

```
{
  "time1": "2019-09-17 9:00:00"
}
```

- 加工规则：将`time1`表示的日期时间转化为Unix时间戳。

```
e_set("time2", dt_totimestamp(dt_parse(v("time1"))))
```

- 加工结果

```
{
  "time1": "2019-09-17 9:00:00",
  "time2": 1568710800
}
```

### 6.1.5 场景五：使用 default 传参为日志不存在的字段填充默认值

- 原始日志

```
{
  "content": "this is a log"
}
```

- 加工规则：

```
e_if(e_not_has("type"), e_set("type", "log"))
```

- 加工结果

```
{
  "type": "log",
  "content": "this is a log"
}
```

### 6.1.6 场景六：使用 e\_if 函数或 e\_switch 函数判断日志并增加字段

推荐使用`e_if`函数或`e_switch`函数进行日志判断。更多信息，请参见[流程控制函数](#)。

- `e_if`函数

```
e_if(条件1, 操作1, 条件2, 操作2, 条件3, 操作3, ....)
```

- `e_switch`函数

`e_switch`函数是条件与操作的配对组合。依次根据条件进行判断，满足条件的进行对应操作。不满足条件的不进行对应操作，直接进行下一个条件判断。如果没有满足任一条件并且配置了`default`参数，则执行`default`配置的操作并返回。

```
e_switch(条件1, 操作1, 条件2, 操作2, 条件3, 操作3, ..., default=None)
```

示例如下所示：

- 原始日志

```
{
  "status1": 200,
  "status2": 404
}
```

- e\_if函数
  - 加工规则

```
e_if(e_match("status1", "200"), e_set("status1_info", "normal"),
e_match("status2", "404"), e_set("status2_info", "error"))
```
  - 加工结果

```
{
  "status1": 200,
  "status2_info": "error",
  "status1_info": "normal",
  "status2": 404
}
```
- e\_switch函数
  - 加工规则

```
e_switch(e_match("status1", "200"), e_set("status1_info", "normal"),
e_match("status2", "404"), e_set("status2_info", "error"))
```
  - 加工结果：只要有一个条件满足，就返回结果，不再进行后续条件判断。

```
{
  "status1": 200,
  "status2_info": "error",
  "status1_info": "normal",
  "status2": 404
}
```

### 6.1.7 场景七：数据转化纳秒级的 Unix 时间戳

部分场景需要云日志服务的数据加工才能够满足纳秒级精度时间戳的需求，当原始日志中存在 Unix 时间格式字段，您可以使用 e\_set 字段操作函数，将其解析成纳秒精度的日志时间。

- 原始日志

```
{
  "source": "1.2.3.4",
  "time": 1704983810,
  "topic": "test",
  "log_time_nano": "1705043680630940602"
}
```
- 加工规则

```
e_set(
  "time", op_div_floor(ct_int(v("log_time_nano")), 1000000000),
)
e_set(
  "time_ns_part", op_mod(ct_int(v("log_time_nano")), 1000000000),
)
```
- 加工结果

```
{
  "time_ns_part": 630940602,
  "log_time_nano": "1705043680630940602",
  "topic": "test",
  "source": "1.2.3.4",
  "time": 1705043680
}
```

### 6.1.8 场景八：数据转化微秒级标准 ISO8601 时间戳

部分场景需要日志服务的数据加工满足高精度时间戳的需求，当原始日志中存在标准 ISO8601 时间格式的字段，您可以使用 e\_set 字段操作函数，将其解析成微秒精度的日志时间。

- 原始日志

```
{
  "source": "1.2.3.4",
```

- ```
"time": 1704983810,  
"topic": "test",  
"log_time": "2024-01-11 23:10:43.992847200"  
}
```
- 加工规则

```
e_set(  
  "time", dt_parsetimestamp(v("log_time"), tz="Asia/Shanghai"), mode="overwrite",  
)  
e_set("tmp_ms", dt_prop(v("log_time"), "microsecond"))  
e_set(  
  "time_ns_part", op_mul(ct_int(v("tmp_ms")), 1000),  
)
```
  - 加工结果

```
{  
  "time_ns_part": 992847000,  
  "tmp_ms": 992847,  
  "topic": "test",  
  "source": "1.2.3.4",  
  "time": 1704985843,  
  "log_time": "2024-01-11 23:10:43.992847200"  
}
```

## 6.2 使用 DSL 加工函数进行事件判断

通过事件判断可以更好地对符合特定条件的数据进行相应操作，让加工逻辑更可靠。本文主要介绍使用函数进行事件判断的常见场景和最佳方案示例。

### 场景一：判断字段是否存在

- 原始日志

```
{  
  "a": "a_value"  
}
```
- 加工规则

```
e_if(e_has("a"),e_set("has_a", true))  
e_if(e_has("b"),e_set("has_b", true))  
e_if(e_not_has("a"),e_set("not_has_a", true))  
e_if(e_not_has("b"),e_set("not_has_b", true))
```
- 加工结果

```
{  
  "a": "a_value",  
  "not_has_b": true,  
  "has_a": true  
}
```

### 场景二：判断字段值是否存在且不为空

- 原始日志

```
{  
  "a": "a_value",  
  "b": ""  
}
```
- 加工规则

```
e_if_else(v("a"), e_set("not_empty_a", true),e_set("not_empty_a", false))  
e_if_else(v("b"), e_set("not_empty_b", true),e_set("not_empty_b", false))
```
- 加工结果

```
{  
  "a": "a_value",  
  "not_empty_b": false,  
  "b": "",  
}
```

```
"not_empty_a": true
}
```

### 场景三：判断字段值是否存在且为空

- 原始日志

```
{
  "a": "a_value",
  "b": ""
}
```

- 加工规则

```
e_if_else(op_and(e_has("a"), op_not(v("a"))), e_set("empty_a", true), e_set("empty_a", false))
e_if_else(op_and(e_has("b"), op_not(v("b"))), e_set("empty_b", true), e_set("empty_b", false))
```

- 加工结果

```
{
  "a": "a_value",
  "b": "",
  "empty_b": true,
  "empty_a": false
}
```

