阿里云
日志服务

用户指南

文档版本：20180925

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告：** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明：** 您也可以通过按**Ctrl** + **A**选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

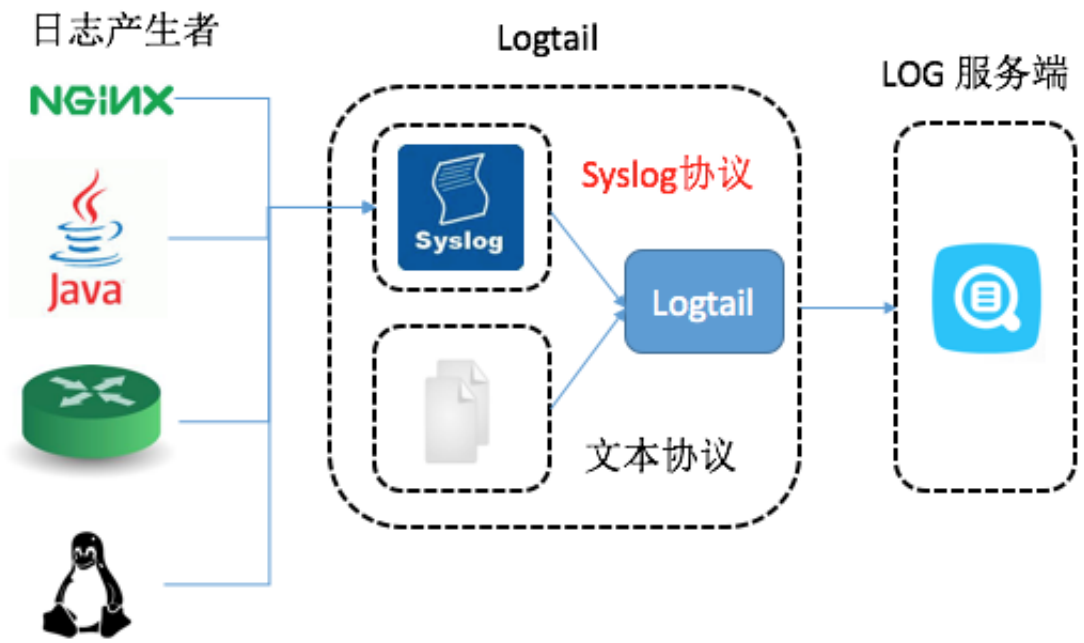
法律声明.....I
通用约定.....I
1 heng隐藏文件夹.....1
1.1 Syslog-采集参考.....1
1.2 Syslog.....6
2 Logtail采集.....11
2.1 简介.....11
2.2 选择网络.....15
2.3 数据源.....18
2.3.1 文本日志.....18
2.3.2 文本-配置解析.....28
2.3.3 文本-配置时间格式.....30
2.3.4 文本-导入历史日志文件.....32
2.3.5 文本-生成主题.....35
2.3.6 自定义插件.....37
2.3.7 插件-MySQL Binlog方式.....38
2.3.8 插件-JDBC查询结果.....47
2.3.9 插件 HTTP方式.....52
2.3.10 插件-Beats和Logstash输入源.....56
2.3.11 插件-Syslog输入源.....60
2.3.12 插件-处理采集数据.....66
2.3.13 容器-文本日志.....80
2.3.14 容器-标准输出.....88
2.3.15 Kubernetes-CRD配置日志采集.....97
2.4 相关限制说明.....106
3 索引与查询.....110
3.1 简介.....110
3.2 索引数据类型.....113
3.2.1 文本类型.....114
3.2.2 Json类型.....115
3.2.3 数值类型.....116
3.3 查询语法.....117
3.4 上下文查询.....122
3.5 其他功能.....125
3.6 快速查询.....129

1 heng隐藏文件夹

1.1 Syslog-采集参考

Logtail目前支持的接入端为syslog和文本文件，如下图所示：

图 **1-1: Logtail**支持的接入端



Logtail通过TCP协议支持syslog。配置Logtail采集syslog日志详细步骤请参见*Syslog*通过Logtail采集syslog日志。

syslog优势

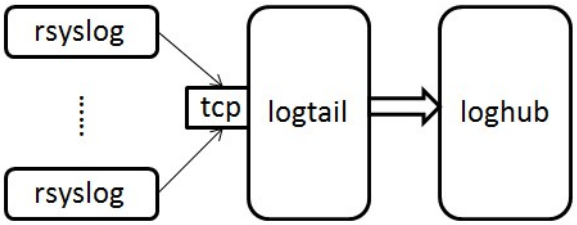
syslog概念请参考 *syslog*。

和利用文本文件相比，使用syslog时日志数据直接收集到LogHub，syslog不落盘、保密性好。免去了文件落盘和解析的代价，单机可达 80MB/S 吞吐率。

基本原理

Logtail 支持在本地配置TCP端口，接收syslog Agent转发的日志。Logtail开启TCP端口，接收rsyslog或者其他syslog Agent通过TCP协议转发过来的syslog数据，Logtail解析接收到的数据并转发到LogHub中。配置Logtail采集syslog日志过程请参见*Syslog*。Logtail、syslog、LogHub三者之间的关系如下图所示。

图 **1-2:** 基本原理



syslog日志格式

Logtail 通过 TCP 端口接收到的数据是流式的，如果要从流式的数据中解析出一条条的日志，日志格式必须满足以下条件：

- 每条日志之间使用换行符（`\n`）分隔，一条日志内部不可以出现换行符。
- 日志内部除了消息正文可以包含空格，其他字段不可以包含空格。

syslog日志格式如下：

```
$version $tag $unixtimestamp $ip [$user-defined-field-1 $user-defined-field-2 $user-defined-field-n] $msg\n"
```

各个字段含义为：

日志字段	含义
version	该日志格式的版本号，Logtail使用该版本号解析user-defined-field 字段。
tag	数据标签，用于寻找Project或Logstore，不可以包含空格和换行符。
unixtimestamp：	该条日志的时间戳。
ip	该条日志的对应的机器IP，如果日志中的该字段是 127.0.0.1，最终发往服务端的日志数据中该字段会被替换成TCP socket的对端地址。
user-defined-field	用户自定义字段，中括号表示是可选字段，可以有 0 个或多个，不可以包含空格和换行符。
msg	日志消息正文，不可以包含换行符，末尾的 `\n` 表示换行符。

以下示例日志即为满足格式要求的日志：

```
2.1 streamlog_tag 1455776661 10.101.166.127 ERROR com.alibaba.
streamlog.App.main(App.java:17) connection refused, retry
```

另外，不仅 syslog 日志可以接入Logtail，任何日志工具只要能满足以下条件，都可以接入：

- 可以将日志格式化，格式化之后的日志格式满足格式要求。
- 可以通过 TCP 协议将日志 append 到远端。

Logtail解析syslog日志规则

Logtail 需要增加配置以解析syslog日志。例如：

```
"streamlog_formats":
[
    {"version": "2.1", "fields": ["level", "method"]},
    {"version": "2.2", "fields": []},
    {"version": "2.3", "fields": ["pri-text", "app-name", "syslogtag
"]}
]
```

Logtail通过读取到的`version`字段到`streamlog_formats`中找到对应的`user-defined`字段的格式，应用该配置，上面的日志样例`version`字段为 2.1，包含两个自定义字段`level`和`method`，因此日志样例将被解析为如下格式：

```
{
    "source": "10.101.166.127",
    "time": 1455776661,
    "level": "ERROR",
    "method": "com.alibaba.streamlog.App.main(App.java:17)",
    "msg": "connection refused, retry"
}
```

`version`用于解析`user-defined`字段，tag用于寻找数据将要被发送到的Project或Logstore，这两个字段不会作为日志内容发送到阿里云日志服务。另外，Logtail预定义了一些日志格式，这些内置的格式都使用 0.1、1.1 这样以“0.”、“1.”开头的`version`值，所以用户自定义`version`不可以以“0.”、“1.”开头。

常见日志工具接入 Logtail syslog log

- **log4j**

  — 引入 log4j 库。

  ```
  <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.5</version>
    </dependency>
  ```

```
  <dependency>
      <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-core</artifactId>
      <version>2.5</version>
  </dependency>
```

- 程序中引入 log4j 配置文件 `log4j_aliyun.xml`。

```
 <?xml version="1.0" encoding="UTF-8"?>
  <configuration status="OFF">
      <appenders>
          <Socket name="StreamLog" protocol="TCP" host="10.101.166
.173" port="11111">
              <PatternLayout pattern="%X{version} %X{tag} %d{UNIX}
 %X{ip} %-5p %l %enc{%m}%n" />
          </Socket>
      </appenders>
      <loggers>
          <root level="trace">
              <appender-ref ref="StreamLog" />
          </root>
      </loggers>
  </configuration>
```

其中 10.101.166.173:11111 是 Logtail 所在机器的地址。

- 程序中设置 ThreadContext。

```
package com.alibaba.streamlog;
  import org.apache.logging.log4j.LogManager;
  import org.apache.logging.log4j.Logger;
  import org.apache.logging.log4j.ThreadContext;
  public class App
  {
      private static Logger logger = LogManager.getLogger(App.
class);
      public static void main( String[] args ) throws Interrupte
dException
      {
          ThreadContext.put("version", "2.1");
          ThreadContext.put("tag", "streamlog_tag");
          ThreadContext.put("ip", "127.0.0.1");
          while(true)
          {
              logger.error("hello world");
              Thread.sleep(1000);
          }
          //ThreadContext.clearAll();
      }
  }
```

- **tengine**

  tengine 可以通过 syslog 接入 ilogtail。

  tengine 使用 `ngx_http_log_module`模块将日志打入本地 syslog agent，在本地 syslog agent 中 forward 到 rsyslog。

tengine 配置 syslog 请参考：*tengine配置syslog*

示例：

以 user 类型和 info 级别将 access log 发送给本机 Unix dgram(/dev/log)，并设置应用标记为 nginx。

```
access_log syslog:user:info:/var/log/nginx.sock:nginx
```

rsyslog 配置：

```
module(load="imuxsock") # needs to be done just once
input(type="imuxsock" Socket="/var/log/nginx.sock" CreatePath="on")
$template ALI_LOG_FMT,"2.3 streamlog_tag %timegenerated:::date-
unixtimestamp% %fromhost-ip% %pri-text% %app-name% %syslogtag% %msg
:::drop-last-lf%\n"
if $syslogtag == 'nginx' then @@10.101.166.173:11111;ALI_LOG_FMT
```

- **nginx**

以收集 nginx accesslog 为例。

access log 配置：

```
access_log syslog:server=unix:/var/log/nginx.sock,nohostname,tag=
nginx;
```

rsyslog 配置：

```
module(load="imuxsock") # needs to be done just once
input(type="imuxsock" Socket="/var/log/nginx.sock" CreatePath="on")
$template ALI_LOG_FMT,"2.3 streamlog_tag %timegenerated:::date-
unixtimestamp% %fromhost-ip% %pri-text% %app-name% %syslogtag% %msg
:::drop-last-lf%\n"
if $syslogtag == 'nginx' then @@10.101.166.173:11111;ALI_LOG_FMT
```

参考：*http://nginx.org/en/docs/syslog.html*

- **Python Syslog**

示例：

```
import logging
import logging.handlers
logger = logging.getLogger('myLogger')
logger.setLevel(logging.INFO)
#add handler to the logger using unix domain socket '/dev/log'
handler = logging.handlers.SysLogHandler('/dev/log')
#add formatter to the handler
formatter = logging.Formatter('Python: { "loggerName":"%(name)s",
 "asciTime":"%(asctime)s", "pathName":"%(pathname)s", "logRecordC
reationTime":"%(created)f", "functionName":"%(funcName)s", "levelNo
":"%(levelno)s", "lineNo":"%(lineno)d", "time":"%(msecs)d", "
levelName":"%(levelname)s", "message":"%(message)s"}')
```

```
handler.formatter = formatter
logger.addHandler(handler)
logger.info("Test Message")
```

1.2 Syslog

Logtail支持在本地配置TCP端口，接收syslog Agent通过TCP协议转发过来的syslog数据，Logtail解析接收到的数据并转发至LogHub中。

前提条件

设置使用Logtail收集日志前，您需要安装Logtail。Logtail支持Windows和Linux两大操作系统，安装方法参见 *Linux* 和*Windows*。

步骤 1 在日志服务管理控制台创建Logtail syslog配置

1. 在日志服务云控制台单击目标项目，进入**Logstore**列表。
2. 选择目标Logstore，并单击**数据接入向导**图标，进入数据接入流程。
3. 选择数据源类型。

   单击**自定义数据**中的**Syslog**，并单击下一步。
4. 指定 Logtail配置的名称。

   配置名称只能包含小写字母、数字、连字符（-）和下划线（_），且必须以小写字母和数字开头和结尾，长度为 3~63 字节。

   > **说明：**
   > 配置名称设置后不可修改。
5. 填写**Tag**设置。

   如何设置Tag，请参考*Syslog-采集参考*。

图 1-3: 设置Tag

6. 酌情配置高级选项。

   请选择是否打开本地缓存。当日志服务不可用时，日志可以缓存在机器本地目录，服务恢复后进行续传。默认开启缓存，最大缓存值1GB。

7. 根据页面提示，应用Logtail配置到机器组。

   确认勾选所需的机器组并单击 应用到机器组 将配置应用到机器组。

   如果您还未创建机器组，需要先创建一个机器组。有关如何创建机器组，参见 创建*IP*地址机器组。

图 1-4: 应用到机器组

步骤 2 配置Logtail使协议生效

从机器Logtail安装目录下找到 `ilogtail_config.json`，一般在 `/usr/local/ilogtail/`目录下面。根据需求修改和syslog相关的配置。

1. 确认syslog功能已开启。

true表示syslog功能处于打开状态，false表示关闭状态。

```
"streamlog_open" : true
```

2. 配置syslog用于接收日志的内存池大小。程序启动时会一次性申请指定大小的内存，请根据机器内存大小以及实际需求填写，单位是MB。

```
"streamlog_pool_size_in_mb" : 50
```

3. 配置缓冲区大小。需要配置Logtail每次调用socket io rcv 接口使用的缓冲区大小，单位是byte。

```
"streamlog_rcv_size_each_call" : 1024
```

4. 配置日志syslog格式。

```
"streamlog_formats":[]
```

5. 配置TCP绑定地址和端口。需要配置Logtail用于接收syslog日志的TCP绑定地址和端口，默认是绑定0.0.0.0下的11111端口。

```
"streamlog_tcp_addr" : "0.0.0.0",
"streamlog_tcp_port" : 11111
```

6. 配置完成后重启Logtail。重启Logtail要执行以下命令关闭Logtail客户端，并再次打开。

```
sudo /etc/init.d/ilogtaild stop
sudo /etc/init.d/ilogtaild start
```

步骤 3 安装rsyslog并修改配置

如果机器已经安装rsyslog，忽略这一步。

1. 安装rsyslog。

   安装方法请参见：

   - *Ubuntu 安装方法*
   - *Debian 安装方法*
   - *RHEL/CENTOS 安装方法*

2. 修改配置。

   在 `/etc/rsyslog.conf` 中根据需要修改配置，例如：

```
$WorkDirectory /var/spool/rsyslog # where to place spool files
$ActionQueueFileName fwdRule1 # unique name prefix for spool files
$ActionQueueMaxDiskSpace 1g # 1gb space limit (use as much as
possible)
$ActionQueueSaveOnShutdown on # save messages to disk on shutdown
$ActionQueueType LinkedList # run asynchronously
```

```
$ActionResumeRetryCount -1 # infinite retries if host is down
# 定义日志数据的字段
$template ALI_LOG_FMT,"0.1 sys_tag %timegenerated:::date-unixtimest
amp% %fromhost-ip% %hostname% %pri-text% %protocol-version% %app-
name% %procid% %msgid% %msg:::drop-last-lf%\n"
*.* @@10.101.166.173:11111;ALI_LOG_FMT
```

说明：

模板 `ALI_LOG_FMT` 中第二个域的值是 `sys_tag`，这个取值必须和步骤 1 中创建的一致，这个配置的含义是将本机接收到的所有（`\*.\*`）syslog 日志按照 `ALI_LOG_FMT`格式化，使用 TCP 协议转发到 10.101.166.173:11111。机器 10.101.166.173 必须在步骤 1 中的机器组中并且按照步骤 2 配置。

3. 启动rsyslog。

```
sudo /etc/init.d/rsyslog restart
```

启动之前请先检查机器上是否安装了其他syslog的Agent，比如 syslogd、sysklogd、syslog-ng 等，如果有的话请关闭。

上面三步完成之后就可以将机器上的syslog收集到日志服务了。

更多信息

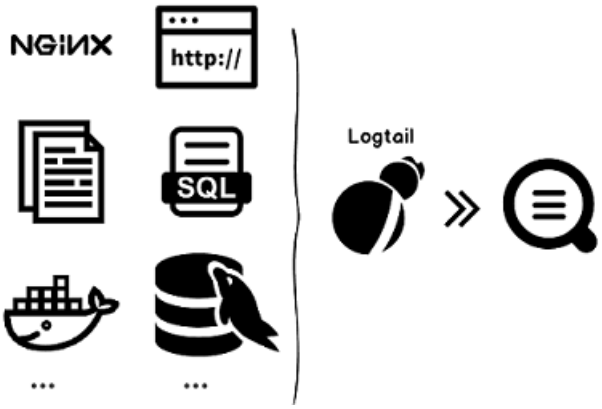
有关syslog日志采集的更多信息以及如何格式化syslog数据，请参见*Syslog-采集参考*。

2 Logtail采集

2.1 简介

Logtail接入服务是日志服务提供的日志采集Agent，通过控制台方式帮助您实时采集阿里云ECS、自建IDC、其他云厂商等服务器上的日志。

图 **2-1: Logtail采集功能**



功能优势

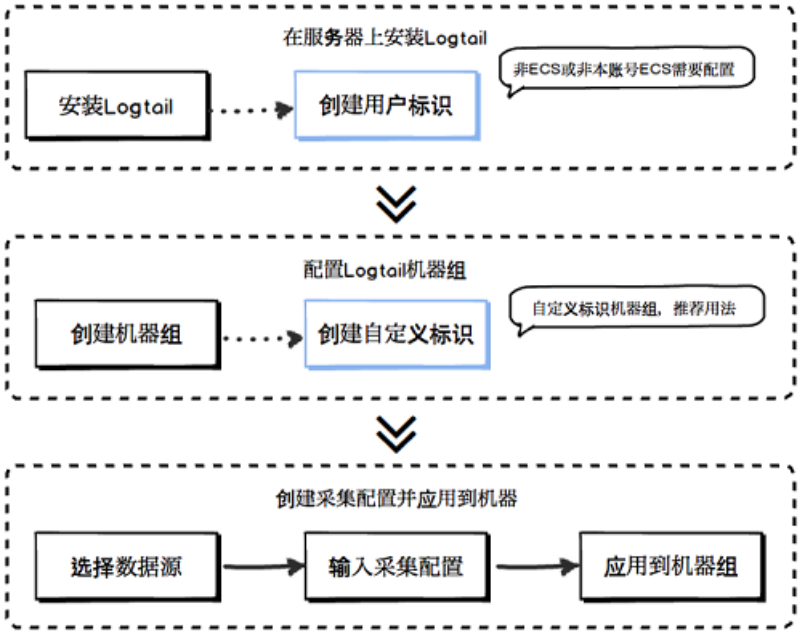
- 基于日志文件、无侵入式的收集日志。用户无需修改应用程序代码，且日志收集不会影响用户应用程序的运行逻辑。
- 除支持文本日志采集外，还支持binlog、http、容器stdout等采集方式。
- 对于容器支持友好，支持标准容器、swarm集群、Kubernetes集群等容器集群的数据采集。
- 能够稳定地处理日志收集过程中各种异常。当遇到网络异常、服务端异常等问题时会采用主动重试、本地缓存数据等措施保障数据安全。
- 基于服务端的集中管理能力。用户在安装Logtail后（参见 *Windows* 和 *Linux* ），只需要在服务端集中配置需要收集的机器、收集方式等信息即可，无需逐个登录服务器进行配置。
- 完善的自我保护机制。为保证运行在客户机器上的收集Agent不会明显影响用户自身服务的性能，Logtail客户端在CPU、内存及网络使用方面都做了严格的限制和保护机制。

处理能力与限制

参见相关限制说明。

配置流程

图 2-2: 配置流程



通过Logtail采集服务器日志可以通过以下步骤完成：

1. 安装Logtail。在需要采集日志的源服务器上安装Logtail操作请参见*Windows* 和 *Linux*。
2. 创建用户自定义标识机器组。从阿里云ECS采集日志不需要执行此步骤。
3. 创建*IP*地址机器组。日志服务通过机器组的方式管理所有需要通过Logtail客户端采集日志的服务器。日志服务支持通过IP或者自定义标识的方式定义机器组。您也可以在应用Logtail配置到机器组时，根据提示创建机器组。
4. 创建Logtail采集配置，并应用到机器组。您可以通过**数据接入向导**创建Logtail配置以 文本日志、*Syslog*等，并将该Logtail配置应用到机器组。

在完成如上流程后，您的ECS服务器上需要收集的新增日志会被主动收集、发送到对应Logstore中，历史数据不会被收集。您可以通过日志服务控制台或者SDK及API查询到这些日志。您还可以通过日志服务查询到所有ECS服务器上的Logtail收集日志状态，例如是否在正常收集，是否有错误等。

Logtail接入服务在日志服务控制台上的完整操作请参考*Logtail* 收集日志 。

容器

- 阿里云容器服务Swarm：参见集成日志服务。

- 阿里云容器服务Kubernetes：参见采集*Kubernetes*日志
- 自建Kubernetes：参见自建*Kubernetes*安装方式
- 自建其他Docker集群：参见标准*Docker*日志采集

核心概念

- **机器组**：一个机器组包含一或多台需要收集一类日志的机器。通过绑定Logtail配置到机器组，可以让日志服务根据同样的Logtail配置采集一个机器组内所有服务器上的日志。您也可以通过日志服务控制台方便地对机器组进行管理（包括创建、删除机器组，添加、移除机器等）。同一个机器组内不可同时包含Windows和Linux机器，但可以包含不同版本的Windows Server或者不同发行版本的Linux机器。
- **Logtail客户端**：Logtail是运行在需要收集日志的服务器上上执行日志收集工作的Agent。安装步骤请参考 *Windows* 和 *Linux*。 在服务器上安装Logtail后，需要配置Logtail并应用到机器组。
  - **Linux** 下，Logtail安装在 `/usr/local/ilogtail` 目录下，并启动两个以 ilogtail 开头的个独立进程，一个为收集进程，另外一个为守护进程，程序运行日志为 `/usr/local/ilogtail/ilogtail.LOG`。
  - **Windows** 下，Logtail安装在目录 `C:\Program Files\Alibaba\Logtail`（32 位系统）或 `C:\Program Files (x86)\Alibaba\Logtail`（64 位系统）下。您可以通过Windows管理工具>服务查看到两个Windows Service，LogtailWorker负责收集日志，LogtailDaemon负责守护工作程序。程序运行日志为安装目录下的 `logtail_*.log`。
- **Logtail配置**：是Logtail收集日志的策略集合。通过为Logtail配置数据源、收集模式等参数，来对机器组内所有服务器进行定制化的收集策略。Logtail配置定义了如何在机器上收集一类日志并解析、发送到日志服务的指定日志库。您可以通过控制台对每个Logstore添加Logtail配置，表示该Logstore接收以此Logtail配置收集的日志。

基本功能

Logtail接入服务提供如下功能：

- **实时收集日志**：动态监控日志文件，实时地读取、解析增量日志。日志从生成到发往服务端的延迟一般在3秒内。

  > 说明：
  > Logtail接入服务不支持对历史数据的收集。对于一条日志，读取该日志的时刻减去日志产生的时刻，差值超过5分钟的会被丢弃。

- **自动处理日志轮转**：很多应用会按照文件大小或者日期对日志文件进行轮转（rotation），把原日志文件重命名，并新建一个空日志文件等待写入。例如：监控*app.LOG*，日志轮转会产生*app.LOG.1*，*app.LOG.2* 等。您可以指定收集日志写入的文件，如 *app.LOG*，Logtail会自动检测到日志轮转过程，保证这个过程中不会出现日志数据丢失。
- **多种采集输入源**：Logtail除支持文本日志采集外，还支持syslog、http、MySQL binlog等输入源，更多内容参见采集数据源配置章节。
- **兼容开源采集Agent**：Logtail支持Logstash、Beats等开源软件采集的数据作为输入源，更多内容参见采集数据源配置章节。
- **自动处理收集异常**：因为服务端错误、网络措施、Quota超限等各种异常导致数据发送失败，Logtail会按场景主动重试。如果重试失败则会将数据写入本地缓存，稍后自动重发。
- **灵活配置收集策略**：可以通过Logtail配置来非常灵活地指定如何在一台ECS服务器上收集日志。具体来说，您可以根据实际场景选择日志目录、文件，既可精确匹配，也可通过通配符模糊匹配。您可以自定义日志收集提取的方式和各个提取字段的名称，日志服务支持正则表达式方式的日志提取。另外，由于日志服务日志数据模型要求每条日志必须有精确的时间戳信息，Logtail提供了自定义的日志时间格式，方便您从不同格式的日志数据中提取必须要的日志时间戳信息。
- **自动同步收集配置**：您在日志服务控制台上新建或更新配置，Logtail一般在3分钟时间内即可自动接受并使之生效，更新配置过程中数据收集不丢失。
- **自动升级客户端**：在您手动安装Logtail到服务器后，日志服务负责Logtail 自动运维升级，此过程无需您参与。在整个Logtail升级过程中日志数据不丢失。
- **自我监控状态**：为避免Logtail客户端消耗您太多资源而影响您其他服务。Logtail客户端会实时监控自身CPU和内存消耗。如果Logtail客户端在运行过程中，资源使用超出限制将会自动重启，避免影响机器上的其它作业。同时，该客户端也会有主动的网络限流保护措施，防止过度消耗用户带宽。
- **签名数据发送**：为保证您的数据在发送过程中不会被篡改，Logtail客户端会主动获取用户的阿里云访问秘钥并对所有发送日志的数据包进行数据签名。

说明：

Logtail客户端在获取您的阿里云访问秘钥时采用HTTPS通道，保障您的访问秘钥安全性。

2.2 选择网络

采集日志数据到日志服务时，日志数据可以通过阿里云内网、公网和全球加速网络传输。

网络类型

- 公网：使用公网传输日志数据，不仅会受到网络带宽的限制，还可能会因网络抖动、延迟、丢包等影响数据采集的速度和稳定性。
- 阿里云内网：阿里云内网为千兆共享网络，日志数据通过阿里云内网传输比公网传输更快速、稳定。内网包括**VPC**环境和经典网络环境。
- 全球加速：利用阿里云CDN边缘节点进行日志采集加速，相对公网采集在网络延迟、稳定性上具有很大优势。

如何选择网络

- 内网：

  您的日志数据是否通过阿里云内网传输，取决于您的服务器类型以及服务器和日志服务Project是否在同一地域。仅有以下两种情况可以使用阿里云内网传输：

  - 本账号下的**ECS**和日志服务**Project**在同一地域。
  - 其他账号的**ECS**和本账号的日志服务**Project**在同一地域。

  因此，建议您在ECS的相同地域下创建日志服务Project，并将日志采集到同地域的日志服务Project中。ECS上的日志数据自动通过阿里云内网写入日志服务，不消耗公网带宽。

  > 说明：
  > 在服务器上安装Logtail时，选择的地域必须和Project所在地域一致，否则无法正常采集日志数据。

- 全球加速：

  如果您的服务器分布在海外各地的自建机房、或者来自海外云厂商，使用公网传输数据可能会出现网络延迟高、传输不稳定等问题，可以通过全球加速传输数据。全球加速利用阿里云CDN边缘节点进行日志采集加速，相对公网采集在网络延迟、稳定性上具有很大优势。

- 公网：

  在以下两种情况时，您可以选择网络类型为公网：

  - 服务器为ECS，但和日志服务Project位于不同地域。
  - 服务器为其他云厂商服务器、自建IDC。

服务器类型	是否与Project同一地域	是否需要配置AliUid	网络类型
本账号下的ECS	同一地域	不需要	阿里云内网
	不同地域	不需要	公网或全球加速
其他账号下的ECS	同一地域	需要	阿里云内网
	不同地域	需要	公网或全球加速
其他云厂商服务器、自建IDC	-	需要	公网或全球加速

说明：

日志服务无法获取非本账号下ECS、其他服务器的属主信息，请在安装Logtail后手动配置用户标识（AliUid），否则安装Logtail的服务器会心跳异常、无法收集日志。详细步骤请参见 为非本账号*ECS*、自建*IDC*配置*AliUid*。

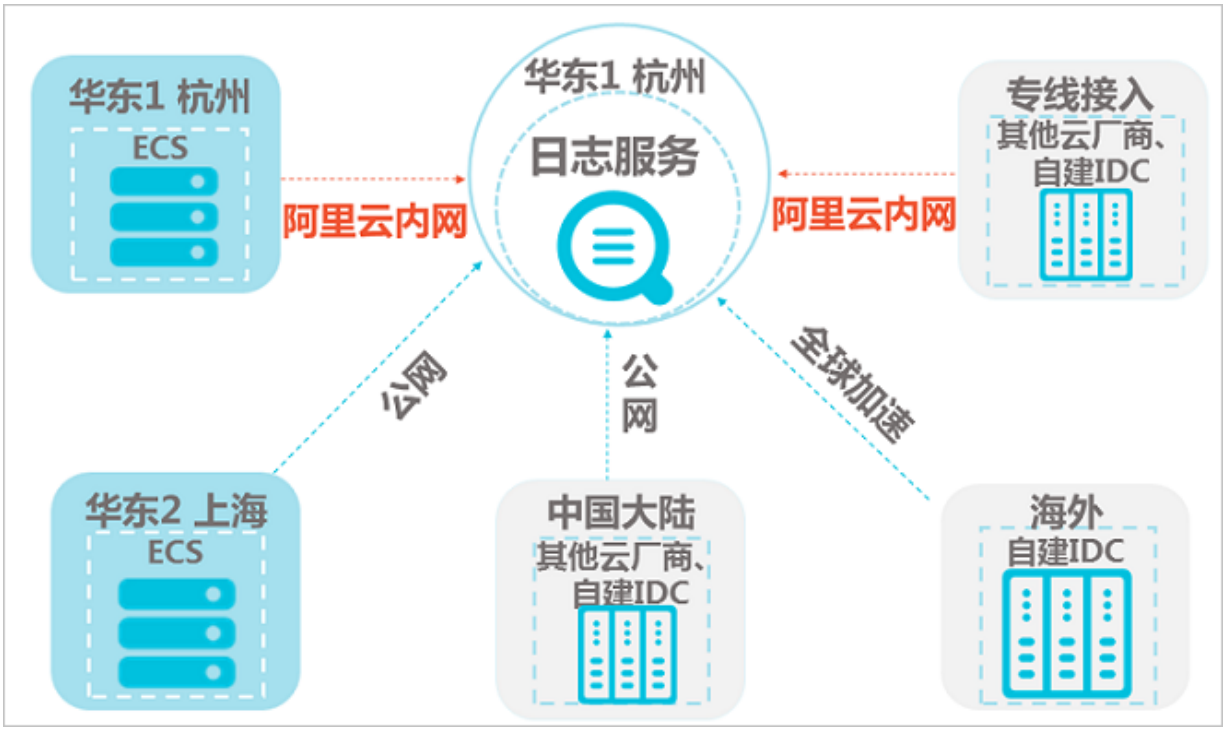
网络选择示例

以下是各种常见场景的网络选择示例。

场景类型	日志服务Project地域	服务器类型	ECS地域	安装Logtail时选择的地域	网络类型	是否需要配置AliUid
相同地域场景	华东1（杭州）	本账号ECS	华东1（杭州）	华东1（杭州）	内网	不需要
不同地域场景	华东2（上海）	本账号ECS	华北1（北京）	华北1（北京）	公网	不需要
其他账号场景	华东2（上海）	其他账号ECS	华北1（北京）	华北1（北京）	公网	需要
本地机房场景	华东5（深圳）	自建IDC	-	华东5（深圳）	公网	需要
全球加速场景	香港	自建IDC	-	香港	全球加速	需要

说明：

全球加速场景中，日志服务Project创建在香港地域，服务器为全球各地的自建机房，数据采集的速度和可靠性尤为重要，所以建议您在类似场景下安装Logtail时选择香港地域的全球加速网络类型。日志数据通过全球加速传输，比公网传输的网络稳定性更高、性能更好。



经典网络切换VPC后的配置更新

安装Logtail后，如果您的ECS网络类型由经典网络切换为VPC，请参考以下步骤更新配置。

1. 以管理员身份重启Logtail。
   - **Linux**：

     ```
     sudo /etc/init.d/ilogtaild stop
     sudo /etc/init.d/ilogtaild start
     ```

   - **Windows**：

     打开控制面板中的管理工具，打开服务，找到*LogtailWorker*并右键单击重新启动。
2. 更新机器组配置。
   - 自定义标识

     若机器组中配置了自定义标识，则无需手动更新机器组配置，可以直接正常使用VPC网络。
   - **IP地址**

若机器组中配置了ECS云服务器IP地址，则需将机器组内的IP更换为重启Logtail后获取的IP地址，即*app_info.json*中的ip字段。

*app_info.json*文件地址：

- Linux：*/usr/local/ilogtail/app_info.json*
- Windows x64：*C:\Program Files (x86)\Alibaba\Logtail\app_info.json*
- Windows x32：*C:\Program Files\Alibaba\Logtail\app_info.json*

2.3 数据源

2.3.1 文本日志

Logtail客户端可以帮助日志服务用户简单地通过控制台收集ECS云服务器或您本地服务器上的文本日志。

前提条件

- 设置使用Logtail收集日志前，您需要安装Logtail。Logtail支持Windows和 Linux两大操作系统，安装方法参见*Linux*和*Windows*。
- 采集ECS或本地服务器日志，请确保您已开启了80端口和443端口。

使用限制

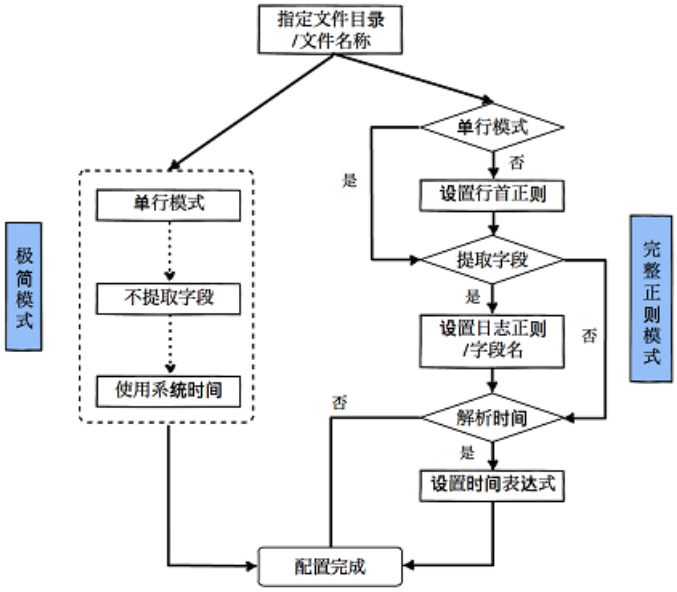
- 一个文件只能被一个配置收集。如果需要采集多份，建议以软链接形式实现。例如`/home/log/nginx/log`下需要采集两份，则其中一个配置原始路径，另外创建一个该文件夹的软链接`ln -s /home/log/nginx/log /home/log/nginx/link_log`，另一个配置软链接路径即可。
- Logtail客户端支持的操作系统可参考简介。
- 经典网络或者VPC下的ECS必须和日志服务Project在相同地域；如果您的日志源数据通过公网传输（类似IDC用法），可以根据地域说明选择日志服务Project的地域。

  经典网络或者VPC下的ECS必须和日志服务Project在相同地域；如果您的日志源数据通过公网传输（类似IDC用法），可以根据实际需求选择日志服务Project的地域。

日志收集配置流程

通过控制台配置Logtail收集文本日志，可以通过极简模式、分隔符模式、JSON 模式、完整正则模式等方式收集日志进行设置。以极简模式和完整正则模式为例，配置流程如下。

图 2-3: 配置流程



操作步骤

1. 在日志服务管理控制台单击目标项目，进入**Logstore**列表。
2. 选择目标Logstore，并单击Logstore名称右侧的**数据接入向导**图标，进入配置数据接入流程。
3. 选择数据类型。