### 场景四：基于字段值的逻辑查询判断

- 原始日志

```
[
  {
    "http_host": "example.com",
    "status": 200,
    "request_method": "GET",
    "scheme": "https",
    "header_length": 700,
    "body_length": 1200
  },
  {
    "http_host": "example.org",
    "status": 200,
    "request_method": "POST",
    "scheme": "https",
    "header_length": 100,
    "body_length": 800
  },
  {
    "http_host": "example.net",
    "status": 200,
    "request_method": "GET",
    "scheme": "http",
    "header_length": 200,
    "body_length": 800
  },
  {
    "http_host": "example.cn",
    "status": 404,
    "request_method": "GET",
    "scheme": "https",
    "header_length": 100,
    "body_length": 300
  }
]
```

- 加工需求1：为所有status字段值为200的日志事件，添加一个新字段type，其值为normal。

加工规则：

```
e_if(e_match("status", "200"), e_set("type", "normal"))
```

加工结果：

```
{
  "scheme": "https",
  "header_length": 700,
  "request_method": "GET",
  "type": "normal",
  "http_host": "example.com",
  "status": 200,
  "body_length": 1200
}
{
  "scheme": "https",
  "header_length": 100,
  "request_method": "POST",
  "type": "normal",
  "http_host": "example.org",
  "status": 200,
  "body_length": 800
}
{
  "scheme": "http",
  "header_length": 200,
  "request_method": "GET",
  "type": "normal",
  "http_host": "example.net",
  "status": 200,
  "body_length": 800
}
{
  "scheme": "https",
  "header_length": 100,
  "request_method": "GET",
  "http_host": "example.cn",
  "status": 404,
  "body_length": 300
}
```

- 加工需求2：为所有status字段值为200，且request\_method字段值为GET，且scheme字段值为https的日志事件添加一个新字段type，其值为normal。

加工规则：

```
e_if(e_match_all("status", "200", "request_method", "GET", "scheme", "https"), e_set("type", "normal"))
```

加工结果：

```
{
  "scheme": "https",
  "header_length": 700,
  "request_method": "GET",
  "type": "normal",
  "http_host": "example.com",
  "status": 200,
  "body_length": 1200
}
{
  "scheme": "https",
  "header_length": 100,
  "request_method": "POST",
  "http_host": "example.org",
  "status": 200,
  "body_length": 800
}
{
  "scheme": "http",
  "header_length": 200,
  "request_method": "GET",
  "http_host": "example.net",
  "status": 200,
  "body_length": 800
}
{
  "scheme": "https",
```

```
"header_length": 100,  
"request_method": "GET",  
"http_host": "example.cn",  
"status": 404,  
"body_length": 300  
}
```

- 加工需求3：为所有status字段值为200，或request\_method字段值为GET，或scheme字段值为https的日志事件添加一个字段type，其值为normal。

加工规则：

```
e_if(e_match_any("status", "200", "request_method", "GET", "scheme", "https"), e_set("type",  
"normal"))
```

加工结果：

```
{  
  "scheme": "https",  
  "header_length": 700,  
  "request_method": "GET",  
  "type": "normal",  
  "http_host": "example.com",  
  "status": 200,  
  "body_length": 1200  
}  
{  
  "scheme": "https",  
  "header_length": 100,  
  "request_method": "POST",  
  "type": "normal",  
  "http_host": "example.org",  
  "status": 200,  
  "body_length": 800  
}  
{  
  "scheme": "http",  
  "header_length": 200,  
  "request_method": "GET",  
  "type": "normal",  
  "http_host": "example.net",  
  "status": 200,  
  "body_length": 800  
}  
{  
  "scheme": "https",  
  "header_length": 100,  
  "request_method": "GET",  
  "type": "normal",  
  "http_host": "example.cn",  
  "status": 404,  
  "body_length": 300  
}
```

## 6.3 使用 DSL 加工函数处理日期时间

处理日期时间，将方便您对日志后续查询与可视化展示。本文档主要介绍使用函数进行日期时间数据类型转换和日期时间偏移。

### 6.3.1 概念解释

LTS DSL语法中的日期时间处理主要涉及三种数据类型：日期时间字符串、日期时间对象和Unix时间戳。

1. 日期时间字符串：日期时间字符串的主要用途是为了便于展示以及提升用户可读性。

LTS DSL语法中的日期时间字符串主要分为两种形式：

- 带有时区信息的日期时间字符串，如2018-06-02 18:41:26+08:00。
- 不带时区信息的日期时间字符串，如2019-06-02 10:41:26。

带有时区信息的日期时间字符串通过在日期时间后添加额外的时差信息来表达时区：

- 2019-06-02 18:41:26+08:00表示该时间是东8区时区下的2019-06-02 18:41:26。
- 2019-06-02 18:41:26-07:00表示该时间是西7区时区下的2019-06-02 18:41:26。

## 2. 日期时间对象

实例化的日期时间，专指Datetime类型的数据。日期时间对象的主要用途是为了便于展示以及提升用户可读性。

## 3. Unix时间戳

从1970年1月1日（UTC/GMT的午夜）开始所经过的秒数。Unix时间戳的主要应用场景有：

表示系统时间：日志事件中表示日志接收时间的字段collectTime字段的值就是使用Unix时间戳来表示对应的系统时间，如下例所示。

```
source: 192.0.2.1
collectTime: 1562741899
topic:
```

时间相关的计算：Unix时间戳是从1970年1月1日开始所经过的秒数，因此在很多场景下便于直接进行日期时间相关的计算，例如如下示例。

- 原始日志

```
{
  "time1": 1562741899,
  "time2": 1562731122
}
```
- 加工规则

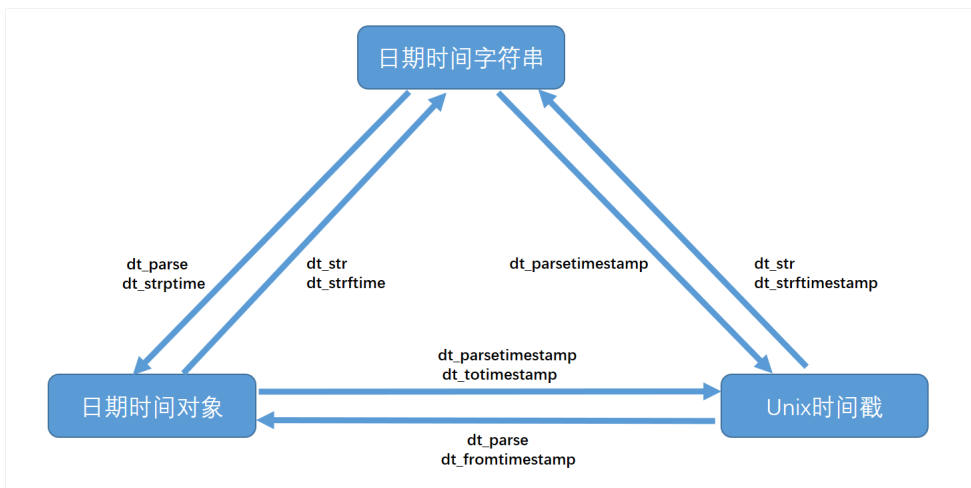
```
e_set("time_diff", op_sub(v("time1"), v("time2")))
```
- 加工结果

```
{
  time1: 1562741899
  time2: 1562731122
  time_diff: 10777
}
```