   单击**自定义数据**中的**文本文件**，并单击下一步，进入**数据源设置**界面。
4. 指定**配置名称**。

   配置名称只能包含小写字母、数字、连字符（-）和下划线（_），且必须以小写字母和数字开头和结尾，长度为3~63字节。

   > 说明：
   > 配置名称设置后不可修改。
5. 指定日志的目录和文件名。

   目录结构支持完整路径和通配符两种模式。

   > 说明：
   > 目录通配符只支持 *和? 两种。

   日志文件名支持完整文件名和通配符两种模式，文件名规则请参考*Wildcard matching*。

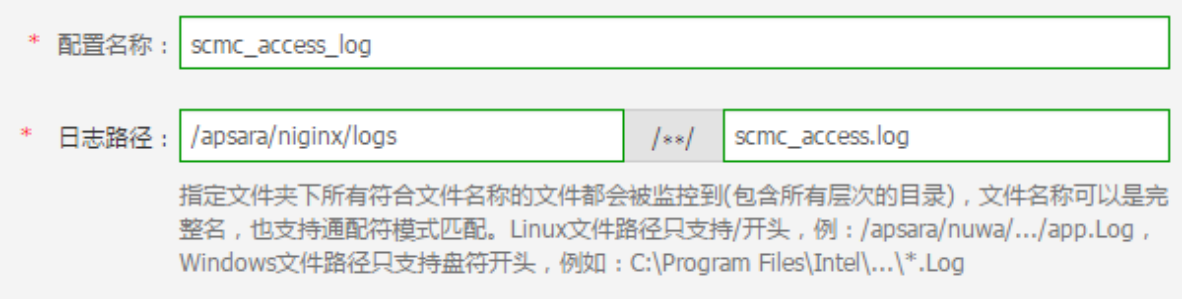
日志文件查找模式为多层目录匹配，即指定文件夹下所有符合文件名模式的文件都会被监控到，包含所有层次的目录。

- 例如`/apsara/nuwa/ ··· /*.log`表示`/apsara/nuwa`目录中（包含该目录的递归子目录）后缀名为`.log`的文件。
- 例如`/var/logs/app_* ··· /*.log*`表示`/var/logs`目录下所有符合`app_*`模式的目录中（包含该目录的递归子目录）文件名包含`.log`的文件。

说明：

一个文件只能被一个配置收集。

图 2-4: 指定目录和文件名



6. 设置收集模式。

Logtail支持极简模式、分隔符模式、JSON 模式、完整正则模式等方式收集日志，具体说明请参考采集方式。本示例以极简模式和完整正则模式为例介绍收集模式的设置。

- 极简模式

  目前极简模式即单行模式。单行模式下默认一行日志内容为一条日志，即日志文件中，以换行符分隔两条日志。单行模式下，不提取日志字段，即默认正则表达式为(.*)，同时记录当前服务器的系统时间作为日志产生的时间。如果后续您需要对极简模式进行更详细的设置，可以通过修改配置进入完整模式逐项调整。有关如何修改Logtail配置，参见管理采集配置。

  极简模式下，您只需要指定文件目录和文件名称，Logtail会按照每行一条日志进行收集，同时将日志时间设定为抓取该条日志时服务器的系统时间，不会提取日志内容中的字段。

- 完整正则模式

  如果需要对内容做更多个性化的字段提取设置（比如跨行日志，提取字段等），选择 完整正则模式 即可进行个性化定制。您可以参考 简介了解这些参数的具体含义和设置方式。

1. 输入 **日志样例**。

   让您提供日志样例的目的是方便日志服务控制台自动提取其中的正则匹配模式，请务必使用实际场景的日志。

2. 关闭 **单行模式**。

   默认为使用单行模式，即按照一行为一条日志进行分割，如果需要收集跨行日志（比如 Java 程序日志），需要关闭**单行模式**，然后设置 **行首正则表达式**。

3. 设置 **行首正则表达式**。

   提供**自动生成**和**手动输入**两种功能。填写完日志样例后，单击 **自动生成** 即会生成正则；如果无法自动生成，可以切换为手动模式输入进行验证。

4. 开启**提取字段**。

   如果需要对日志内容中的字段单独分析处理，可以使用 **提取字段** 功能将指定字段变成 Key-Value 对后发送到服务端，所以需要您指定解析一条日志内容的方式，即正则表达式。

   日志服务控制台提供两种方式让您指定解析正则表达式。第一种方式是通过简单交互自动生成正则表达式。您通过“划选”的方式操作日志样例，选中需要提取的字段，日志服务控制台会自动生成正则表达式。

   如下图所示：

图 **2-5:** 自动生成正则表达式

尽管自动生成方式避免了您自己写正则表达式的困扰，但是自动生成的正则表达式很多时候并不是完美的，您可以手动直接输入正则表达式。单击 **手动输入正则表达式** 切换到手动输入模式。手动输入完成后，单击右侧的**验证** 即会验证您输入的正则表达式是否可以解析、提取日志样例。

无论使用自动生成还是手动输入方式，产生日志解析正则表达式后，您都需要给每个提取字段命名，设定对应字段的 Key，如下图所示：

图 **2-6:**

5. 设置使用系统时间。

   默认设置 **手动输入正则表达式**。如果关闭，您需要在提取字段时指定某一字段（Value）为时间字段，并命名为 `time`（如上图）。在选取 `time` 字段后，您可以单击 **时间转换格式** 中的 **自动生成** 生成解析该时间字段的方式。关于日志时间格式的更多信息请参考文本-配置时间格式。

7. （可选）配置高级选项，设置完成后，单击下一步。

   请根据您的需求配置 **本地缓存**、**配置日志*Topic*生成方式**、**日志文件编码**、**最大监控目录深度**、**超时属性**和**过滤器配置**。如没有特殊需求，可以保持默认配置。

配置项	详情
本地缓存	请选择是否打开**本地缓存**。当日志服务不可用时，日志可以缓存在机器本地目录，服务恢复后进行续传，默认最大缓存值1GB。
丢弃解析失败日志	请选择解析失败的日志是否上传到日志服务。 开启后，解析失败的日志不上传到日志服务；关闭后，日志解析失败时上传原始日志，其中Key为`__raw_log__`、Value为日志内容。

配置项	详情
上传原始日志	请选择是否需要上传原始日志。开启该功能后，原始日志内容会作为__raw__字段与解析过的日志一并上传。
Topic生成方式	• 空-不生成Topic：默认选项，表示设置Topic为空字符串，在查询日志时不需要输入Topic即可查询。 • 机器组Topic属性：设置Topic生成方式为机器组Topic属性，可以用于明确区分不同前端服务器产生的日志数据。 • 文件路径正则：选择此项之后，您需要填写下方的自定义正则，用正则式从路径里提取一部分内容作为Topic。可以用于区分具体用户或实例产生的日志数据。
自定义正则	如您选择了文件路径正则方式生成Topic，需要在此处填写您的自定义正则式。
日志文件编码	• utf8：指定使用UTF-8编码。 • gbk：指定使用GBK编码。
最大监控目录深度	指定从日志源采集日志时，监控目录的最大深度，即最多监控几层日志。最大目录监控深度范围0-1000，0代表只监控本层目录。
超时属性	如果一个日志文件在指定时间内没有任何更新，则认为该文件已超时。您可以对超时属性指定以下配置。 • 永不超时：指定持续监控所有日志文件，永不超时。 • 30分钟超时：如日志文件在30分钟内没有更新，则认为已超时，并不再监控该文件。
过滤器配置	日志只有完全符合过滤器中的条件才会被收集。 例如： • 满足条件即收集：配置`Key:level Regex:WARNING|ERROR`，表示只收集level为WARNING或ERROR类型的日志。 • 过滤不符合条件的数据： — 配置`Key:level Regex:^(?!.*(INFO|DEBUG))`，表示代表不收集level为INFO或DEBUG类型的日志。 — 配置`Key:url Regex:.*^(?!.*(healthcheck)).*`，表示不采集url中带有healthcheck的日志，例如key为url，value为`/inner/healthcheck/jiankong.html`的日志将不会被采集。 更多示例可参考*regex-exclude-word*、*regex-exclude-pattern*。

8. 勾选所需的机器组并单击应用到机器组将配置应用到机器组。

如果您还未创建机器组，需要先创建一个机器组。有关如何创建机器组，参见 创建*IP*地址机器组。

> 说明：
>
> - Logtail配置推送生效时间最长需要 3 分钟，请耐心等待。
> - 如果需要收集 IIS 的访问日志，请务必首先参考*Logstash* 收集 *IIS* 日志配置 IIS。
> - 创建Logtail配置后，您可以查看Logtail配置列表、修改Logtail配置或删除Logtail配置。详细信息，参见管理采集配置。

图 **2-7:** 应用到机器组

完成配置后，日志服务开始收集日志。

后续操作

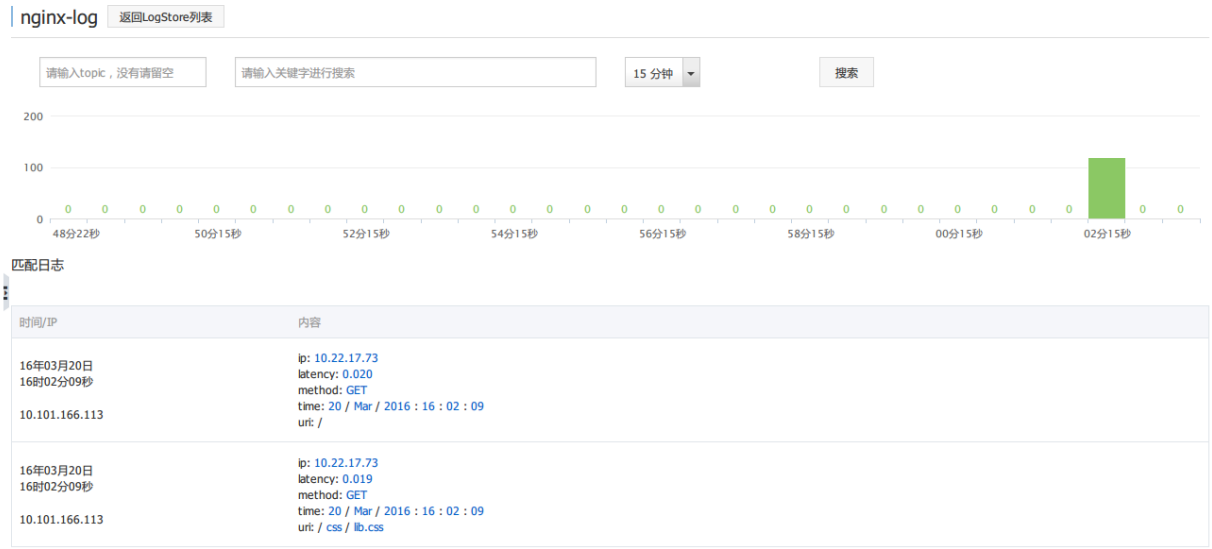
完成以上操作后，您可以在页面指引下继续配置查询分析 **&** 可视化和投递 **&** ETL。

极简模式下收集到服务端的日志如下所示。每条日志的所有内容都在名为 **content** 的KEY下面。

图 2-8: 极简模式收集预览

完整正则模式下，收集到服务端的日志内容如下所示。每条日志的内容都按照设定的 Key-Value收集到了服务端。

图 2-9: 正则模式收集预览



Logtail配置项

配置Logtail时需要填写配置项，常用配置项具体描述与限制如下：

配置项	描述
日志路径	目录结构支持完整路径和通配符两种模式。通配符模式为多层目录匹配，即指定文件夹下所有符合文件名称的文件都会被监控到，包含所有层次的目录。
日志文件名	指定收集日志文件名称，区分大小写，可以使用通配符。例如`*.log`。Linux下的文件名通配符包括”*“, “?”和”[…]”。
本地存储	表示是否启用本地缓存临时存储因网络短暂中断而无法发送的日志。
日志首行头	指定多行日志的起始头，需指定正则表达式。在多行日志收集场景下（如应用程序日志中的堆栈信息），无法使用换行符来分割每条日志。这时需要指定一个多行日志的起始头，当发现该起始头则表示上条日志已经结束，新的一条已经开始。由于每条日志的起始头可能并不一样（如时间戳），故需要指定一个起始头的匹配规则，即这里的正则表达式。
日志解析表达式	定义如何提取一条日志信息，并转化成为日志服务日志的格式。用户需要指定一个正则表达式提

	取需要的日志字段信息，并且定义每个提取的字段名称。
日志时间格式	定义如何解析日志数据中的时间戳字符串的时间格式，具体请参见文本-配置时间格式。

日志写入方式

除了使用Logtail收集日志外，日志服务还提供API和SDK的方式，以方便您写入日志。

- 使用 API 写入日志

  日志服务提供REST风格的API帮助您写入日志。您可以通过API中的 *PostLogStoreLogs* 接口写入数据。关于API的完整参考请见 概览。

- 使用 SDK 写入日志

  除了API，日志服务还提供了多种语言（Java、.NET、PHP 和 Python）的SDK方便您写入日志。关于SDK的完整参考请见 概述。

2.3.2 文本-配置解析

指定日志行分割方式

一条完整的访问日志一般为一行一条，例如Nginx的访问日志，每条日志以换行符分割。例如以下两条访问日志：

```
10.1.1.1 - - [13/Mar/2016:10:00:10 +0800] "GET / HTTP/1.1" 0.011 180
404 570 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; 360se
)"
10.1.1.1 - - [13/Mar/2016:10:00:11 +0800] "GET / HTTP/1.1" 0.011 180
404 570 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; 360se
)"
```

Java应用中的程序日志，一条日志通常会跨越多行，因此只能通过日志开头的特征区分每条日志行首，例如以下 Java 程序日志：

```
[2016-03-18T14:16:16,000] [INFO] [SessionTracker] [SessionTrackerImpl.
java:148] Expiring sessions
0x152436b9a12aecf, 50000
0x152436b9a12aed2, 50000
0x152436b9a12aed1, 50000
0x152436b9a12aed0, 50000
```

以上Java日志起始字段均为时间格式，即行首正则表达式为：[\d+-\d+-\w+:\d+:\d+,\d+]\s.*。在控制台可按照如下格式填写：

图 2-10: 完整模式解析正则表达式

提取日志字段内容

根据日志服务数据模型要求，一条日志的内容包含一个或者多个Key-Value对，如果提取指定字段进行分析处理，需要设置正则表达式提取指定内容；如果不需要对日志内容进行处理，可以将整条日志做为一对Key-Value对。

对于上例中的访问日志，您可以选择提取字段或不提取字段。

- 提取字段

  正则表达式为`(\S+)\s-\s-\s\[(\S+)\s[^]]+]\s"(\w+).*`，提取内容为：`10.1.1.1`、`13/Mar/2016:10:00`和`GET`。

- 不提取字段

正则表达式为`(.*)`，提取内容为：`10.1.1.1 - - [13/Mar/2016:10:00:10 +0800] "GET / HTTP/1.1" 0.011 180 404 570 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; 360se)"`。

指定日志时间

根据日志服务数据模型要求，一条日志必须要有时间（time）字段，并且格式为UNIX时间戳。目前提供使用系统时间（即Logtail抓取该条日志的时间）或者日志内容中的时间做为日志的时间。

对于上例中的访问日志：

- 如果提取日志内容中的时间字段，时间为`13/Mar/2016:10:00:10`，时间表达式为`%d/%b/%Y:%H:%M:%S`。
- 抓取日志时的系统时间，时间为抓取日志时的时间戳。

2.3.3 文本-配置时间格式

每条日志服务日志都必须包括该日志发生的时间戳信息，Logtail接入服务在采集用户日志文件中的日志数据时，必须提取该条日志中时间戳字符串并把它解析为时间戳。因此，Logtail需要您指定其日志的时间戳格式帮助解析。

Linux 平台下的 Logtail 支持 strftime 函数提供的所有时间格式。只需要您的日志时间戳字符串能够被该函数定义的日志格式所表达，即可以被 Logtail 解析并使用。

现实环境中的日志时间戳字符串格式非常多样化，为方便用户配置，Logtail 支持的常见日志时间格式如下：

支持格式	说明	示例
%a	星期的缩写。	Fri
%A	星期的全称。	Friday
%b	月份的缩写。	Jan
%B	月份的全称。	January
%d	每月第几天，十进制格式，范围为01~31。	07, 31
%h	月份的缩写，与%b相同。	Jan
%H	小时，24小时制。	22
%I	小时，12小时制。	11

支持格式	说明	示例
%m	月份，十进制格式。	08
%M	分钟，十时制格式，范围为00~59。	59
%n	换行符。	换行符
%p	本地的AM（上午）或PM（下午）。	AM/PM
%r	12小时制的时间组合，与`%I:%M:%S %p`相同。	11:59:59 AM
%R	小时和分钟组合，与`%H:%M`相同。	23:59
%S	秒数，十进制，范围为00~59。	59
%t	TAB符。	TAB符
%y	年份，十进制，不带世纪，范围为00~99。	04；98
%Y	年份，十进制。	2004；1998
%z	时区或者缩写。	-07:00, +0800
%C	十进制世纪，范围为00~99。	16
%e	每月第几天，十进制格式，范围为1~31。如果是个位数字，前面需要加空格。	7，31
%j	一年天数的十进制表示，范围为00~366。	365
%u	星期的十进制表示，范围为1~7，1 表示周一。	2
%U	每年的第几周，星期天是一周的开始。范围为00~53。	23
%V	每年的第几周，星期一是一周的开始。如果一月份刚开始的一周>=4天，则认为是第1周，否则认为下一个星期一是第1周。范围为01~53。	24

支持格式	说明	示例
%w	星期几，十进制格式 ，范围为0~6，0代表周日。	5
%W	每年的第几周，星期一是一周的开始。范围为00~53。	23
%c	标准的日期、时间。	需要指定长日期、短日期等更多信息，可以考虑用上面支持的格式更精确表达。
%x	标准的日期。	需要指定长日期、短日期等更多信息，可以考虑用上面支持的格式更精确表达
%X	标准的时间。	需要指定长日期、短日期等更多信息，可以考虑用上面支持的格式更精确表达
%s	Unix时间戳。	1476187251

2.3.4 文本-导入历史日志文件

Logtail默认只采集增量的日志文件，如果您需要导入历史文件，可使用Logtail自带的导入历史文件功能。

前提条件

- Logtail版本需在`0.16.6`及以上。
- 需要被采集的历史文件必须处于采集配置覆盖的范围下，且此前未被Logtail采集过。
- 历史文件的最后修改时间必须在配置Logtail之前。
- 本地事件导入最长延迟为1分钟。
- 由于加载本地配置属于特殊行为，Logtail会向服务器发送`LOAD_LOCAL_EVENT_ALARM`以提醒用户。

背景信息

Logtail基于事件进行文件采集，事件通常由监听或定期轮询文件修改产生。除以上方式外，Logtail还支持从本地文件中加载事件，以此驱动日志采集。历史文件采集就是基于本地事件加载实现的功能。

操作步骤

1. 创建采集配置。

   请根据文本日志创建采集配置并应用到机器组，确保文件处于对应采集配置覆盖范围。

2. 获取配置唯一标识。

   采集配置的唯一标识可在本地`/usr/local/ilogtail/user_log_config.json`中获取，示例如下：

```
grep "##" /usr/local/ilogtail/user_log_config.json | awk '{print $1
}'
                    "##1.0##log-config-test$multi"
                    "##1.0##log-config-test$ecs-test"
                    "##1.0##log-config-test$metric_system_test"
                    "##1.0##log-config-test$redis-status"
```

3. 添加本地事件。

   本地事件保存的路径为`/usr/local/ilogtail/local_event.json`，类型为标准JSON，格式为：

```
[
  {
    "config" : "${your_config_unique_id}",
    "dir" : "${your_log_dir}",
    "name" : "${your_log_file_name}"
   },
  {
    ...
   }
    ...
]
```

   - 配置项

配置项	说明	示例
config	步骤2获取的配置唯一标识。	##1.0#
dir	日志所在文件夹。 说明： 文件夹不能以/结尾。	/data
name	日志文件名	acces

   说明：

为防止Logtail加载无效的JSON，建议您将本地事件配置保存在临时文件中，编辑完成后拷贝到`/usr/local/ilogtail/local_event.json`文件中。

- 配置示例

```
$ cat /usr/local/ilogtail/local_event.json
[
  {
    "config": "##1.0##log-config-test$ecs-test",
    "dir": "/data/log/",
    "name": "access.log.2017-08-08"
  },
  {
    "config": "##1.0##log-config-test$ecs-test",
    "dir": "/tmp",
    "name": "access.log.2017-08-09"
  }
]
```

- 检查**Logtail**是否加载配置

  通常情况下，本地`local_event.json`保存后，Logtail会在1分钟内将本地配置文件加载到内存中，并将`local_event.json`内容清空。

  您可以通过以下三种方式检查Logtail是否已经读取事件：

  — 若`local_event.json`文件被清空，说明Logtail已经读取到事件信息。

  — 检查`/usr/local/ilogtail/ilogtail.LOG`文件中是否包含`process local event`关键字。若`local_event.json`被清空但未查询到该组关键字，可能因为您的本地配置文件内容不合法而被过滤。

  — 通过查询诊断错误查询是否存在`LOAD_LOCAL_EVENT_ALARM`提示。

- 配置被加载但未采集到数据

  若Logtail已经加载配置但数据未被采集到，可能由以下几种原因造成：

  — 配置不合法。

  — 本地配置`config`不存在。

  — 日志文件不在Logtail采集配置已设定的路径下。

  — 该日志文件被Logtail采集过。

- 如何采集已经被采集过的数据

  如果您需要采集已经被采集过的数据，可通过以下方式实现：

  1. 执行命令`/etc/init.d/ilogtaild stop`停止Logtail。

2. 在`/tmp/logtail_check_point`文件中查找对应的日志文件路径。
3. 删除该日志文件的checkpoint（JSON object）并保存。
4. 按照本文步骤3添加本地事件。
5. 执行命令`/etc/init.d/ilogtaild start`启动Logtail。

2.3.5 文本-生成主题

说明：

syslog不支持配置Topic。

Topic生成方式

用户可以在Logtail收集日志时设置Topic，也可以使用API/SDK上传数据时设置Topic。目前支持通过控制台设置Topic生成方式为空-不生成**Topic**、机器组**Topic**属性和文件路径正则。

- 空-不生成**Topic**

  通过控制台配置Logtail收集文本文件时，日志Topic生成方式默认为空-不生成**Topic**，即Topic为空字符串，在查询日志时不需要输入Topic即可查询。

- 机器组**Topic**属性

  机器组**Topic**属性方式用于明确区分不同服务器产生的日志数据。如果您的不同服务器日志数据均保存在相同文件路径或相同文件中，当您需要在收集日志时通过Topic区分不同服务器的日志数据，可以将机器分为不同的机器组，即在创建机器组时，为不同的机器组设置不同的**Topic**属性，并设置**Topic**生成方式为机器组**Topic**属性。将机器组应用之前创建的Logtail配置后，即完成对应配置。

  如选择机器组**Topic**属性，Logtail上报数据时会将机器所在机器组的Topic属性作为主题名称上传至日志服务，在使用日志索引分析功能查询时需要指定Topic，即需要指定目标机器组Topic属性为查询条件。

- 文件路径正则

  文件路径正则方式用于区分具体用户或实例产生的日志数据。如果服务日志根据不同的用户或者实例将日志记录在不同目录下面，但是只要下级目录、日志文相同件名称相同，日志服务在收集日志文件时就无法明确区分日志内容是由那个用户哪个产生的。此时可以设置**Topic**生成方式为文件路径正则，并且输入文件路径的正则表达式，配置Topic为实例名称。

当选择**文件路径正则**主题生成方式时，Logtail上报数据时会将实例名称作为主题名称上传至日志服务。根据您的目录结构和配置，会生成不同的Topic，在使用**日志索引分析**功能查询时需要指定主题名称为实例名称。

设置日志Topic

1. 根据文本日志，通过控制台配置Logtail。

   如您需要配置Topic生成方式为**机器组Topic属性**，请在创建机器组/修改机器组页面中配置**机器组Topic**。

2. 在Logtail配置页面中，展开**高级选项**，在**Topic生成方式**中选择Topic的生成方式。

图 **2-11:** 设置日志Topic

修改日志Topic

如您需要修改日志Topic的生成方式，请直接在Logtail配置界面修改**Topic生成方式**选项。

说明：

修改后的配置仅对生效后采集的新数据有效。

2.3.6 自定义插件

日志服务Logtail除了支持采集文本日志及syslog之外，也通过Logtail插件支持配置若干数据源，例如Http、MySQL查询结果和MySQL Binlog。

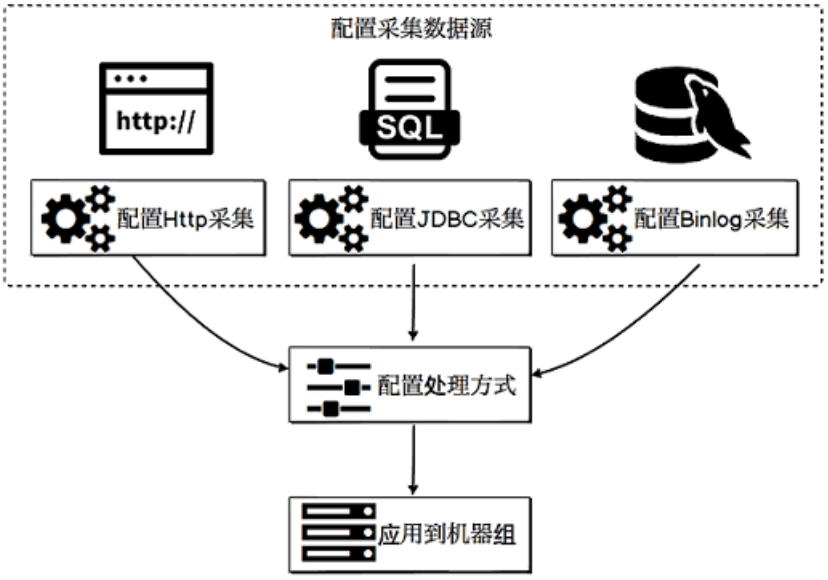
通过配置采集HTTP数据，并将处理结果上传到日志服务，可以进行实时的服务可用性检测和持续的可用性监控；配置MySQL查询结果为数据源，可以根据自增ID或时间等标志进行增量数据同步；配置SQL数据源以同步MySQL Binlog，可以增量订阅数据库改动，并进行实时查询与分析。

说明：

此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

配置流程

图 **2-12:** 配置流程



1. 配置输入源采集方式。

   每种不同的数据源具有不同的采集配置格式，请按照您的数据源类型选择对应的采集配置格式。

   - 采集 *MySQLBinlog*数据
   - 采集 *MySQL*查询结果
   - 采集 *HTTP*数据
   - 采集 容器标准输出

2. 配置处理方式。

Logtail对于Binlog、MySQL查询结果、Nginx监控和HTTP输入源提供了统一的数据处理配置。用户可对一个输入源配置多个处理方式，各类输入源均支持所有类型的处理方式。Logtail会根据配置顺序逐一执行各个处理方式。

详细说明请参考通用处理配置。

3. 应用到机器组。

配置采集方式和处理方式之后，保存并应用到指定机器组，Logtail会自动拉取配置并开始采集。

2.3.7 插件-MySQL Binlog方式

MySQL Binlog同步类似 *canal* 功能，以MySQL slave的形式基于Binlog进行同步，性能较高。

注意： 此功能目前只支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

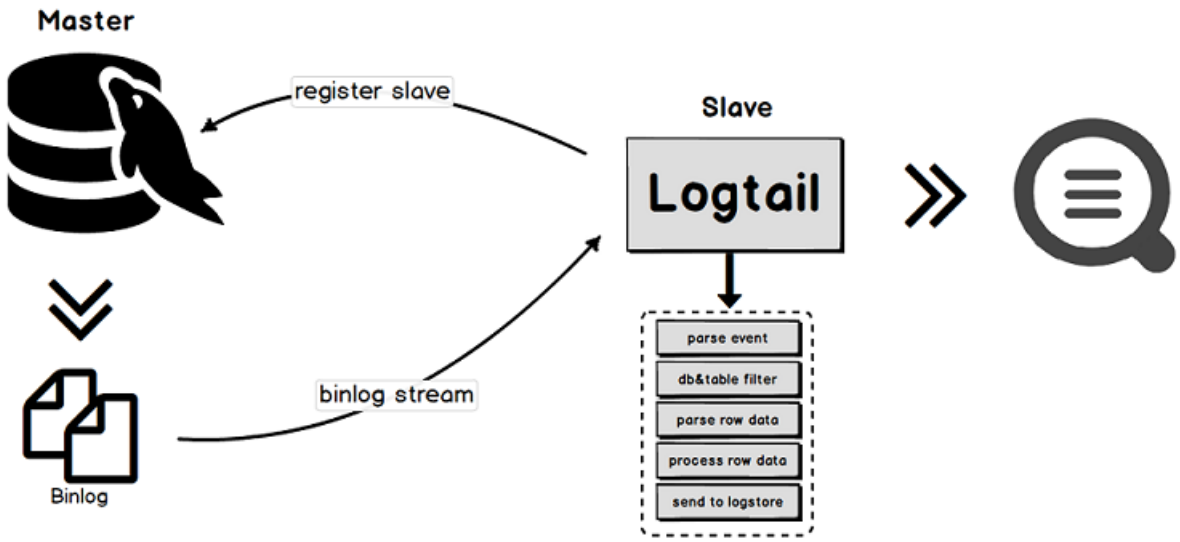
功能

- 通过Binlog订阅数据库增量更新数据，性能优越，支持RDS等MySQL协议的数据库。
- 支持数据库过滤（支持正则）。
- 支持同步点设置。
- 支持Checkpoint保存同步状态。

实现原理

如下图所示，Logtail内部实现了MySQL Slave的交互协议，将自己伪装成为MySQL的Slave节点，向MySQL master发送dump协议；MySQL的master收到dump请求后，会将自身的Binary log实时推送给Logtail，Logtail对Binlog进行事件解析、过滤、数据解析等，并将解析好的数据上传到日志服务。

图 2-13: 实现原理



应用场景

适用于数据量较大且性能要求较高的数据同步场景。

- 增量订阅数据库改动进行实时查询与分析。
- 数据库操作审计。
- 通过数据库更新信息使用日志服务进行自定义查询分析、可视化、对接下游流计算、导入MaxCompute离线计算、导入OSS长期存储等。

参数说明

MySQL Binlog方式输入源类型为：`service_canal`。

参数	类型	必选或可选	参数说明
Host	string	可选	数据库主机，默认为127.0.0.1。
Port	int	可选	数据库端口，默认为3306。
User	string	可选	数据库用户名，默认为root。
Password	string	可选	数据库密码；默认为空。

参数	类型	必选或可选	参数说明
ServerID	int	可选	Logtail伪装成的Mysql Slave ID，默认为125。 说明：ServerID对于一个MySQL数据库必须唯一，否则同步失败。
IncludeTables	string 数组	可选	包含的表名称（包括db，例如`test_db.test_table`），为正则表达式，若某表不符合IncludeTables任一条件则该表不会被采集；不设置时默认收集所有表。 说明：若需要完全匹配，请在前后分别加上^$，例如`^test_db.test_table$`。
ExcludeTables	string 数组	可选	忽略的表名称（包括db，例如`test_db.test_table`），为正则表达式，若某表符合ExcludeTables任一条件则该表不会被采集；不设置时默认收集所有表。 说明：若需要完全匹配，请在前后分别加上^$，例

参数	类型	必选或可选	参数说明
			如`^test_db.test_table$`。
StartBinName	string	可选	首次采集的Binlog文件名，不设置时默认从当前时间点开始采集。
StartBinLogPos	int	可选	首次采集的Binlog文件名的offset，默认为0。
EnableGTID	bool	可选	是否附加全局事务*ID*，默认为true，为false时上传的数据将不附加。
EnableInsert	bool	可选	是否收集insert事件的数据，默认为true，为false时insert事件将不采集。
EnableUpdate	bool	可选	是否收集update事件的数据，默认为true，为false时update事件将不采集。
EnableDelete	bool	可选	是否收集delete事件的数据，默认为true，为false时delete事件将不采集。
EnableDDL	bool	可选	是否收集DDL（data definition language）事件的数。 说明： 该选项默认为false，为false时DDL事件将不采集。该选项不支持`IncludeTables`、`ExcludeTables`过滤。

参数	类型	必选或可选	参数说明
Charset	string	可选	编码方式，默认为`utf-8`。
TextToString	bool	可选	是否将`text`类型数据转换成string，默认为false。

使用限制

此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

- MySQL 必须开启Binlog，且Binlog必须为row模式（默认RDS已经开启）。

```
# 查看是否开启Binlog
mysql> show variables like "log_bin";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
1 row in set (0.02 sec)
# 查看Binlog类型
mysql> show variables like "binlog_format";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | ROW   |
+---------------+-------+
1 row in set (0.03 sec)
```

- ServerID 必须唯一，确保需同步的MySQL所有Slave的ID不重复。
- 需保证配置的用户具有需要采集的数据库读权限以及MySQL REPLICATION权限，示例如下：

```
CREATE USER canal IDENTIFIED BY 'canal';
GRANT SELECT, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'canal
'@'%';
-- GRANT ALL PRIVILEGES ON *.* TO 'canal'@'%' ;
FLUSH PRIVILEGES;
```

- 首次配置采集时如果不配置`StartBinName`，默认从当前时间点采集。

  如果想从指定位置采集，可以查看当前的Binlog以及offset，并将`StartBinName`、`StartBinLogPos`设置成对应的值，示例如下：

  说明：

当指定StartBinName时，第一次采集会产生较大开销。

```
# StartBinName 设置成 "mysql-bin.000063", StartBinLogPos 设置成 0
mysql> show binary logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000063 |       241 |
| mysql-bin.000064 |       241 |
| mysql-bin.000065 |       241 |
| mysql-bin.000066 |     10778 |
+------------------+-----------+
4 rows in set (0.02 sec)
```

- 如安全需求较高，建议将SQL访问用户名和密码配置为xxx，待配置同步至本地机器后，在/usr/local/ilogtail/user_log_config.json文件找到对应配置进行修改。

  说明：

  - 修改完毕后请执行以下命令重启Logtail：

    ```
    sudo /etc/init.d/ilogtaild stop; sudo /etc/init.d/ilogtaild start
    ```

    。
  - 如果您再次在Web端修改此配置，您的手动修改会被覆盖，请再次手动修改本地配置。

- **RDS** 相关限制：
  - 无法直接在RDS服务器上安装Logtail，您需要将Logtail安装在能连通RDS实例的任意一台ECS上。
  - RDS 只读备库当前不支持Binlog采集，您需要配置主库进行采集。

操作步骤

从RDS中同步user_info库中不以_inner结尾的表，配置如下。

1. 选择输入源。

   单击**数据接入向导**图标或**创建配置**，进入数据接入向导。并在**选择数据库类型**步骤中选择**BINLOG**。

2. 填写输入配置。

   进入**输入源配置**页面，填写**插件配置**。

   插件配置输入框中已为您提供配置模板，请根据您的需求替换配置参数信息。

inputs部分为采集配置，是必选项；processors部分为处理配置，是可选项。采集配置部分需要按照您的数据源配置对应的采集语句，处理配置部分请参考插件-处理采集数据配置一种或多种采集方式。

说明：

- 若安全需求较高，建议将SQL访问用户名/密码配置为xxx，待配置同步至本地机器后，在/usr/local/ilogtail/user_log_config.json文件找到对应配置进行修改。
- 请将Host/User/Password/Port替换为实际访问参数。

示例配置如下：

```
{
 "inputs": [
     {
         "type": "service_canal",
         "detail": {
             "Host": "************.mysql.rds.aliyuncs.com",
             "User" : "root",
             "ServerID" : 56321,
             "Password": "********",
             "IncludeTables": [
                 "user_info\\..*"
             ],
             "ExcludeTables": [
                 ".*\\.\\S+_inner"
             ],
             "TextToString" : true,
             "EnableDDL" : true
         }
     }
 ]
}
```

3. 应用到机器组。

   进入应用到机器组页面。请在此处勾选运行有此插件的**Logtail**机器组。

   如您之前没有创建过机器组，单击**+**创建机器组以创建一个新的机器组。

   说明：

   Binlog采集只需要一台安装Logtail的机器，您的机器组中只需要配置一个服务器IP即可。如您的机器组中有多台机器，请不要配置一样的ServerID。

4. 修改本地配置。

   如果您没有在输入源配置页面输入真实的URL、账号、密码等信息，需要在采集配置下发到本地后手动修改其中的内容。

> 说明：
> 如果您服务端输入的是真实信息，则无需此步骤。

a. 登录Logtail所在服务器，查找`/usr/local/ilogtail/user_log_config.json`文件中`service_canal`关键词，修改下述对应的`Host`、`User`、`Password`等字段。

b. 执行以下命令重启Logtail。

```
sudo /etc/init.d/ilogtaild stop; sudo /etc/init.d/ilogtaild start
```