## 6.3.2 数据类型转换和转换函数

日期时间字符串、日期时间对象和Unix时间戳的相互转换方式和对应转换函数如下图所示。





上图所示的转换场景和对应的转换函数具体描述如下表所示。

| 转换场景                  |                   | 转换函数                                                                                                                                                         |
|-----------------------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日期时间对象和Unix时间戳的相互转换   | 日期时间对象转为Unix时间戳。  | <ul style="list-style-type: none"> <li>dt_parsetimestamp智能转换函数，可以将日期时间对象或日期时间字符串转换为Unix时间戳。</li> <li>dt_totimestamp专用函数，只支持将日期时间对象转换为Unix时间戳。</li> </ul>     |
|                       | Unix时间戳转为日期时间对象。  | <ul style="list-style-type: none"> <li>dt_parse智能转换函数，可以将Unix时间戳或日期时间字符串转换为日期时间对象。</li> <li>dt_fromtimestamp专用函数，只支持将Unix时间戳转换为日期时间对象。</li> </ul>            |
| 日期时间对象和日期时间字符串的相互转换。  | 日期时间对象转为日期时间字符串。  | <ul style="list-style-type: none"> <li>dt_str智能转换函数，可以将日期时间对象、Unix时间戳和日期时间字符串转换为指定格式的日期时间字符串。</li> <li>dt_strftime专用函数，只支持将日期时间对象转换为日期时间字符串。</li> </ul>      |
|                       | 日期时间字符串转为日期时间对象。  | <ul style="list-style-type: none"> <li>dt_parse智能转换函数，可以将日期时间字符串或Unix时间戳转换为日期时间对象。</li> <li>dt_strptime专用函数，只支持将日期时间字符串转化为日期时间对象。</li> </ul>                 |
| 日期时间字符串和Unix时间戳的相互转换。 | 日期时间字符串转为Unix时间戳。 | dt_parsetimestamp智能转换函数，可以将日期时间字符串或日期时间对象转换为Unix时间戳。                                                                                                         |
|                       | Unix时间戳转为日期时间字符串。 | <ul style="list-style-type: none"> <li>dt_str智能转换函数，可以将Unix时间戳、日期时间对象和日期时间字符串转换为指定格式的日期时间字符串。</li> <li>dt_strtimestamp专用函数，只支持将Unix时间戳转换为日期时间字符串。</li> </ul> |

上图和上表展示了三种数据类型之间的六种转换，转换过程涉及两种方式，一种使用智能转换函数，另一种使用该转换的专用函数。

- 智能转换函数  
以dt\_parse函数为代表的智能转换函数可以接收Unix时间戳、日期时间对象以及日期时间字符串等不同类型的参数，实现智能转换。
- 专用函数  
智能转换函数无法满足用户的全部需求。如对于用户自定义的特殊日期格式，dt\_parse等智能转换函数无法自动解析日志，需要使用dt\_strptime函数来进行解析指定格式。

### 6.3.3 日期时间对象和 Unix 时间戳的相互转换

- 处理函数
  - 推荐dt\_parse函数将Unix时间戳转换为日期时间字符串。
  - e\_set函数中的tz参数设置会将不带时区的日期时间对象处理为带时区的，或将原时区的转换为目标时区。
- Unix时间戳转换成带时区的时间字符串对象。
  - 原始日志

```
{
  "time": 1562741899
}
```
  - 加工规则

```
e_set("new_time", dt_parse(v("time"), tz="Asia/Shanghai"))
```
  - 加工结果

```
{
  "new_time": "2019-07-10 14:58:19+08:00",
  "time": 1562741899
}
```

### 6.3.4 日期时间字符串和 Unix 时间戳的相互转换

1. 处理函数
  - dt\_str智能转换函数，可以将Unix时间戳、日期时间对象和日期时间字符串转化为指定格式的日期时间字符串。
  - dt\_strtimestamp函数只支持将Unix时间戳转化为日期时间字符串。
  - dt\_parsetimestamp智能转换函数，可以将日期时间字符串或日期时间对象转换为Unix时间戳。
2. 场景1：不带时区信息的日期时间字符串类型转换为Unix时间戳。  
对于不带时区信息的日期时间字符串如2019-06-02 18:41:26，将日期时间转化为Unix时间戳，需要指定该日期时间的时区，不同的时区转化得到的Unix时间戳的值不一样。
  - 原始日志

```
{
  "time": "2019-07-10 14:58:19"
}
```
  - 加工规则

```
e_set("Shanghai_timestamp", dt_parsetimestamp(v("time"), tz="Asia/Shanghai"))
e_set("Los_Angeles_timestamp", dt_parsetimestamp(v("time"), tz="America/Los_Angeles"))
e_set("UTC_timestamp", dt_parsetimestamp(v("time")))
```
  - 加工结果

- ```
{
  "time": "2019-07-10 14:58:19",
  "Shanghai_timestamp": 1562741899,
  "Los_Angeles_timestamp": 1562795899,
  "UTC_timestamp": 1562770699
}
```
3. 子场景2：自定义的不带时区的特殊日期格式转换成Unix时间戳。
- 原始日志

```
{
  "time1": "2019-07-10 06:58:19",
  "time2": "2019/07/10 06-58-19"
}
```
  - 加工规则

```
e_set("time3", dt_parsetimestamp(v("time1")))
e_set("time4", dt_parsetimestamp(dt_strptime(v("time2"), "%Y/%m/%d %H-%M-%S")))
```
  - 加工结果

```
{
  "time1": "2019-07-10 06:58:19",
  "time2": "2019/07/10 06-58-19",
  "time3": 1562741899,
  "time4": 1562741899
}
```