Binlog采集配置已经完成，如果您的数据库有在进行对应的修改操作，则Logtail会立即将变化的数据采集到日志服务。

> 说明：
> Logtail默认采集Binlog的增量数据，如查看不到数据请确认在配置更新后数据库对应的表存在修改操作。

示例

例如，按照以上操作步骤配置处理方式后，对`user_info`下的`SpecialAlarm`表分别执行`INSERT`、`UPDATE`、`DELETE`操作。数据库表结构、数据库操作及Logtail采集的样例如下。

- 表结构

```
CREATE TABLE `SpecialAlarm` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`time` datetime NOT NULL,
`alarmtype` varchar(64) NOT NULL,
`ip` varchar(16) NOT NULL,
`count` int(11) unsigned NOT NULL,
PRIMARY KEY (`id`),
KEY `time` (`time`) USING BTREE,
KEY `alarmtype` (`alarmtype`) USING BTREE
) ENGINE=MyISAM AUTO_INCREMENT=1;
```

- 数据库操作

对数据库执行INSERT、DELETE和UPDATE三种操作：

```
insert into specialalarm (`time`, `alarmType`, `ip`, `count`) values
(now(), "NO_ALARM", "10.10.**.***", 55);
delete from specialalarm where id = 4829235  ;
update specialalarm set ip = "10.11.***.**" where id = "4829234";
```

并为`zc.specialalarm`创建一个索引：

```
ALTER TABLE `zc`.`specialalarm`
```

```
ADD INDEX `time_index` (`time` ASC);
```

- 样例日志

  在数据预览或在查找页面中，可以看到对应每种操作的样例日志如下：

  - INSERT语句

    ```
    __source__:  10.30.**.**
    __tag__:__hostname__:  iZbp145dd9fccu*****
    __topic__:
    _db_:  zc
    _event_:  row_insert
    _gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:536
    _host_:  *********.mysql.rds.aliyuncs.com
    _id_:  113
    _table_:  specialalarm
    alarmtype:  NO_ALARM
    count:  55
    id:  4829235
    ip:  10.10.22.133
    time:  2017-11-01 12:31:41
    ```

  - DELETE语句

    ```
    __source__:  10.30.**.**
    __tag__:__hostname__:  iZbp145dd9fccu****
    __topic__:
    _db_:  zc
    _event_:  row_delete
    _gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:537
    _host_:  *********.mysql.rds.aliyuncs.com
    _id_:  114
    _table_:  specialalarm
    alarmtype:  NO_ALARM
    count:  55
    id:  4829235
    ip:  10.10.22.133
    time:  2017-11-01 12:31:41
    ```

  - UPDATE语句

    ```
    __source__:  10.30.**.**
    __tag__:__hostname__:  iZbp145dd9fccu****
    __topic__:
    _db_:  zc
    _event_:  row_update
    _gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:538
    _host_:  *********.mysql.rds.aliyuncs.com
    _id_:  115
    _old_alarmtype:  NO_ALARM
    _old_count:  55
    _old_id:  4829234
    _old_ip:  10.10.22.133
    _old_time:  2017-10-31 12:04:54
    _table_:  specialalarm
    alarmtype:  NO_ALARM
    count:  55
    id:  4829234
    ```

```
ip:  10.11.145.98
time:  2017-10-31 12:04:54
```

- DDL（data definition language）语句

```
__source__:  10.30.**.**
__tag__:__hostname__:  iZbp145dd9fccu****
__topic__:
_db_:  zc
_event_:  row_update
_gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:539
_host_:  *********.mysql.rds.aliyuncs.com
ErrorCode:  0
ExecutionTime:  0
Query:  ALTER TABLE `zc`.`specialalarm`
ADD INDEX `time_index` (`time` ASC)
StatusVars:
```

2.3.8 插件-JDBC查询结果

以SQL形式定期采集数据库中的数据，根据SQL可实现各种自定义采集方式。

说明：

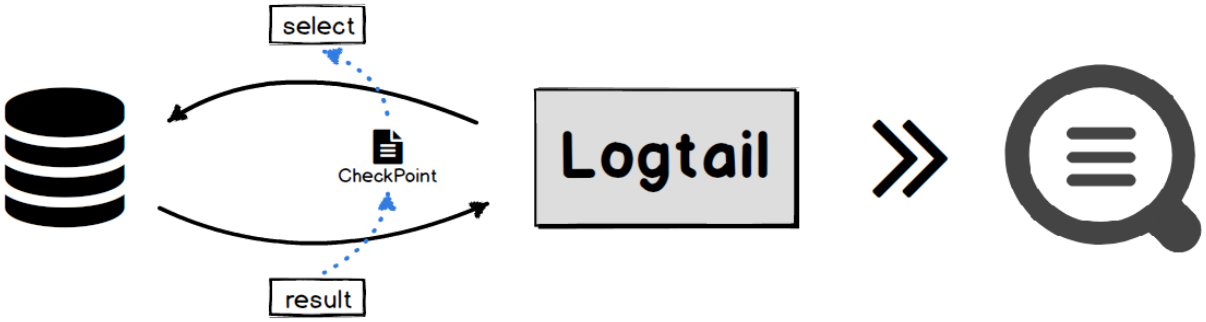
此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

功能

- 支持提供MySQL接口的数据库，包括RDS
- 支持分页设置
- 支持时区设置
- 支持超时设置
- 支持checkpoint状态保存
- 支持SSL
- 支持每次最大采集数量限制

实现原理

图 **2-14:** 实现原理



如上图所示，Logtail内部根据用户配置定期执行指定的SELECT语句，将SELECT返回的结果作为数据上传到日志服务。

Logtail在获取到执行结果时，会将结果中配置的CheckPoint字段保存在本地，当下次执行SQL时，会将上一次保存的CheckPoint带入到SELECT语句中，以此实现增量数据采集。

应用场景

- 根据自增ID或时间等标志进行增量数据同步。
- 自定义筛选同步。

参数说明

该输入源类型为：`service_mysql`。

参数	类型	必选或可选	参数说明
Address	string	可选	MySQL地址，默认为”127.0.0.1:3306”。
User	string	可选	数据库用户名，默认为”root”。
Password	string	可选	数据库密码，默认为空。
DialTimeOutMs	int	可选	数据库连接超时时间，单位为ms，默认为5000ms。

参数	类型	必选或可选	参数说明
ReadTimeOutMs	int	可选	数据库连接超时时间，单位为ms，默认为5000ms。
StateMent	string	必选	SQL语句。
Limit	bool	可选	是否使用Limit分页，默认为false。
PageSize	int	可选	分页大小，Limit为true时必须配置。
MaxSyncSize	int	可选	每次同步最大记录数，为0表示无限制，默认为0。
CheckPoint	bool	可选	是否使用checkpoint，默认为false。
CheckPointColumn	string	可选	checkpoint列名称，CheckPoint为true时必须配置。
CheckPoint ColumnType	string	可选	checkpoint列类型，支持`int`和`time`两种类型。
CheckPointStart	string	可选	checkpoint初始值。
CheckPoint SavePerPage	bool	可选	为true时每次分页时保存一次checkpoint，为false时每次同步完后保存checkpoint。
IntervalMs	int	必选	同步间隔，单位为ms。

使用限制

- 建议使用`Limit`分页，使用`Limit`分页时，SQL查询会自动在`StateMent`后追加LIMIT语句。
- 使用`CheckPoint`时，`StateMent`中SELECT出的数据中必须包含checkpoint列，且where条件中必须包含checkpoint列，该列的值填?

  例如checkpoint为”id”，`StateMent`为`SELECT * from ... where id > ?`。

- `CheckPoint`为true时必须配置`CheckPointColumn`、`CheckPointColumnType`、`CheckPointStart`。
- `CheckPointColumnType`只支持`int`和`time`类型，`int`类型内部存储为int64，`time`支持MySQL的date、datetime、time。

操作步骤

从MySQL中增量同步logtail.VersionOs中count > 0的数据，同步间隔为10s，checkpoint起始时间为2017-09-25 11:00:00，请求方式为分页请求，每页100，每次分页后保存checkpoint。具体配置步骤如下。

1. 选择输入源。

   单击**数据接入向导**图标或**创建配置**，进入数据接入向导。并在**选择数据库类型**步骤中选择**MYSQL查询结果**。

2. 填写输入配置。

   进入**输入源配置**页面，填写**插件配置**。

   插件配置输入框中已为您提供配置模板，请根据您的需求替换配置参数信息。

   `inputs`部分为采集配置，是必选项；`processors`部分为处理配置，是可选项。采集配置部分需要按照您的数据源配置对应的采集语句，处理配置部分请参考插件-处理采集数据配置一种或多种采集方式。

   说明：

   若安全需求较高，建议将SQL访问用户名/密码配置为xxx，待配置同步至本地机器后，在/usr/local/ilogtail/user_log_config.json文件找到对应配置进行修改。

   示例配置如下：

```
{
 "inputs": [
     {
         "type": "service_mysql",
         "detail": {
             "Address": "********:3306",
             "User": "logtail",
             "Password": "********",
             "DataBase": "logtail",
             "Limit": true,
             "PageSize": 100,
             "StateMent": "SELECT * from logtail.VersionOs where
time > ?",
             "CheckPoint": true,
             "CheckPointColumn": "time",
```

```
            "CheckPointStart": "2017-09-25 11:00:00",
            "CheckPointSavePerPage": true,
            "CheckPointColumnType": "time",
            "IntervalMs": 10000
        }
    }
 ]
}
```

3. 应用到机器组。

   进入**应用到机器组**页面。请在此处勾选**支持此插件的Logtail机器组**。

   如您之前没有创建过机器组，单击**+创建机器组**以创建一个新的机器组。

4. 修改本地配置。

   如果您没有在**输入源配置**页面输入真实的URL、账号、密码等信息，需要在采集配置下发到本地后手动修改其中的内容。

   说明：

   如果您服务端输入的是真实信息，则无需此步骤。

   a. 登录Logtail所在服务器，查找`/usr/local/ilogtail/user_log_config.json`文件中`service_mysql`关键词，修改下述对应的`Address`、`User`、`Password`等字段。

   b. 执行以下命令重启Logtail。

   ```
   sudo /etc/init.d/ilogtaild stop; sudo /etc/init.d/ilogtaild start
   ```

示例

例如，按照以上操作步骤配置处理方式后，即可在日志服务控制台查看采集处理过的日志数据。表结构和Logtail采集的日志样例如下。

- 表结构

```
CREATE TABLE `VersionOs` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT COMMENT 'id',
  `time` datetime NOT NULL,
  `version` varchar(10) NOT NULL DEFAULT '',
  `os` varchar(10) NOT NULL,
  `count` int(11) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `timeindex` (`time`)
)
```

- 样例输出

```
"count":  "4"
"id:  "721097"
"os:  "Windows"
```

```
"time:  "2017-08-25 13:00:00"
"version":  "1.3.0"
```

2.3.9 插件 HTTP方式

HTTP输入可根据您的配置定期请求指定URL，将请求返回body值作为输入源上传到日志服务。

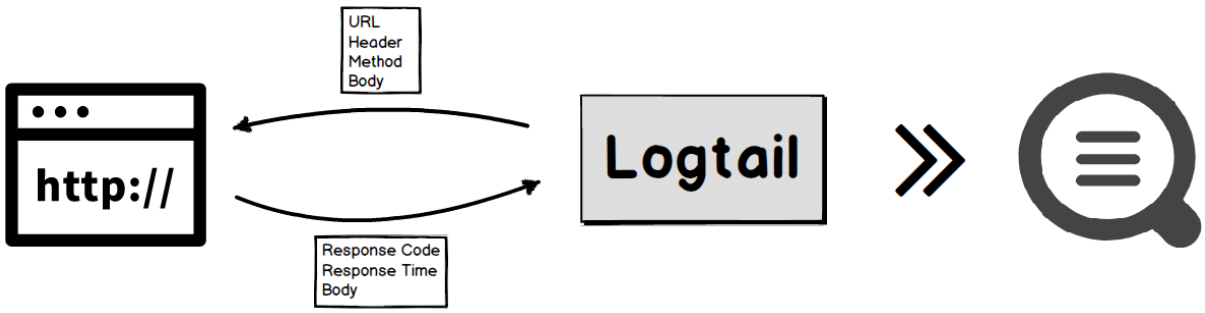
> 说明：
> 此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

功能

- 支持配置多个URL。
- 支持请求间隔配置。
- 支持HTTP方法配置。
- 支持自定义header。
- 支持设置method。
- 支持HTTPS。
- 支持检测body是否匹配固定pattern。

实现原理

图 **2-15:** 实现原理



如上图所示，Logtail内部会根据用户配置的HTTP请求url、method、header、body等信息定期对指定URL发起请求，将请求的返回的状态码、body内容以及响应时间做为数据上传到日志服务。

应用场景

- 应用状态监控（以HTTP方式提供监控接口），例如：
  - Nginx
  - Docker（以HTTP方式）

- Elastic Search
- Haproxy
- 其他以HTTP提供监控接口的服务

- 服务可用性检测，定期请求服务，通过状态码以及请求延迟做可用性监控。
- 数据定期拉取，例如微博评论、粉丝数等。

参数说明

参数说明

该输入源类型为：`metric_http`。

参数	类型	必选或可选	参数说明
Addresses	string 数组	必选	URL列表。 **说明：** 必须以`http`或`https`开头。
IntervalMs	int	必选	每次请求间隔，单位ms。
Method	string	可选	请求方法名，大写，默认为`GET`。
Body	string	可选	http body字段内容，默认为空。
Headers	key:string value:string map	可选	http header内容，默认为空。
PerAddressSleepMs	int	可选	Addresses列表中每个url请求间隔时间，单位ms，默认100ms。
ResponseTimeoutMs	int	可选	请求超时时间，单位ms，默认5000ms。
IncludeBody	bool	可选	是否采集请求的body，若为true，则将body以`content`为key存放，默认为false。

参数	类型	必选或可选	参数说明
FollowRedirects	bool	可选	是否自动处理重定向，默认为false。
InsecureSkipVerify	bool	可选	是否跳过https安全检查，默认为false。
ResponseStringMatch	string	可选	对于返回body进行正则表达式检查，检查结果以`_response_match_`为key存放：若匹配，value为`yes`；若不匹配，value为`false`。

使用限制

- 此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。
- URL必须以`http`或`https`开头。
- 目前不支持自定义证书。
- 不支持交互式通信方式。

默认字段

每次请求默认会上传以下字段。

字段名	说明	示例
`_address_`	请求地址。	“*http://127.0.0.1/ngx_sta*
`_method_`	请求方法。	“GET”
`_response_time_ms_`	响应延迟，单位为ms。	“1.320”
`_http_response_code_`	状态码。	“200”
`_result_`	是否成功，取值范围:`success`、`invalid_body`、`match_regex_invalid`、`mismatch`、`timeout`。	“success”
`_response_match_`	返回body是否匹配`ResponseStringMatch`字段，若不存在`ResponseStringMatch`字段则为空，取值范围:`yes`、`no`。	“yes”

操作步骤

每隔1000ms请求一次nginx status模块，URL为 `http://127.0.0.1/ngx_status`，将返回的body使用正则提取出其中的状态信息。详细配置如下：

1. 选择输入源。

   单击**数据接入向导**图标或**创建配置**，进入数据接入向导。并在**选择数据库类型**步骤中选择**Logtail**自定义插件。

2. 填写输入配置。

   进入**输入源配置**页面，填写**插件配置**。

   `inputs`部分为采集配置，是必选项；`processors`部分为处理配置，是可选项。采集配置部分需要按照您的数据源配置对应的采集语句，处理配置部分请参考插件-处理采集数据配置一种或多种采集方式。

   示例配置如下：

```
{
 "inputs": [
     {
         "type": "metric_http",
         "detail": {
             "IntervalMs": 1000,
             "Addresses": [
                 "http://127.0.0.1/ngx_status"
             ],
             "IncludeBody": true
         }
     }
 ],
 "processors" : [
     {
         "type": "processor_regex",
         "detail" : {
             "SourceKey": "content",
             "Regex": "Active connections: (\\d+)\\s+server accepts
 handled requests\\s+(\\d+)\\s+(\\d+)\\s+(\\d+)\\s+Reading: (\\d+)
Writing: (\\d+) Waiting: (\\d+).*",
             "Keys": [
                 "connection",
                 "accepts",
                 "handled",
                 "requests",
                 "reading",
                 "writing",
                 "waiting"
             ],
             "FullMatch": true,
             "NoKeyError": true,
             "NoMatchError": true,
             "KeepSource": false
         }
```

```
        }
    ]
}
```

3. 应用到机器组。

   进入应用到机器组页面。请在此处勾选**支持此插件的Logtail机器组**。

   如您之前没有创建过机器组，单击**+创建机器组**以创建一个新的机器组。

示例

按照以上操作步骤配置处理方式后，即可在日志服务控制台查看采集处理过的日志数据。

解析后上传的日志数据除包含正则解析后的数据外，还包括HTTP请求附加的method、address、time、code、result信息。

```
"Index" : "7"
"connection" : "1"
"accepts" : "6079"
"handled" : "6079"
"requests" : "11596"
"reading" : "0"
"writing" : "1"
"waiting" : "0"
"_method_" : "GET"
"_address_" : "http://127.0.0.1/ngx_status"
"_response_time_ms_" : "1.320"
"_http_response_code_" : "200"
"_result_" : "success"
```

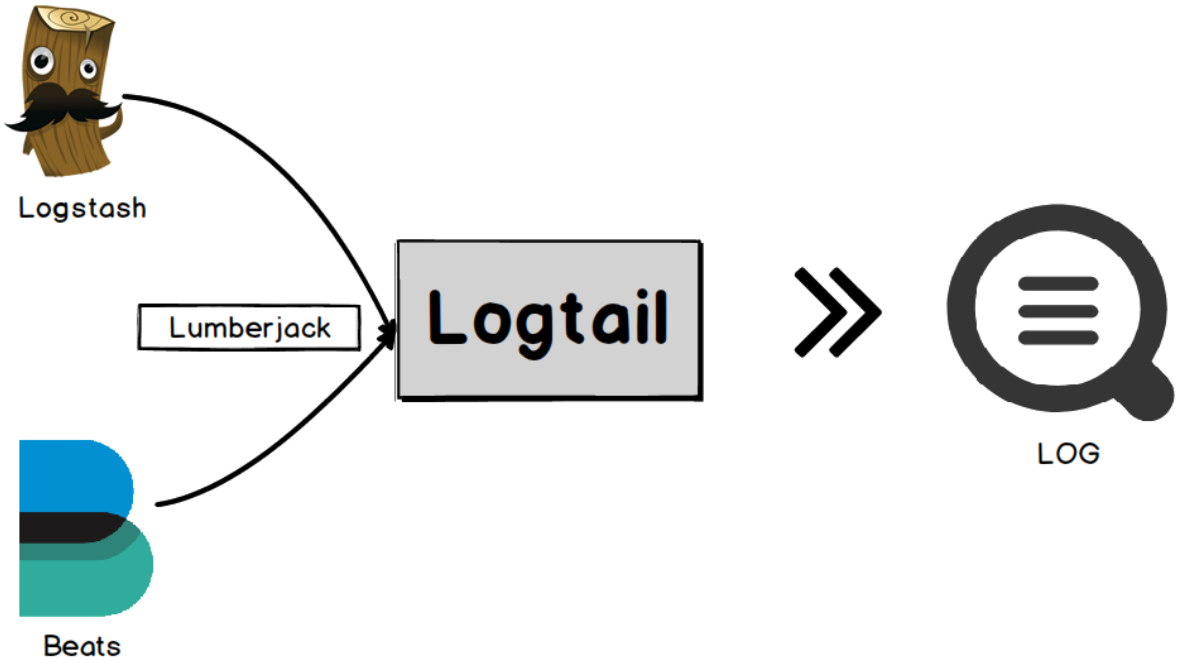
2.3.10 插件-Beats和Logstash输入源

简介

Logtail除支持syslog协议外，还支持Lumberjack协议作为数据输入，可支持将Beats系列软件（MetricBeat、PacketBeat、Winlogbeat、Auditbeat、Filebeat、Heartbeat等）、Logstash采集的数据转发到日志服务。

实现原理

图 2-16: 实现原理



基于Logstash、Beats系列软件对Lumberjack协议的支持，日志服务可以通过Logtail，同时对Logstash、Beats系列软件进行Logtail Lumberjack插件监听配置，实现数据采集。

配置项

该输入源类型为: `service_lumberjack`。

> 说明：
> Logtail支持对采集的数据进行处理加工后上传，处理方式参见数据处理配置。

配置项	类型	是否必须	说明
BindAddress	string	否	绑定地址，默认为`127.0.0.1:5044`，IP和端口号均可自定义。如需被局域网内其他主机访问，请设置为0.0.0.0:5044。
V1	bool	否	Lumberjack V1版本协议版，默认为false。目

配置项	类型	是否必须	说明
			前Logstash支持的协议版本为V1
V2	bool	否	Lumberjack V2版本协议版，默认为true。目前Beats系列软件支持的协议版本均为V2。
SSLCA	string	否	证书授权机构（Certificate Authority）颁发的签名证书路径，默认为空。如果是自签名证书可不设置此选项。
SSLCert	string	否	证书的路径，默认为空。
SSLKey	string	否	证书对应私钥的路径，默认为空。
InsecureSkipVerify	bool	否	是否跳过SSL安全检查，默认为false。

使用限制

- 该功能在Logtail 0.16.9及以上版本支持。
- 如需将Logstash采集的数据上传，请参考*Logstash-Lumberjack-Output*。
- 如需将Beats（MetricBeat、PacketBeat、Winlogbeat、Auditbeat、Filebeat、Heartbeat等）系列软件采集的数据上传，请参考*Beats-Lumberjack-Output*。
- 同一Logtail可配置多个Lumberjack插件，但多个插件不能监听同一端口。
- 该插件支持SSL，Logstash采集的数据上传需要使用该功能

操作步骤

使用PacketBeat采集本地网络数据包，并使用Logtail Lumberjack插件上传到日志服务。详细配置如下：

1. 选择输入源。

   单击数据接入向导图标或创建配置，进入数据接入向导。并在选择数据库类型步骤中选择Logtail自定义插件。

2. 填写输入配置。

   进入输入源配置页面，填写插件配置。

   inputs部分为采集配置，是必选项；processors部分为处理配置，是可选项，由于Beats和Logstash输出的都是JSON格式数据，因此我们使用`processor_anchor`将json展开。

   关于处理配置部分请参考插件-处理采集数据配置一种或多种采集方式。

```
{
  "inputs": [
    {
      "detail": {
        "BindAddress": "0.0.0.0:5044"
      },
      "type": "service_lumberjack"
    }
  ],
  "processors": [
    {
      "detail": {
        "Anchors": [
          {
            "ExpondJson": true,
            "FieldType": "json",
            "Start": "",
            "Stop": ""
          }
        ],
        "SourceKey": "content"
      },
      "type": "processor_anchor"
    }
  ]
}
```

3. 应用到机器组。

   进入应用到机器组页面。请在此处勾选支持此插件的Logtail机器组。

   如您之前没有创建过机器组，单击+创建机器组以创建一个新的机器组。

4. 配置PacketBeat。

   配置PacketBeat输出方式为`Logstash`，具体配置方式请参见*PacketBeat-Logstash-Output*。

   在本示例中。配置如下：

```
output.logstash:
```

```
  hosts: ["127.0.0.1:5044"]
```

示例

按照以上操作步骤配置处理方式后，可以尝试在本机输入命令`ping 127.0.0.1`，即可在日志服务控制台查看采集处理过的日志数据。

```
_@metadata_beat:  packetbeat
_@metadata_type:  doc
_@metadata_version:  6.2.4
_@timestamp:  2018-06-05T03:58:42.470Z
__source__:  **.**.**.**
__tag__:__hostname__:  *******
__topic__:
_beat_hostname:  bdbe0b8d53a4
_beat_name:  bdbe0b8d53a4
_beat_version:  6.2.4
_bytes_in:  56
_bytes_out:  56
_client_ip:  192.168.5.2
_icmp_request_code:  0
_icmp_request_message:  EchoRequest(0)
_icmp_request_type:  8
_icmp_response_code:  0
_icmp_response_message:  EchoReply(0)
_icmp_response_type:  0
_icmp_version:  4
_ip:  127.0.0.1
_path:  127.0.0.1
_responsetime:  0
_status:  OK
_type:  icmp
```

2.3.11 插件-Syslog输入源

Logtail支持通过自定义插件采集syslog。

简介

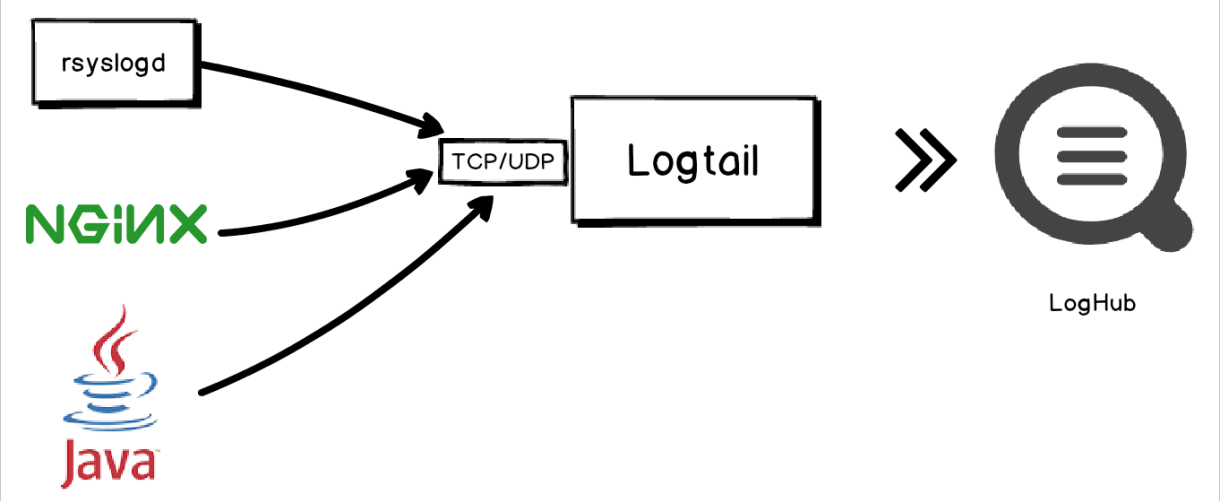
在Linux上，本地的syslog数据可以通过rsyslog等syslog agent转发到指定服务器IP地址和端口。为指定服务器添加Logtail配置之后，Logtail插件会以TCP或UDP协议接收转发过来的syslog数据。并且，插件能够将接收到的数据按照指定的syslog协议进行解析，提取日志中的facility、tag(program)、severity、content等字段。syslog协议支持*RFC3164*和*RFC5424*。

说明：

Windows Logtail不支持该插件。

实现原理

通过插件对指定的地址和端口进行监听后，Logtail能够作为syslog服务器采集来自各个数据源的日志，包括通过rsyslog采集的系统日志、 *Nginx*转发的访问日志或错误日志，以及*Java*等语言的syslog客户端库转发的日志。



注意事项

- Linux版 Logtail 0.16.13及以上版本支持该功能。
- Logtail可同时配置多个syslog插件，比如同时使用TCP和UDP监听127.0.0.1:9999。

Logtail配置项

该插件的输入类型为：`service_syslog`。

配置项	类型	是否必须	说明
Address	string	否	指定插件监听的协议、地址和端口，Logtail插件会根据配置进行监听并获取日志数据。格式为`[tcp/udp]://[ip]:[port]`，默认为`tcp://127.0.0.1:9999`。 **说明：** • Logtail插件配置监听的协议、地址和端口号必须与rsyslog配置文件设置的转发规则相同。 • 如果安装Logtail的服务器有多个IP地址可接收日志，可以将地址配置为0.0.0.0，表示监听服务器的所有IP地址。

配置项	类型	是否必须	说明
ParseProtocol	string	否	指定解析所使用的协议，默认为空，表示不解析。其中： • rfc3164：指定使用RFC3164协议解析日志。 • rfc5424：指定使用RFC5424协议解析日志。 • auto：指定插件根据日志内容自动选择合适的解析协议。
IgnoreParseFailure	boolean	否	指定解析失败后的行为，默认为true。其中： • true：放弃解析直接填充所返回的content字段。 • false：会丢弃日志。

默认字段

字段名	字段类型	字段含义
hostname	string	主机名，如果日志中未提供则获取当前主机名。
program	string	对应协议中的tag字段。
priority	string	对应协议中的priority字段。
facility	string	对应协议中的facility字段。
severity	string	对应协议中的severity字段。
unixtimestamp	string	日志对应的时间戳。
content	string	日志内容，如果解析失败的话，此字段包含未解析日志的所有内容。
_ip_	string	当前主机的IP地址。

前提条件

- 已创建Project和Logstore。
- 已创建机器组，并在机器组内服务器上安装了0.16.13及以上版本的Logtail。
- 需要被采集syslog的服务器上已安装了rsyslog。

配置**Logtail**插件采集**syslog**

1. 为rsyslog添加一条转发规则。

在需要采集syslog的服务器上修改rsyslog的配置文件*/etc/rsyslog.conf*，在配置文件的最后添加一行转发规则。添加转发规则后，rsyslog会将syslog转发至指定地址端口。

- 通过当前服务器采集本机syslog：配置转发地址为127.0.0.1，端口为任意非知名的空闲端口。
- 通过其他服务器采集本机syslog：配置转发地址为其他服务器的公网IP，端口为任意非知名的空闲端口。

例如以下配置表示将所有的日志都通过TCP转发至127.0.0.1:9000，配置文件详细说明请参考官网说明。

```
*.* @@127.0.0.1:9000
```

2. 执行以下命令重启rsyslog，使日志转发规则生效。

```
sudo service rsyslog restart
```

3. 登录日志服务控制台，单击Project名称。
4. 在**Logstore**列表页面单击**数据接入向导**图标。
5. 在**自定义数据**中单击**Logtail自定义插件**。
6. 填写**配置名称**。
7. 填写**插件配置**，并单击**下一步**。

`inputs`部分为采集配置，是必选项；`processors`部分为处理配置，是可选项。采集配置部分需要按照您的数据源配置对应的采集语句，处理配置部分请参考插件-处理采集数据配置一种或多种采集方式。

同时监听UDP和TCP的示例配置如下：

```
{
    "inputs": [
        {
            "type": "service_syslog",
            "detail": {
                "Address": "tcp://127.0.0.1:9000",
                "ParseProtocol": "rfc3164"
            }
        },
        {
            "type": "service_syslog",
            "detail": {
                "Address": "udp://127.0.0.1:9001",
                "ParseProtocol": "rfc3164"
            }
        }
    ]
```

```
}
```

8. 应用到机器组。

   为机器组添加Logtail配置。请在此处勾选运行有此插件的**Logtail**机器组，并单击应用到机器组。

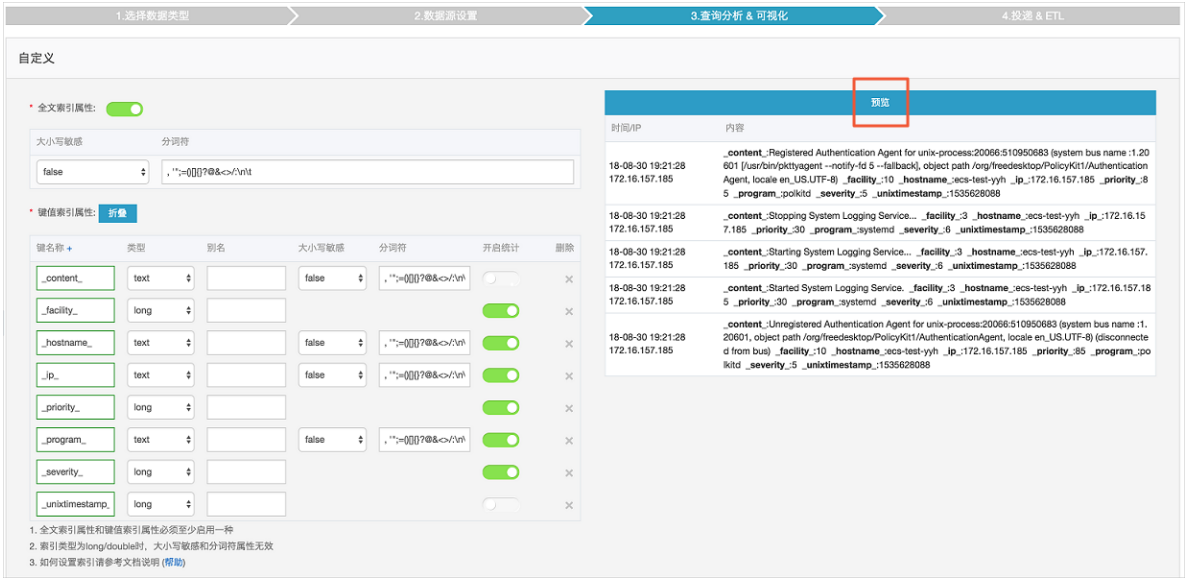
   > **说明：**
   > 接收syslog的服务器必须在机器组中，且安装了0.16.13及以上版本的Logtail。

   若您之前没有创建过机器组，单击**+创建机器组**创建一个新的机器组，再勾选机器组并单击**应用到机器组**。

9. 开启并配置索引。

   确保日志机器组心跳正常的情况下，可以点击右侧浏览按钮获取到采集上来的数据。

   如果您后续需要对采集到的syslog进行实时查询与分析，可以单击**展开**，根据日志内容和格式配置键值索引，后续可以按照指定字段进行查询和分析。



配置Logtail插件采集Nginx访问日志

Nginx支持直接把访问日志以syslog协议转发到指定地址和端口。如果您希望把服务器上包括Nginx访问日志在内的所有数据都以syslog的形式集中投递到日志服务，可以创建Logtail配置并应用到该服务器所在的机器组。

1. 在Nginx服务器的 *nginx.conf* 文件中增加转发规则。

   关于Nginx配置文件的详细信息请参考*Nginx*官网说明。

   例如，在配置文件中增加如下内容：

   ```
   http {
       ...

       # Add this line.
       access_log syslog:server=127.0.0.1:9000,facility=local7,tag=
   nginx,severity=info combined;

       ...
   }
   ```

2. 执行以下命令重启Nginx服务，使配置生效。

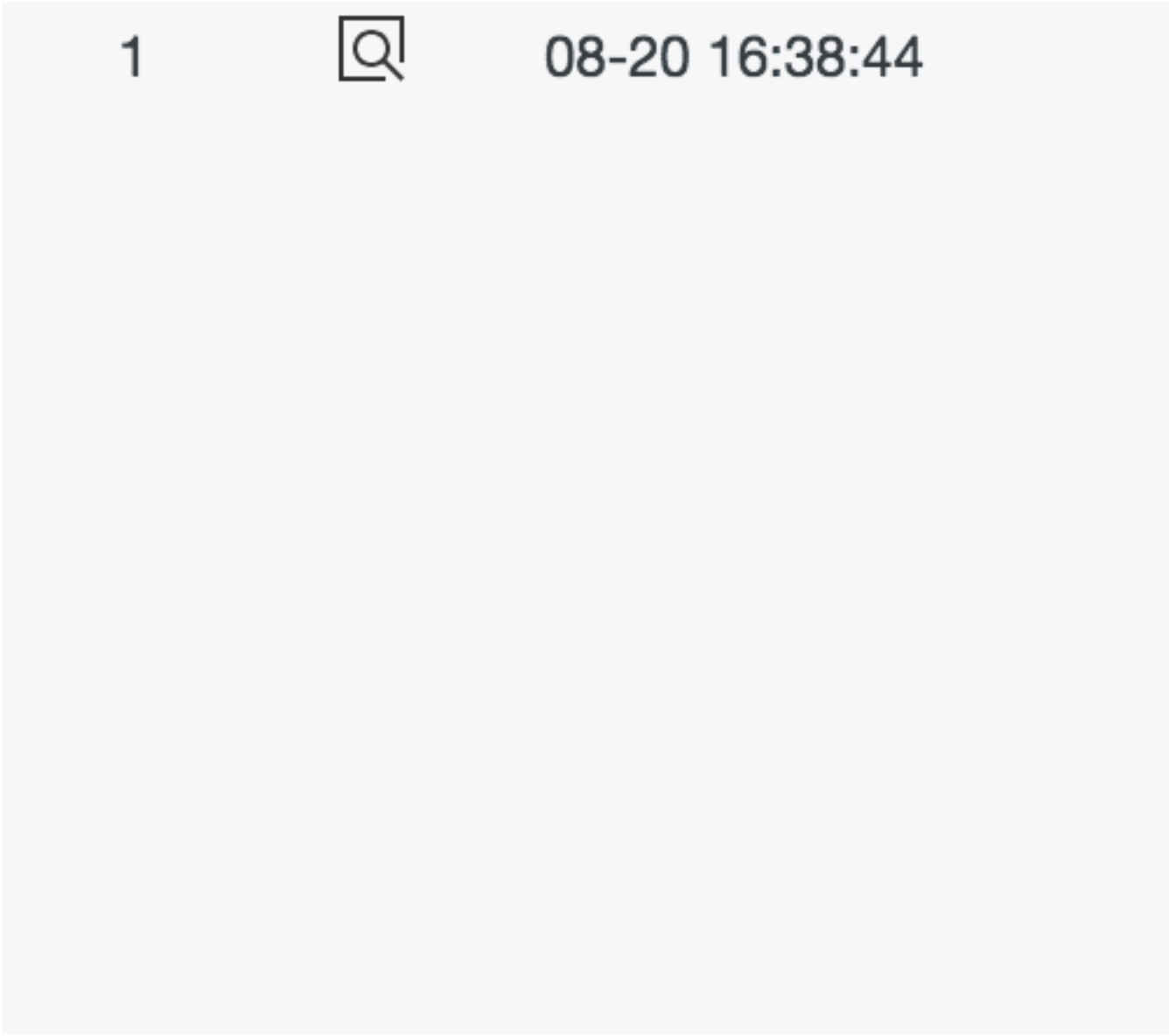
   ```
   sudo service nginx restart
   ```