### 6.3.5 日期时间对象和日期时间字符串的相互转换

1. 处理函数
  - dt\_parse智能转换函数可以将日期时间字符串或Unix时间戳转换为日期时间对象。
  - dt\_astimezone函数返回一个带新时区信息的日期时间对象。
2. 场景1：不带时区信息的日期时间字符串转换成指定时区的日期时间对象。

对于不带时区信息的日期时间字符串2019-06-02 18:41:26，可以通过Unix时间戳，实现不同时区下的日期时间的相互转换。将洛杉矶时区的日期时间转换为上海时区的日期时间。

  - 原始日志：已知time字段的值的时间是洛杉矶时间

```
{
  "time": "2019-06-04 2:41:26"
}
```
  - 加工规则

```
e_set("timestamp", dt_parsetimestamp(v("time"), tz="America/Los_Angeles"))
e_set("Shanghai_time", dt_parse(v("timestamp"), tz="Asia/Shanghai"))
```
  - 加工结果

```
{
  "Shanghai_time": "2019-06-04 17:41:26+08:00",
  "time": "2019-06-04 2:41:26",
  "timestamp": 1559641286
}
```
3. 场景2：不带时区的日期时间字符串转换成带时区的日期时间对象。
  - 原始日志

```
{
  "time": "2019-07-10 06:58:19"
}
```
  - 加工规则

```
e_set("new_time", dt_parse(v("time"), tz="Asia/Shanghai"))
```

- 加工结果

```
{
  "new_time": "2019-07-10 06:58:19+08:00",
  "time": "2019-07-10 06:58:19"
}
```
- 4. 场景3：带时区的日期时间字符串转换为目标时区的日期时间对象。
  - 原始日志

```
{
  "time" : "2019-06-04 2:41:26+08:00"
}
```
  - 加工规则

```
e_set("new_time",dt_astimezone(v("time"), "America/Los_Angeles"))
```
  - 加工结果

```
{
  "new_time": "2019-06-03 11:41:26-07:00",
  "time": "2019-06-04 2:41:26+08:00"
}
```

## 6.3.6 日期时间偏移

1. 处理函数
  - dt\_add函数的参数如下：

```
dt_add(字段名, 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, weeks(s)=None,
weekday=None)
```

year(s)、month(s)、day(s)等参数的后面都带有s，表示这些参数可以有两种形式，即year和years，month和months等。以year和years为例，如果参数传递的是year，表示在年份粒度上覆盖为year参数的值；如果传递的是years，表示在年份粒度上增加years参数的值。同时要一起组合使用的dt\_add函数支持在特定时间粒度上修改（增加、减少、覆盖）日期时间的值。
  - dt\_add中weekday参数通常和dt\_MO、dt\_TU等参数一起使用，表示特定星期几的偏移，如下例所示。
2. 场景1：按年和月进行日期偏移。
  - 原始日志

```
{
  "time1" : "2019-06-04 2:41:26"
}
```
  - 加工规则1

```
e_set("time2", dt_add(v("time1"), year=2018))
```
  - 加工结果1

```
{
  "time1": "2019-06-04 2:41:26",
  "time2": "2018-06-04 02:41:26"
}
```
  - 加工规则2

```
e_set("time2", dt_add(v("time1"), years=2018))
```
  - 加工结果2

```
{
  "time1": "2019-06-04 2:41:26",
  "time2": "4037-06-04 02:41:26"
}
```
3. 场景2：按周进行日期偏移。
  - 原始日志：2019-06-04是周二

```
{
  "time1" : "2019-06-04 2:41:26"
}

- 加工规则
#time1的下一个星期一对应的日期
e_set("nex_Monday", dt_add(v("time1"), weekday=dt_MO(1)))

#time1的上一个星期二对应的日期
e_set("previous_Tuesday", dt_add(v("time1"), weekday=dt_TU(op_neg(1))))

#time1的下下一个星期六对应的日期
e_set("nex_next_Saturday", dt_add(v("time1"), weekday=dt_SA(2)))

#time1的上上一个星期日对应的日期
e_set("previous_previous_Sunday", dt_add(v("time1"), weekday=dt_SU(op_neg(2))))

- 加工结果
{
  "time1": "2019-06-04 2:41:26",
  "previous_Tuesday": "2019-05-28 02:41:26",
  "previous_previous_Sunday": "2019-05-26 02:41:26",
  "nex_next_Saturday": "2019-06-15 02:41:26",
  "nex_Monday": "2019-06-10 02:41:26"
}
```

## 6.4 使用 DSL 加工函数对 LTS 日志数据脱敏

数据脱敏可以有效地减少敏感数据在加工、传输、使用等环节中的暴露，降低敏感数据泄露的风险，保护用户权益。本文介绍日志服务数据加工过程中常见的脱敏场景、对应的脱敏方法及示例。

### 背景信息

使用敏感数据包括手机号、银行卡号、邮箱、IP地址、AK、身份证号、网址、订单号、字符串等场景中，您需要为敏感数据进行脱敏操作。在日志服务数据加工服务中，常见的脱敏方法有正则表达式替换（关键函数`regex_replace`）、Base64转码（关键函数`base64_encoding`）、MD5编码（关键函数`md5_encoding`）、`str_translate`映射（关键函数`str_translate`）等。更多信息，请参见[正则表达式函数](#)和[编码解码函数](#)。

### 场景 1：手机号脱敏

日志中包含不希望被暴露的手机号，可采用正则表达式，运用`regex_replace`函数脱敏。参考如下示例：

- 原始日志

```
{
  "iphone": "13900001234"
}
```
- 加工规则

```
e_set(
  "sec_iphone",
  regex_replace(v("iphone"), r"(\d{0,3})\d{4}(\d{4})", replace=r"1****\2"),
)
```
- 加工结果

```
{
  "sec_iphone": "139****1234",
  "iphone": "13900001234"
}
```

## 场景 2：银行卡信息脱敏

日志中包含银行卡或者信用卡信息，可采用正则表达式，运用`regex_replace`函数脱敏。

- 原始日志

```
{
  "content": "bank number is 491648411333978312 and credit card number is 4916484113339780"
}
```

- 加工规则

```
e_set(
  "bank_number",
  regex_replace(
    v("content"), r"([1-9]{1})(\d{14})(\d{13})(\d{11})(\d{4})", replace=r"****\3"
  ),
)
```