3. 创建Logtail配置并应用到该服务器所在的机器组。

   配置过程请参考配置*Logtail*插件采集*syslog*。

4. 检验Logtail配置是否生效。

   在shell中执行命令`curl http://127.0.0.1/test.html`生成一条访问日志。如果采集配置已生效，可以在日志服务控制台的查询页面查看到日志信息。



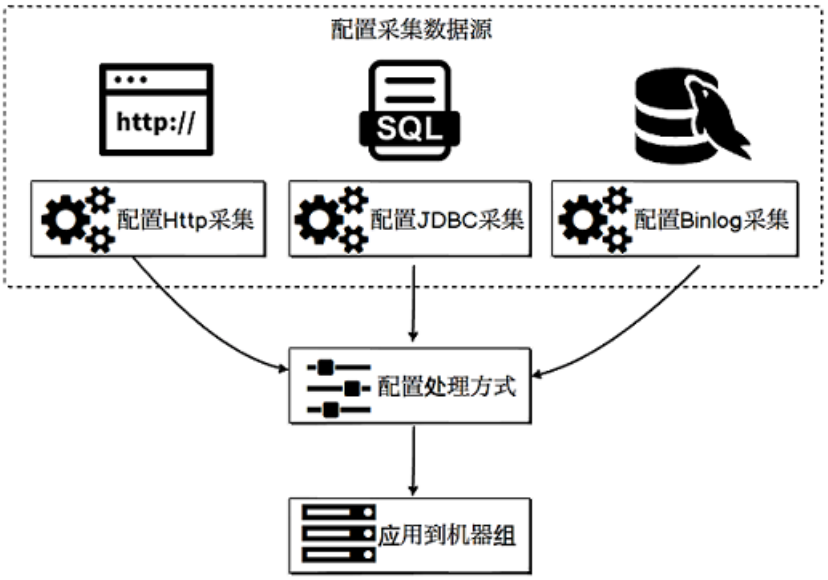
2.3.12 插件-处理采集数据

Logtail对于各类非文本的输入源提供了统一的数据处理配置，用户可对一个输入源配置多个处理方式，Logtail会根据配置顺序逐一执行各个处理方式。

说明：

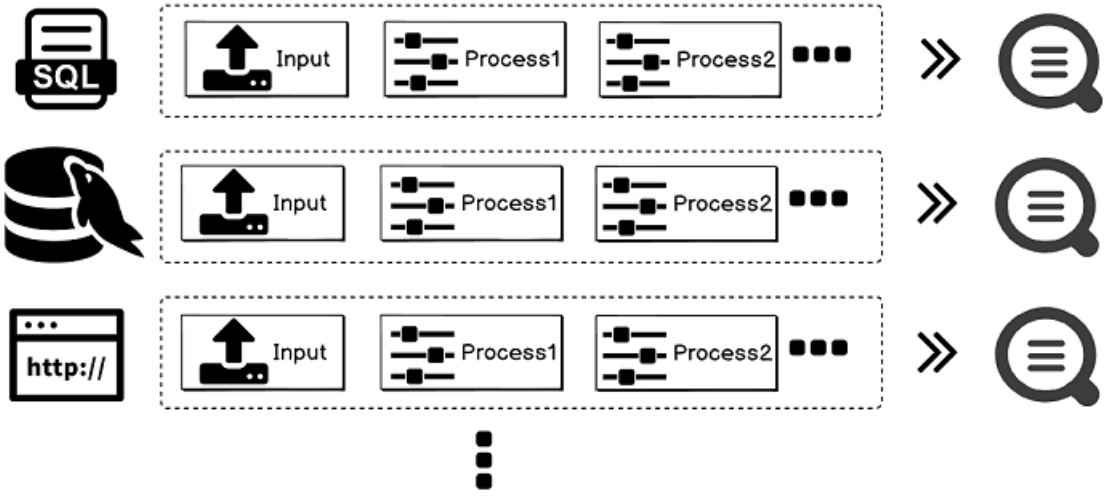
目前数据处理配置只适用于Binlog、MySQL查询结果、Nginx监控和HTTP输入源。

图 2-17: 采集过程



实现原理

图 2-18: 实现原理



处理方式

目前支持的处理方式如下：

- 正则提取
- 标定提取
- 单字分隔符
- 多字符分隔符

- *GeoIP*转换
- 正则过滤

您也可以根据以上处理方式，为您的输入源定制组合配置。

使用说明

为采集数据配置处理方式，处理配置的key为`processors`，value为json object的数组，数组内每个object代表一个处理方式配置。

单一处理方式包含两个字段:`type`、`detail`，其中`type`为该处理方式的类型，`detail`内部为该处理方式的详细配置。

```
"processors" : [
        {
            "type" : "processor_anchor",
            "detail" : {
                ...
            }
        },
        {
            "type" : "processor_regex",
            "detail" : {
                ...
            }
        }
    ]
```

正则提取

该方式通过对指定字段进行正则表达式匹配来提取其中匹配的字段。

参数说明

正则提取方式的类型（type）为`processor_regex`。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行正则提取的字段。
Regex	string	必选	用于匹配的正则表达式，需要提取的字段使用()标注，详细内容请参考维基百科。
Keys	string 数组	必选	需要提取的字段名，例如[“key1”, “key2” …]。

参数	类型	必选或可选	参数说明
NoKeyError	bool	可选	默认为false，为true时若没有找到SourceKey字段则报错。
NoMatchError	bool	可选	默认为false，为true时若正则不匹配则报错。
KeepSource	bool	可选	默认为false，为true时匹配完后不丢弃SourceKey字段。
FullMatch	bool	必选	默认为true，为true时只有字段完全匹配Regex时才会进行提取，为false时部分字段匹配也会进行提取。

示例

配置提取Access日志，详细配置如下：

- 输入

```
"content" : "10.200.**.** - - [10/Aug/2017:14:57:51 +0800] \"POST /
PutData?
Category=YunOsAccountOpLog&AccessKeyId=***************&Date=Fri
%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature
=*****************************
 HTTP/1.1\" 0.024 18204 200 37 \"-\" \"aliyun-sdk-java"
```

- 配置详情

```
{
    "type" : "processor_regex",
    "detail" : {"SourceKey" : "content",
         "Regex" : "([\\d\\.]+) \\S+ \\S+ \\[(\\S+) \\S+\\] \"(\\
w+) ([^\\\"]*)\" ([\\d\\.]+) (\\d+) (\\d+) (\\d+|-) \"([^\\\"]*)\"
 \"([^\\\"]*)\" (\\d+)",
         "Keys"   : ["ip", "time", "method", "url", "request_time",
 "request_length", "status", "length", "ref_url", "browser"],
         "NoKeyError" : true,
         "NoMatchError" : true,
         "KeepSource" : false
    }
```

```
}
```

- 处理后结果

```
"ip" : "10.200.**.**"
"time" : "10/Aug/2017:14:57:51"
"method" : "POST"
"url" : "/PutData?Category=YunOsAccountOpLog&AccessKeyId
=****************&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT
&Topic=raw&Signature=********************************"
"request_time" : "0.024"
"request_length" : "18204"
"status" : "200"
"length" : "27"
"ref_url" : "-"
"browser" : "aliyun-sdk-java"
```

标定提取

该方式通过标定指定字段起始和结束关键字进行提取，标定后的字符串支持直接提取和JSON展开方式。

参数说明

标定提取方式的类型（type）为`processor_anchor`。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行提取的字段。
Anchors	**Anchor** 数组	必选	标定项列表，具体参见下述表格。
NoAnchorError	bool	可选	默认为false，为true时若查找不到关键字则报错。
NoKeyError	bool	可选	默认为false，为true时若没有找到`SourceKey`字段则报错。
KeepSource	bool	可选	默认为false，为true时匹配完后不丢弃`SourceKey`字段。

Anchor类型说明

参数	类型	必选或可选	参数说明
Start	string	必选	起始关键字，若为空则代表匹配字符串开头。
Stop	string	必选	结束关键字，若为空则代表匹配字符串结尾。
FieldName	string	必选	提取的字段名。
FieldType	string	必选	提取字段类型，支持”string”、”json”两种类型。
ExpondJson	bool	可选	默认为false，为true且`FieldType`为`json`时将提取的json逐层展开。
ExpondConnecter	string	可选	JSON展开的连接符，默认为`_`。
MaxExpondDepth	int	可选	JSON展开最大深度，默认为`0`（无限制）。

示例

如下配置对某混合类型输入的处理结果如下：

- 输入

```
"content" : "time:2017.09.12 20:55:36\tjson:{\"key1\" : \"xx\", \"
key2\": false, \"key3\":123.456, \"key4\" : { \"inner1\" : 1, \"
inner2\" : false}}"
```

- 配置详情

```
{
   "type" : "processor_anchor",
   "detail" : {"SourceKey" : "content",
      "Anchors" : [
          {
              "Start" : "time",
              "Stop" : "\t",
              "FieldName" : "time",
              "FieldType" : "string",
              "ExpondJson" : false
          },
          {
              "Start" : "json:",
              "Stop" : "",
```

```
                "FieldName" : "val",
                "FieldType" : "json",
                "ExpondJson" : true
            }
        ]
    }
}
```

- 处理后结果

```
"time" : "2017.09.12 20:55:36"
"val_key1" : "xx"
"val_key2" : "false"
"val_key3" : "123.456"
"value_key4_inner1" : "1"
"value_key4_inner2" : "false"
```

单字符分隔符

该方式通过指定分隔符对字段进行分割，可指定Quote进行分隔符屏蔽。

参数说明

单字分隔符方式的类型（type）为`processor_split_char`。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行提取的字段。
SplitSep	string	必选	分隔符，必须为单字符，可以设置不可见字符，例如`\u0001`。
SplitKeys	string 数组	必选	切分后的字段名，例如[“key1”, “key2”…]。
QuoteFlag	bool	可选	默认为false，为true时代表使用quote。
Quote	string	可选	必须为单字符，QuoteFlag为true时生效，可以设置不可见字符，例如`\u0001`。
NoKeyError	bool	可选	默认为false，为true时若没有找到`SourceKey`字段则报错。

参数	类型	必选或可选	参数说明
NoMatchError	bool	可选	默认为false，为true时若切分。
KeepSource	bool	可选	默认为false，为true时匹配完后不丢弃SourceKey字段。

示例

对分隔符数据输入配置单字符分隔符处理方式，配置详情及处理结果如下：

- 输入

```
"content" : "10.**.**.**|10/Aug/2017:14:57:51 +0800|POST|PutData?
Category=YunOsAccountOpLog&AccessKeyId=***************&Date=Fri
%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature
=*************************|0.024|18204|200|37|-|
aliyun-sdk-java"
```

- 配置详情

```
{
   "type" : "processor_split_char",
   "detail" : {"SourceKey" : "content",
      "SplitSep" : "|",
      "SplitKeys" : ["ip", "time", "method", "url", "request_time",
 "request_length", "status", "length", "ref_url", "browser"]
  }
}
```

- 处理后结果

```
"ip" : "10.**.**.**"
"time" : "10/Aug/2017:14:57:51 +0800"
"method" : "POST"
"url" : "/PutData?Category=YunOsAccountOpLog&AccessKeyId
=***************&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT
&Topic=raw&Signature=********************************"
"request_time" : "0.024"
"request_length" : "18204"
"status" : "200"
"length" : "27"
"ref_url" : "-"
"browser" : "aliyun-sdk-java"
```

多字符分隔符

和单字符分隔符类似，多字符分隔符不支持Quote，完全按照分隔符拆分日志。

参数说明

多字符分隔符方式类型（type）为processor_split_string。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行提取的字段。
SplitSep	string	必选	分隔符，可以设置不可见字符，例如`\u0001\u0002`。
SplitKeys	string 数组	必选	切分后的字段名，例如[“key1”, “key2”…]。
PreserveOthers	bool	可选	默认为false, 为true时若分割的字段大于`SplitKeys`长度会保留超出部分。
ExpandOthers	bool	可选	默认为false，为true时继续解析超出部分。
ExpandKeyPrefix	string	可选	超出部分命名前缀，例如配置的`expand_`，则key为`expand_1`、`expand_2`。
NoKeyError	bool	可选	默认为false，为true时若没有找到`SourceKey`字段则报错。
NoMatchError	bool	可选	默认为false，为true时若切分。
KeepSource	bool	可选	默认为false，为true时匹配完后不丢弃`SourceKey`字段。

示例

对分隔符数据输入采用多字符分隔符方式处理，配置详情及处理结果如下：

- 输入

```
"content" : "10.**.**.**|#|10/Aug/2017:14:57:51 +0800|#|POST|#|
PutData?
Category=YunOsAccountOpLog&AccessKeyId=***************&Date=Fri
%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature
=*******************************|#|0.024|#|18204|#|200|#|37|#|-|#|
```

```
aliyun-sdk-java"
```

- 配置详情

```
{
   "type" : "processor_split_string",
   "detail" : {"SourceKey" : "content",
      "SplitSep" : "|#|",
      "SplitKeys" : ["ip", "time", "method", "url", "request_time",
 "request_length", "status"],
      "PreserveOthers" : true,
      "ExpandOthers" : true,
      "ExpandKeyPrefix" : "expand_"
  }
}
```

- 处理后结果

```
"ip" : "10.**.**.**"
"time" : "10/Aug/2017:14:57:51 +0800"
"method" : "POST"
"url" : "/PutData?Category=YunOsAccountOpLog&AccessKeyId
=******************&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%
20GMT&Topic=raw&Signature=********************************"
"request_time" : "0.024"
"request_length" : "18204"
"status" : "200"
"expand_1" : "27"
"expand_2" : "-"
"expand_3" : "aliyun-sdk-java"
```

GeoIP转换

GeoIP转换对数据输入中的IP进行地理位置转换，能够将IP转换成：国家、省份、城市、经纬度。

 说明：

Logtail安装包本身并不带有GeoIP的数据库，需要您手动下载到本地并配置，建议下载精确到`City`的数据库。

参数说明

GeoIP转换插件类型（type）为`processor_geoip`。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行IP转换的字段。
DBPath	string	必选	GeoIP数据库的全路径，数据库格式为`mmdb`，例如`/user/data/GeoLite2`

参数	类型	必选或可选	参数说明
			`-City_20180102/GeoLite2-City.mmdb`。
NoKeyError	bool	可选	默认为false，为true时若没有找到`SourceKey`字段则报错。
NoMatchError	bool	可选	默认为false，为true时若IP地址无效或数据库中未匹配到该IP则上报错误。
KeepSource	bool	可选	默认为true，为false时转换完毕后丢弃`SourceKey`字段。
Language	string	可选	语言，默认为`zh-CN`，需确保您的GeoIP数据库中包含相应的语言。

示例

对输入的IP采用GeoIP转换成地理位置信息，配置详情及处理结果如下：

- 输入

```
"source_ip" : "**.**.**.**"
```

下载GeoIP数据库下载GeoIP数据库到安装Logtail的主机，例如可使用*MaxMind GeoLite2*中的*City*数据库

说明：

请检查数据库格式为mmdb类型。

- 配置详情

```
{
   "type": "processor_geoip",
    "detail": {
         "SourceKey": "ip",
         "NoKeyError": true,
         "NoMatchError": true,
```

```
        "KeepSource": true,
        "DBPath" : "/user/local/data/GeoLite2-City_20180102/
GeoLite2-City.mmdb"
    }
}
```

- 配置后结果

```
"source_ip_city_" : "**.**.**.**"
"source_ip_province_" : "浙江省"
"source_ip_city_" : "杭州"
"source_ip_province_code_" : "ZJ"
"source_ip_country_code_" : "CN"
"source_ip_longitude_" : "120.********"
"source_ip_latitude_" : "30.********"
```

正则过滤

该方式通过对字段进行正则表达式匹配过滤日志，可组合使用`Include`和`Exclude`两种方式。

参数说明

正则过滤方式的类型（type）为`processor_filter_regex`。

参数	类型	必选或可选	参数说明
Include	key:string value:string 的map	可选	key为日志字段，value为该字段匹配的正则表达式，若指定字段符合该表达式，则该条日志被收集。
Exclude	key:string value:string 的map	可选	key为日志字段，value为该字段匹配的正则表达式，若指定字段符合该表达式，则该条日志不被收集。

说明：

一条日志只有完全被`Include`中的参数匹配，且不被`Exclude`中的任一参数匹配时才会被采集，否则直接丢弃。

示例

对输入日志采正则过滤方式处理，配置详情及处理结果如下：

- 输入

- 日志1

```
"ip" : "10.**.**.**"
"method" : "POST"
...
"browser" : "aliyun-sdk-java"
```

- 日志2

```
"ip" : "10.**.**.**"
"method" : "POST"
...
"browser" : "chrome"
```

- 日志3

```
"ip" : "192.168.*.*"
"method" : "POST"
...
"browser" : "ali-sls-ilogtail"
```

- 配置详情

```
{
   "type" : "processor_filter_regex",
    "detail" : {
         "Include" : {
            "ip" : "10\\..*",
            "method" : "POST"
         },
         "Exclude" : {
            "browser" : "aliyun.*"
         }
    }
}
```

- 处理后结果

日志	是否匹配	原因
日志1	不匹配	browser匹配上了Exclude。
日志2	匹配	-
日志3	不匹配	Include中”ip”字段不匹配，不以”10”开头。