- 加工结果

```
{
  "bank_number": "bank number is ****8312 and credit card number is ****9780",
  "content": "bank number is 491648411333978312 and credit card number is 4916484113339780"
}
```

## 场景 3：邮箱地址脱敏

日志中包含邮箱信息，可采用正则表达式，运用`regex_replace`函数脱敏。

- 原始日志

```
{
  "content": "email is username@example.com"
}
```

- 加工规则

```
e_set(
  "email_encrypt",
  regex_replace(
    v("content"),
    r"[A-Za-z\d]+([_-][A-Za-z\d]+)*(@([A-Za-z\d]+[-.])+[A-Za-z\d]{2,4})",
    replace=r"****\2",
  ),
)
```

- 加工结果

```
{
  "content": "email is username@example.com",
  "email_encrypt": "email is ****@example.com"
}
```

## 场景 4：AK 脱敏

日志中包含AccessKey信息，可采用正则表达式，应用`regex_replace`函数。

- 原始日志

```
{
  "content": "ak id is <testAccessKey ID> and ak key is <testAccessKey Secret>"
}
```

- 加工规则

```
e_set(
  "akid_encrypt",
  regex_replace(
    v("content"),
    r"([a-zA-Z0-9]{4})((([a-zA-Z0-9]{26})|([a-zA-Z0-9]{12})))",
    replace=r"\1****",
  ),
)
```

- 加工结果

```
{
  "akid_encrypt": "ak id is jdhc**** and ak key is Jkde****",
  "content": "ak id is <testAccessKey ID> and ak key is <testAccessKey Secret>"
}
```

## 场景 5：IP 地址脱敏

日志中包含IP地址信息，可同时运用regex\_replace函数，对IP地址进行正则捕获后而脱敏。

- 原始日志

```
{
  "content": "ip is 192.0.2.10"
}
```

- 加工规则

```
e_set("ip_encrypt", regex_replace(v("content"), r"((25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])\.){3}(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])", replace=r"****"))
```

- 加工结果

```
{
  "content": "ip is 2.0.2.10",
  "ip_encrypt": "ip is ****"
}
```

## 场景 6：身份证脱敏

日志中包含身份证信息，可同时运用regex\_replace函数，对身份证号进行正则捕获后而脱敏。

- 原始日志

```
content: Id card is 111222190002309999
```

- 加工规则

```
e_set(
  "id_encrypt", regex_replace(v("content"), r"\b\d{17}(\d|X)\b", replace=r"\1****")
)
```

- 加工结果

```
{
  "id_encrypt": "Id card is 9****",
  "content": "Id card is 111222190002309999"
}
```

## 场景 7：网址脱敏

对日志内容中的网址做脱敏处理，并且将脱敏的数据转成明文格式，可运用Base64编码解码函数，对网址进行转码。

- 原始日志

```
{
  "content": "https://www.huaweicloud.com/"
}
```

- 加工规则

```
e_set("base64_url", base64_encoding(v("content")))
```

- 加工结果

```
{
  "base64_url": "aHR0cHM6Ly93d3cuaHVhd2VpY2xvdWQuY29tLw==",
  "content": "https://www.huaweicloud.com/"
}
```

## 📖 说明

如果想对base64\_url进行解码，可以使用base64\_decoding(v("base64\_url"))DSL语法规则。

## 场景 8：订单号脱敏

对日志内容中的订单号做脱敏处理，同时不希望其他人能够解码，可运用MD5编码函数，对订单号进行编码。

- 原始日志

```
{
  "orderId": "20210101123456"
}
```

- 加工规则

```
e_set("md5_orderId",md5_encoding(v("orderId")))
```

- 加工结果

```
{
  "orderId": 20210101123456,
  "md5_orderId": "9c0ab8e4d9f4eb6fbd5c508bbca05951"
}
```

## 场景 9：字符串脱敏

若希望日志中的关键字字符串不被暴露，可通过str\_translate函数制定映射规则，对关键字或字符串进行映射脱敏。

- 原始日志

```
{
  "content": "message level is info_"
}
```

- 加工规则

```
e_set("data_translate", str_translate(v("content"),"aeiou","12345"))
```

- 加工结果

```
{
  "data_translate": "m2ss1g2 l2v2l 3s 3nf4_",
  "content": "message level is info_"
}
```

## 6.5 文本解析

### 6.5.1 解析 Nginx 日志

Nginx访问日志记录了用户访问的详细信息，解析Nginx访问日志对业务运维具有重要意义。本文介绍如何使用正则表达式函数解析Nginx访问日志。

现以一条Nginx成功访问日志为例，介绍如何使用正则表达式解析Nginx成功访问日志。

- 原始日志

```
{"source":"192.168.0.1",
"client_ip":"192.168.254.254",
"receive_time":"1563443076",
"content":"192.168.0.2 - - [04/Jan/2019:16:06:38 +0800] \"GET http://example.aliyundoc.com/_astats?application=&inf.name=eth0 HTTP/1.1\" 200 273932 \"-\" \"Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.example.com/bot.html)\""}
}
```



- 解析需求
  - 需求1：从Nginx日志中提取出code、ip、datetime、protocol、request、sendbytes、referer、useragent、verb信息。
  - 需求2：对request进行再提取，提取出uri\_proto、uri\_domain、uri\_param信息。
  - 需求3：对解析出来的uri\_param进行再提取，提取出uri\_path、uri\_query信息。

● 加工规则

- 总编排

```

"""第一步：初步解析Nginx日志"""
e_regex("content",r'(?P<ip>\d+\.\d+\.\d+\.\d+)( - - \[ \] (?P<datetime>[\s\S]+)) \'(?P<verb>[A-Z]+) (?P<request>[\S]*) (?P<protocol>[\S]+) \[\' (?P<code>\d+) (?P<sendbytes>\d+) \[\' (?P<refere>[\S]*) \[\' \[\' (?P<useragent>[\S\S]+) \[\' \[\'
"""第二步：解析第一步得到的request"""
e_regex('request',r'(?P<uri_proto>(\w+)):\[\'/(?P<uri_domain>[a-z0-9]*[^\[\'/]) (?P<uri_param>(.*?)$)')
"""第三步：解析第二步得到的uri_param参数"""
e_regex('uri_param',r'(?P<uri_path>\[\'[a-z]+[^\[\'/])? (?P<uri_query>[^\[\'/]+)')
    
```

- 总编排加工结果

```

{
  "request": "http://example.aliyundoc.com/_astats?application=&inf.name=eth0",
  "refere": "-",
  "uri_proto": "http",
  "code": 200,
  "ip": "192.168.0.2",
  "sendbytes": 273932,
  "receive_time": 1563443076,
  "verb": "GET",
  "useragent": "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.example.com/bot.html)",
  "source": "192.168.0.1",
  "content": "192.168.0.2 - - [04/Jan/2019:16:06:38 +0800] \"GET http://example.aliyundoc.com/_astats?application=&inf.name=eth0 HTTP/1.1\" 200 273932 \"-\" \"Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.example.com/bot.html)\"",
  "datetime": "04/Jan/2019:16:06:38 +0800",
  "protocol": "HTTP/1.1",
  "uri_path": "/_astats",
  "uri_query": "application=&inf.name=eth0",
  "uri_param": "/_astats?application=&inf.name=eth0",
  "client_ip": "192.168.254.254",
  "uri_domain": "example.aliyundoc.com"
}
    
```

- 细分编排及对应加工结果

■ 针对需求1的加工编排如下：

```

e_regex("content",r'(?P<ip>\d+\.\d+\.\d+\.\d+)( - - \[ \] (?P<datetime>[\s\S]+)) \'(?P<verb>[A-Z]+) (?P<request>[\S]*) (?P<protocol>[\S]+) \[\' (?P<code>\d+) (?P<sendbytes>\d+) \[\' (?P<refere>[\S]*) \[\' \[\' (?P<useragent>[\S\S]+) \[\' \[\'
    
```

对应加工结果：

```

{
  "request": "http://example.aliyundoc.com/_astats?application=&inf.name=eth0",
  "refere": "-",
  "code": 200,
  "ip": "192.168.0.2",
  "sendbytes": 273932,
  "receive_time": 1563443076,
  "verb": "GET",
  "useragent": "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.example.com/bot.html)",
  "source": "192.168.0.1",
  "content": "192.168.0.2 - - [04/Jan/2019:16:06:38 +0800] \"GET http://example.aliyundoc.com/_astats?application=&inf.name=eth0 HTTP/1.1\" 200 273932 \"-\" \"Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.example.com/bot.html)\"",
  "datetime": "04/Jan/2019:16:06:38 +0800",
}
    
```

```
"protocol": "HTTP/1.1",
"client_ip": "192.168.254.254"
}
```

- 针对需求2解析request，加工编排如下：

```
e_regex('request',r'(?P<uri_proto>(\w+)):\/(?P<uri_domain>[a-z0-9.]*[^\/])?(?P<uri_param>(.*?)$)')
```

对应加工结果：

```
{
  "uri_proto": "http",
  "uri_param": "/_astats?application=&inf.name=eth0",
  "uri_domain": "example.aliyundoc.com"
}
```

- 针对需求3解析uri\_param，加工编排如下：

```
e_regex('uri_param',r'(?P<uri_path>\/[a-z]+[^\?]+)\?(?P<uri_query>[^\?]+)')
```

对应加工结果：

```
{
  "uri_path": "/_astats",
  "uri_query": "application=&inf.name=eth0",
}
```

## 6.5.2 提取字符串动态键值对

本文档介绍如何使用不同方案提取字符串键值对。

### 常用方案比较

字符串动态键值对提取分为关键字提取、值提取、关键字加工和值加工，常用方案为采用e\_kv函数、e\_kv\_delimit函数和e\_regex函数等。不同提取场景的三种方案如下：

方案	关键字提取	值提取	关键字加工	值加工
e_kv	使用特定正则表达式	支持默认的字符集、特定分隔符或者带(、)或(")分隔	支持前后缀	支持文本escape
e_kv_delimit	使用特定正则表达式	使用分隔符	支持前后缀	默认无
e_regex	组合自定义正则表达式和默认字符集过滤	完全自定义	自定义	自定义

大部分键值对的提取使用e\_kv函数并配置特定参数就可以很好地满足，尤其是带括字符和反斜杠需要提取并转义时。其他复杂或高级的场景可以用e\_regex函数来提取。部分特定场景下的键值对使用e\_kv\_delimit函数会更简单。

### 关键字提取

1. 示例1

以k1: q=asd&a=1&b=2&\_1\_=3日志为例，如果要对该格式的日志作关键字和值提取，三种方案如下：

- e\_kv函数

■ 原始日志

```
{
  "k1": "q=asd&a=1&b=2&_1_=3"
}
```

■ 加工规则

```
e_kv("k1")
```

■ 加工结果

```
{
  "q": "asd",
  "a": 1,
  "b": 2,
  "k1": "q=asd&a=1&b=2&_1_=3",
  "_1_": 3
}
```

- e\_kv\_delimit函数

■ 原始日志

```
{
  "k1": "q=asd&a=1&b=2&_1_=3"
}
```

■ 加工规则

```
# 以&分隔键值后,用&分隔提取出关键字
e_kv_delimit("k1", pair_sep="&")
```

■ 加工结果

```
{
  "q": "asd",
  "a": 1,
  "b": 2,
  "k1": "q=asd&a=1&b=2&_1_=3",
  "_1_": 3
}
```

- e\_regex函数

■ 原始日志

```
{
  "k1": "q=asd&a=1&b=2&_1_=3"
}
```

■ 加工规则

```
# 自行指定字符集提取关键字和值
e_regex("k1", r"(\w+)=(\[a-zA-Z0-9+\]),{r\"1\": r\"2\"}")
```

■ 加工结果

```
{
  "q": "asd",
  "a": 1,
  "b": 2,
  "k1": "q=asd&a=1&b=2&_1_=3",
  "_1_": 3
}
```

2. 示例2

以content:k1=v1&k2=v2?k3:v3为例，需要特定正则提取关键字，方案如下：

- e\_kv\_delimit函数

- 原始日志

```
{
  "content": "k1=v1&k2=v2?k3:v3"
}
```
- 加工规则

```
e_kv_delimit("content", pair_sep=r"&?", kv_sep="(?:|=|:)")
```
- 加工结果

```
{
  "k1": "v1",
  "k2": "v2",
  "k3": "v3",
  "content": "k1=v1&k2=v2?k3:v3"
}
```
- e\_regex函数
  - 原始日志

```
{
  "content": "k1=v1&k2=v2?k3:v3"
}
```
  - 加工规则

```
e_regex("content", r"([a-zA-Z0-9]+)[=:]([a-zA-Z0-9]+)", {r"\1": r"\2"})
```
  - 加工结果

```
{
  "k1": "v1",
  "k2": "v2",
  "k3": "v3",
  "content": "k1=v1&k2=v2?k3:v3"
}
```

## 值提取

- 动态键值对之间以及关键字与值之间有明确标识，如a=b或a="cxxx"日志格式的，推荐用e\_kv函数，示例如下。
  - 原始日志

```
{
  "content": "k=\"helloworld\",the change world, k2=\"good\""
}
```
  - 加工规则  
这种情况下使用e\_kv函数，提取内容不包括the change world:

```
e_kv("content")
# e_regex函数写法
e_regex("content", r"(\w+)=\"(\w+)\"", {r"\1": r"\2"})
```
  - 加工结果：提取后的日志为：

```
{
  "k2": "good",
  "k": "helloworld",
  "content": "k=\"helloworld\",the change world, k2=\"good\""
}
```
- 对带"的日志格式content:k1="v1=1"&k2=v2?k3=v3，推荐使用e\_kv函数进行提取，例如：
  - 原始日志

```
{
  "content": "k1=\"v1=1\"&k2=v2?k3=v3"
}
```
  - 加工规则

```
e_kv("content",sep="=", quote="")
```

#### - 加工结果

处理后日志为：

```
{
  "k1": "v1=1",
  "k2": "v2",
  "k3": "v3",
  "content": "k1=\v1=1\&k2=v2?k3=v3"
}
```

## 关键字加工

- e\_kv函数和e\_kv\_delimit函数都可以通过prefix="", suffix=""对关键字和值进行加工。

#### - 原始日志

```
{
  "content": "q=asd&a=1&b=2"
}
```

#### - 加工规则(各语句分开执行，功能相同)

```
e_kv("content", sep="=", quote="", prefix="start_", suffix="_end")
e_kv_delimit("content", pair_sep=r"&", kv_sep="=", prefix="start_", suffix="_end")
e_regex("content", r"(\w+)=(\w+)", {r"start_\1_end": r"\2"})
```

#### - 加工结果

加工后的数据都是关键字加工形式，如下：

```
{
  "start_b_end": 2,
  "start_a_end": 1,
  "start_q_end": "asd",
  "content": "q=asd&a=1&b=2"
}
```

- e\_regex函数对关键字加工的能力更强，例如：

#### - 加工规则

```
e_regex("content", r"(\w+)=(\w+)", {r"\1": r"\2"})
```

#### - 加工结果

加工后的数据都是关键字加工形式，如下：

```
{
  "q_q": "asd",
  "a_a": 1,
  "b_b": 2,
  "content": "q=asd&a=1&b=2"
}
```

## 值加工

日志格式为k1:"v1\"abc"形式，同时值加工的内容存在有双引号符号的情形，使用e\_kv函数可正常进行提取。

- 原始日志

```
""""
这里的\只是普通的符号，不是转义符
""""
```

```
{
  "content": "k1:\v1\\\"abc\", k2:\v2\", k3: \"v3\""
}
```

- 加工规则：

```
e_kv("content",sep=":", quote="")
```

- 加工结果

提取后的日志为：

```
{
  "k1": "v1\\",
  "k2": "v2",
  "k3": "v3",
  "content": "k1:\\v1\\\\"abc\\", k2:\\v2\\", k3: \\v3\\""}
}
```

## 6.6 数据富化

### 6.6.1 从 OBS 获取 csv 文件进行数据富化

本文档介绍如何通过资源函数和映射富化函数从OBS中获取数据对日志数据进行富化。

#### 前提条件

已上传csv文件到OBS。

#### 背景信息

OBS是华为云提供的一个高可靠、高性能、高安全的基于对象的海量存储云服务。针对更新不频繁的数据，建议您存储在OBS上，只需要支付少量的存储费用即可。当您分散存储数据，面临日志数据不完善时，您可以从OBS中获取数据。日志服务数据加工支持使用`res_obs_file`函数从OSS中获取数据，再使用`tab_parse_csv`函数构建表格，最后使用`e_table_map`函数进行字段匹配，返回指定字段和字段值，生成新的日志数据。

#### 实践案例

- 原始日志

```
{
  "account": "Sf24asc4ladDS"
}
```

- OBS Bucket中的CSV文件数据

表 6-1 CSV 文件数据

id	account	nickname
1	Sf24asc4ladDS	多弗朗明哥
2	Sf24asc4ladSA	凯多
3	Sf24asc4ladCD	罗杰

#### 📖 说明

CSV文件需要是UNIX文件格式，windows文件格式暂不支持。

- 加工规则

通过原始日志中的account字段和OBS CSV文件中的account字段进行匹配，只有account字段的值完全相同，才能匹配成功。匹配成功后，返回OBS CSV文件中的nickname字段和字段值，与原始日志中的数据拼接，生成新的数据。

```
e_table_map(tab_parse_csv(res_obs_file("https://obs.xxx.myhuaweicloud.com", "dsl-test-xx", "data.csv")), "account", "nickname")
```

res\_obs\_file函数重要字段说明如下表所示，更多参数说明请参见[资源函数](#)。

字段	说明
endpoint	obs访问域名，详情请参见res_obs_file。
bucket	用于存储CSV文件的OBS Bucket。
file	目标OBS文件的路径。例如test/data.txt，不能以正斜线(/)开头。

- 加工结果

```
{  
  "nickname": "多弗朗明哥",  
  "account": "Sf24asc4ladDS"  
}
```

## 6.6.2 使用 e\_dict\_map、e\_search\_dict\_map 函数进行数据富化

本文介绍使用映射富化函数e\_dict\_map、e\_search\_dict\_map进行数据富化的实践案例。

### 背景信息

日志服务数据加工映射富化函数包括普通映射函数和搜索映射函数，两者区别如下所示：

- 普通映射函数使用文本完全匹配方式来映射。普通映射函数包括e\_dict\_map函数和e\_table\_map函数，两者区别在于e\_dict\_map函数接收的是dict类型的数据，e\_table\_map函数接收的是通过资源函数获取的table类型的数据。

例如：在nginx日志中，将特定的状态码转换为文本格式，可以使用普通映射函数e\_dict\_map。

状态码	文本
200	成功
300	跳转
400	请求错误
500	服务器错误

- 搜索映射函数的映射关键字是查询字符串，支持正则表达式匹配、完全匹配、模糊匹配等形式。搜索映射函数包括e\_search\_dict\_map函数和e\_search\_table\_map函数，两者区别在于e\_search\_dict\_map函数接收的是dict类型的数据，而e\_search\_table\_map函数接收的是通过资源函数获取的table类型的数据。

例如：在nginx日志中，将一定范围内的状态码转换为文本格式，可以使用搜索映射函数e\_search\_dict\_map。

状态码	文本
2XX	成功
3XX	跳转
4XX	请求错误
5XX	服务器错误

## 使用 e\_dict\_map 函数进行数据富化

本案例介绍使用e\_dict\_map函数完成数据富化的方法。

- 原始日志

```
[[{"http_host": "example.com", "http_status": 300, "request_method": "GET"}, {"http_host": "example.org", "http_status": 200, "request_method": "POST"}, {"http_host": "example.net", "http_status": 400, "request_method": "GET"}, {"http_host": "huaweicloud.com", "http_status": 500, "request_method": "GET"}]]
```

- 加工需求

将http\_status字段中的请求状态码转化为文本格式，并添加到status\_desc字段中。

- 加工规则

```
e_dict_map({"400": "请求错误", "500": "服务器错误", "300": "跳转", "200": "成功"}, "http_status", "status_desc")
```

### 说明

在实际情况下，HTTP请求状态码不止以上4种，详情请参见[HTTP请求状态码](#)。当http\_status字段的值为401、404时，需要更新字典覆盖，否则无法匹配。

- 加工结果

```
{ "status_desc": "跳转", "http_status": 300, "request_method": "GET", "http_host": "example.com" }, { "status_desc": "成功", "http_status": 200, "request_method": "POST", "http_host": "example.org" }, { "status_desc": "请求错误",
```



```
"http_status": 400,  
"request_method": "GET",  
"http_host": "example.net"  
}  
{  
"status_desc": "服务器错误",  
"http_status": 500,  
"request_method": "GET",  
"http_host": "huaweicloud.com"  
}
```

## 使用 e\_search\_dict\_map 函数进行数据富化

本案例介绍使用e\_search\_dict\_map函数完成数据富化的方法。

- 原始日志

```
{  
"http_host": "example.com",  
"http_status": 200,  
"request_method": "GET"  
},  
{  
"http_host": "example.org",  
"http_status": 201,  
"request_method": "POST"  
},  
{  
"http_host": "example.net",  
"http_status": 404,  
"request_method": "GET"  
}
```

- 加工需求

根据日志中的http\_status字段的值的不同，为每条日志添加不同的type信息。

- 为http\_status为2XX的日志，添加type字段，并将字段值设置为正常。
- 为http\_status为3XX的日志，添加type字段，并将字段值设置为重定向。
- 为http\_status为4XX的日志，添加type字段，并将字段值设置为错误。

- 加工规则

```
e_search_dict_map({"http_status:2??": "正常","http_status:3??": "重定向","http_status:4??": "错误"},  
"http_status", "type")
```

- 加工结果

```
{  
"http_status": "正常",  
"request_method": "GET",  
"http_host": "example.com"  
}  
{  
"http_status": "正常",  
"request_method": "POST",  
"http_host": "example.org"  
}  
{  
"http_status": "错误",  
"request_method": "GET",  
"http_host": "example.net"  
}
```

### 6.6.3 构建字典与表格进行数据富化

字典和表格是对数据进行富化时主要使用的两种数据结构，本文档主要介绍这两种数据结构的常见构建方式，并对比不同构建方式的优缺点。

## 字典构建

不同字典构建方式对比参考如下：

表 6-2 不同字典构建方式对比

构建方式	优点	缺点
直接构建	直观、简单、方便。	如果内容较多，规则会相对冗长。且静态不灵活。
从任务配置资源构建	内容较多且经常修改时推荐使用，易于维护。	不易于扩展和跨任务复用，不支持自动刷新。
从表格构建	高级场景下使用，维护机制更灵活。	需要构建和维护对应的表格，过程相对繁琐。
从字典函数构建	基于逻辑动态构建字典，特定场景下适用。	较为高级，不易于维护。
从其他表达式构建	从日志事件的字段中动态提取映射关系，特定场景下适用。	较为高级，不易于维护。

- 直接构建

```
e_dict_map({"400": "error", "200": "ok", "*": "other"}, "status", "message")
```

- 从任务高级配置构建

```
e_dict_map(res_local("http_code_map"), "status", "message")
```

其中http\_code\_map是任务高级配置项，值为：



- 从表格构建

使用tab\_to\_dict从表格构建。而表格的构建参见本文后面的表格构建。

```
e_dict_map(tab_to_dict(tab_parse_csv("status_code,status_info\n400,error\n200,ok\n*", "other"), "status_code", "status_info"), "status", "message")
```

- 从字典函数构建

```
e_dict_map(dct_make("400", "error", "200", "ok", "*", "other"), "status", "message")
```

- 从其他表达式构建

```
e_dict_map(json_parse(v("http_code_map")), "status", "message")
```

此处从源日志的http\_code\_map字段中获取映射关系。

## 表格构建

不同表格构建方式对比参考如下：

表 6-3 不同表格构建方式对比

构建方式	优点	缺点
从文本构建	直观、简单、方便。	如果内容较多，规则会相对冗长。不易于维护、扩展和复用。
从OBS资源构建	内容较多且不常修改时推荐使用，易于维护。	编写相对复杂。

- 从文本构建

```
e_table_map(tab_parse_csv("city,name,age\nshanghai,baixiao,10\ncity:nanjing,Maki,18"), "name", ["city", "age"])
```

- 从OBS资源构建

```
e_search_table_map(tab_parse_csv(res_obs_file("https://obs.xxx.myhuaweicloud.com","dsl-test-xx","data.csv")), "name", ["city", "age"])
```

其中data.csv是obs中的文件，值为：

```
e_search_table_map(tab_parse_csv(res_obs_file("https://obs.xxx.myhuaweicloud.com","dsl-test-xx","data.csv")), "name", ["city", "age"])
```