组合配置

各个处理配置可以组合搭配使用。您可以参考下述配置对日志先进行分隔符切分后，再对切分后的`detail`进行标定提取。

- 输入

```
"content" :
"ACCESS|QAS|11.**.**.**|1508729889935|52460dbed4d540b88a973cf5452b14
47|1238|appKey=ba,env=pub,requestTime=1508729889913,latency=22ms,
request={appKey:ba,optional:{\\domains\\:\\daily\\,\\version\\:\\v2\
\},rawQuery:{\\query\\:\\去乐山的路线\\,\\domain\\:\\导航\\,\\intent\\:
\\navigate\\,\\slots\\:\\to_geo:level3=乐山\\,\\location\\:\\北京\\},
requestId:52460dbed4d540b88a973cf5452b1447},
response={answers:[],status:SUCCESS}|"
```

- 配置详情

```
"processors" : [
      {
          "type" : "processor_split_char",
          "detail" : {"SourceKey" : "content",
              "SplitSep" : "|",
              "SplitKeys" : ["method", "type", "ip", "time", "req_id
", "size", "detail"]
          }
      },
      {
          "type" : "processor_anchor",
          "detail" : "SourceKey" : "detail",
              "Anchors" : [
                  {
                          "Start" : "appKey=",
                      "Stop" : ",env=",
                      "FieldName" : "appKey",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",env",
                      "Stop" : ",requestTime=",
                      "FieldName" : "env",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",requestTime=",
                      "Stop" : ",latency",
                      "FieldName" : "requestTime",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",latency=",
                      "Stop" : ",request=",
                      "FieldName" : "latency",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",request=",
                      "Stop" : ",response=",
                      "FieldName" : "request",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",response=",
                      "Stop" : "",
                      "FieldName" : "response",
```

```
                    "FieldType" : "json"
                }
            ]
        }
    }
]
```

- 处理后结果

```
"method" : "ACCESS"
"type" : "QAS"
"ip" : "**.**.**.**"
"time" : "1508729889935"
"req_id" : "52460dbed4d540b88a973cf5452b1447"
"size" : "1238"
"appKey" : "ba"
"env" : "pub"
"requestTime" : "1508729889913"
"latency" : "22ms"
"request" : "{appKey:nui-banma,optional:{\\domains\\:\\daily-faq\\,
\\version\\:\\v2\\},rawQuery:{\\query\\:\\\345\216\273\344\271\220\
345\261\261\347\232\204\350\267\257\347\272\277\\,\\domain\\:\\\345
\257\274\350\210\252\\,\\intent\\:\\navigate\\,\\slots\\:\\to_geo:
level3=\344\271\220\345\261\261\\,\\location\\:\\\345\214\227\344\
272\254\\},requestId:52460dbed4d540b88a973cf5452b1447}"
"response_answers" : "[]"
"response_status" : "SUCCESS"
```

2.3.13 容器-文本日志

Logtail支持将采集容器内产生的文本日志，并附加容器的相关元数据信息一起上传到日志服务。

功能特点

相对于基础的日志文件采集，Docker文件采集还具备以下功能特点：

- 只需配置容器内的日志路径，无需关心该路径到宿主机的映射
- 支持label指定采集的容器
- 支持label排除特定容器
- 支持environment指定采集的容器
- 支持environment指定排除的容器
- 支持多行日志（例如java stack日志等）
- 支持容器数据自动打标
- 支持Kubernetes容器自动打标

限制说明

- **采集停止策略**。当container被停止后，Logtail监听到容器die的事件后会停止采集该容器日志的采集（延迟1~3秒），若此时采集出现延迟，则可能丢失停止前的部分日志。

- **Docker存储驱动限制**。目前只支持overlay、overlay2，其他存储驱动需将日志所在目录mount到本地。
- **Logtail运行方式**。必须以容器方式运行Logtail，且遵循Logtail部署方式进行部署。
- **Label**。 此处的label为docker inspect中的label信息，并不是Kubernetes配置中的label。
- **Environment**。此处的environment为容器启动中配置的environment信息。

配置流程

1. 部署并配置Logtail容器。
2. 设置服务端采集配置。

步骤1 Logtail部署和配置

- **Kubernetes**

  Kubernetes日志采集参见Logtail*Kubernetes*日志采集部署方案 。

- 其他容器管理方式

  Swarm、Mesos等其他容器管理方式，请参考*Docker*日志采集通用部署方案。

步骤2 设置数据源

1. 在**Logstore**列表单击数据接入向导图标，进入配置流程。
2. 选择数据类型。

   单击自建软件中的**Docker**文件，并单击下一步。
3. 设置数据源。

配置项	是否必选	说明
是否为Docker文件	必须勾选	确认采集的目标文件是否为Docker文件。
Label白名单	可选	每项中LabelKey必填，若LabelValue不为空，则只采集容器label中包含LabelKey=LabelValue的容器；若LabelValue为空，则采集所有label中包含LabelKey的容器。 说明： 1. 多个键值对间为或关系，即只要容器的label满足任一键值对即可被采集。 2. 此处的label为docker inspect中的label信息。

配置项	是否必选	说明
Label黑名单	可选	每项中LabelKey必填，若LabelValue不为空，则只排除容器label中包含LabelKey=LabelValue的容器；若LabelValue为空，则排除所有label中包含LabelKey的容器。 说明： 1. 多个键值对间为或关系，即只要容器的label满足任一键值对即可被排除。 2. 此处的label为docker inspect中的label信息。
环境变量白名单	可选	每项中EnvKey必填，若EnvValue不为空，则只采集容器环境变量中包含EnvKey=EnvValue的容器；若EnvValue为空，则采集所有环境变量中包含EnvKey的容器。 说明： • 多个键值对间为或关系，即只要容器的环境变量满足任一键值对即可被采集。 • 此处的environment为容器启动中配置的environment信息。
环境变量黑名单	可选	每项中EnvKey必填，若EnvValue不为空，则只排除容器环境变量中包含EnvKey=EnvValue的容器；若EnvValue为空，则排除所有环境变量中包含EnvKey的容器。 说明： • 多个键值对间为或关系，即只要容器的环境变量满足任一键值对即可被排除。 • 此处的environment为容器启动中配置的environment信息。
其他配置项	-	其他采集配置以及参数说明见文本日志。

说明：

- 本文中label白名单、黑名单与Kubernetes中定义的label不是同一概念，本文档中的label为Docker inspect中的label信息。

- Kubernetes中的namespace和容器名会映射到docker的label中，分别为`io.kubernetes.pod.namespace`和`io.kubernetes.container.name`。例如您创建的pod所属namespace为backend-prod，容器名为worker-server，则可以配置2个label白名单以指定只采集该容器的日志，分别为 `io.kubernetes.pod.namespace : backend-prod`和`io.kubernetes.container.name : worker-server`。
- Kubernetes中除`io.kubernetes.pod.namespace`和`io.kubernetes.container.name`外不建议使用其他label。其他情况请使用环境变量白名单或黑名单。

4. 应用到机器组。

   进入应用到机器组页面。请在此处勾选需要采集的Logtail机器组，并单击右下角的**应用到机器组**。如您之前没有创建过机器组，单击**+创建机器组**以创建一个新的机器组。

5. 完成容器文本日志接入流程。

   如您需要开启检索分析、投递配置等功能，请按照页面提示继续配置。

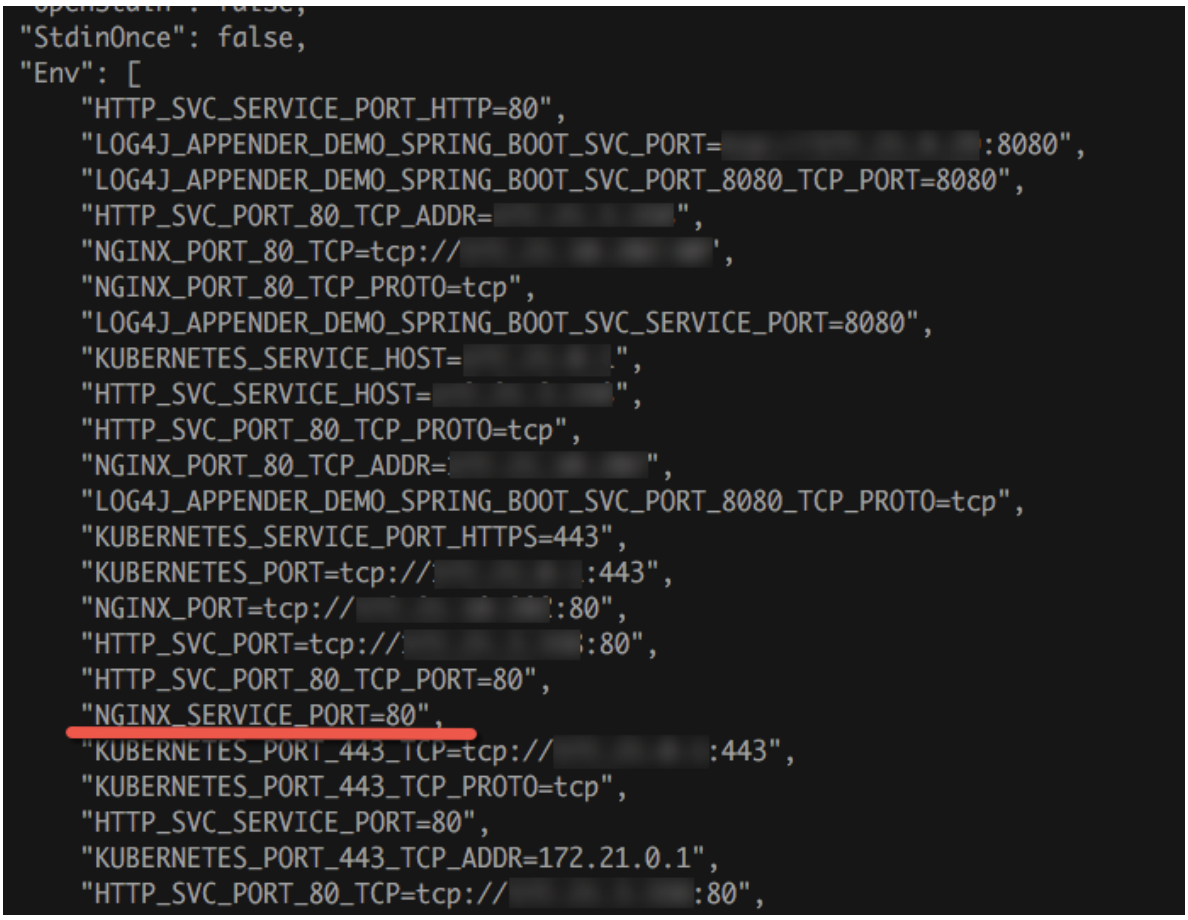
配置示例

- **environment 配置方式**

  采集environment为`NGINX_PORT_80_TCP_PORT=80`且environment不为`POD_NAMESPACE=kube-system`的容器日志，日志文件路径为`/var/log/nginx/access.log`，日志解析方式为极简类型。

  > **说明：**
  > 此处的environment为容器启动中配置的environment信息。

图 **2-19: environment** 配置方式示例



本示例中数据源配置如下。其他采集配置以及参数说明见文本日志。

图 2-20: 数据源配置示例

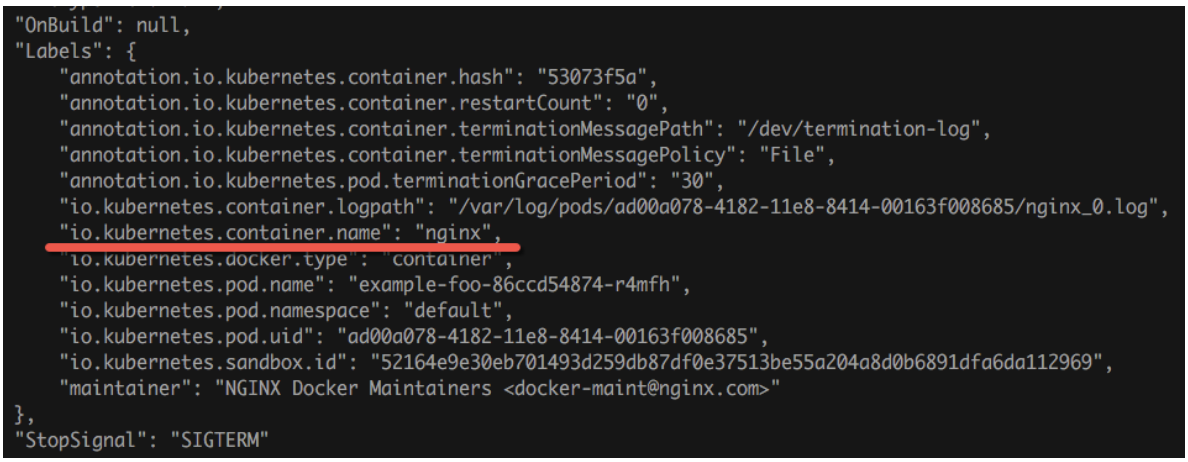
- **label 配置方式**

  采集label为`io.kubernetes.container.name=nginx`的容器日志，日志文件路径为`/var/log/nginx/access.log`，日志解析方式为极简类型。

  说明：

  此处的label为docker inspect中的label信息，并不是Kubernetes配置中的label。

图 2-21: label方式示例



本示例中数据源配置如下。其他采集配置以及参数说明见文本日志。

图 2-22: 数据源配置

默认字段

普通Docker

每条日志默认上传以下字段：

字段名	说明
_image_name_	镜像名
_container_name_	容器名

字段名	说明
_container_ip_	容器IP地址

Kubernetes

若该集群为Kubernetes，则默认每条日志上传以下字段：

字段名	说明
_image_name_	镜像名
_container_name_	容器名
_pod_name_	pod名
namespace	pod所在命名空间
_pod_uid_	pod的唯一标识
_container_ip_	pod的IP地址

2.3.14 容器-标准输出

Logtail支持将容器的标准输出流作为输入源，并附加容器的相关元数据信息一起上传到日志服务。

功能特点

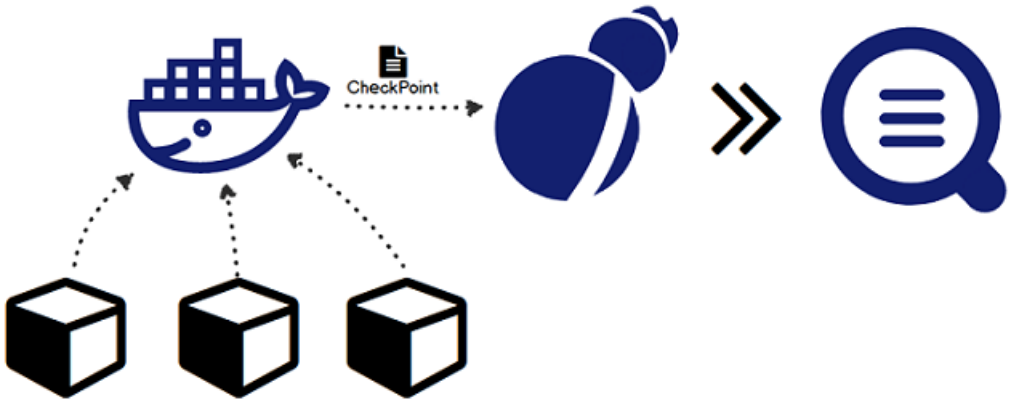
- 支持采集stdout、stderr
- 支持label指定采集的容器
- 支持label排除特定容器
- 支持environment指定采集的容器
- 支持environment指定排除的容器
- 支持多行日志（例如java stack日志等）
- 支持container数据自动打标
- 支持Kubernetes容器自动打标

实现原理

如下图所示，Logtail会与Docker的Domain Socket进行通信，查询该Docker上运行的所有container，并根据container配置的label和environment信息定位需要被采集的container。随后Logtail会通过docker的log命令获取指定container的日志。

Logtail在采集容器的标准输出时，会定期将采集的点位信息保存到checkpoint文件中，若Logtail停止后再次启动，会从上一次保存的点位开始采集日志。

图 **2-23:** 实现原理



使用限制

- 此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。
- Logtail默认通过`/var/run/docker.sock`访问docker engine，请确保该Domain Socket存在且具备访问权限。
- **多行日志限制**。为保证多行组成的一条日志不因为输出延迟而被分割成多条，多行日志情况下，采集的最后一条日志默认都会缓存一段时间。默认缓存时间为3秒，可通过`BeginLineTimeoutMs`设置，但此值不能低于1000，否则容易出现误判。
- **采集停止策略**。当container被停止后，Logtail监听到容器`die`的事件后会停止采集该container的标准输出，若此时采集出现延迟，则可能丢失停止前的部分输出。
- **Docker日志驱动类型限制**。目前标准输出采集仅支持JSON类型的日志驱动。
- **上下文限制**。默认一个采集配置在同一上下文中，若需要每种类型的container一个上下文，请单独配置。
- **数据处理**。采集到的数据默认字段为`content`，支持通用的处理配置。请参考自定义插件配置一种或多种采集方式。
- **Label**。 此处的label为docker inspect中的label信息，并不是Kubernetes配置中的label。
- **Environment**。此处的environment为容器启动中配置的environment信息。

配置流程

1. 部署并配置Logtail容器。
2. 设置服务端采集配置。

步骤1 Logtail部署和配置

- **Kubernetes**

  Kubernetes日志采集参见Logtail *Kubernetes*日志采集部署方案。

- 其他容器管理方式

  Swarm、Mesos等其他容器管理方式，请参考*Docker*日志采集通用部署方案。

步骤2 设置数据源

1. 在**Logstore**列表单击数据接入向导图标，进入配置流程。
2. 选择数据类型。

   单击自建软件中的**Docker**标准输出，并单击下一步。
3. 设置数据源。

   在输入源配置页面，填写您的采集配置。示例如下，配置项说明请查看本文档中配置项说明部分。

```
{
 "inputs": [
     {
         "type": "service_docker_stdout",
         "detail": {
             "Stdout": true,
             "Stderr": true,
             "IncludeLabel": {
                 "io.kubernetes.container.name": "nginx"
             },
             "ExcludeLabel": {
                 "io.kubernetes.container.name": "nginx-ingress-
controller"
             },
             "IncludeEnv": {
                 "NGINX_SERVICE_PORT": "80"
             },
             "ExcludeEnv": {
                 "POD_NAMESPACE": "kube-system"
             }
         }
     }
 ]
}
```

4. 应用到机器组。

   进入应用到机器组页面。请在此处勾选需要采集的Logtail机器组。如您之前没有创建过机器组，单击**+**创建机器组以创建一个新的机器组。

配置项说明

该输入源类型为: `service_docker_stdout`。

> 说明：
> Logtail支持对采集的数据进行处理加工后上传，处理方式参见插件-处理采集数据。

配置项	类型	是否必选	说明
IncludeLabel	map类型，其中key为string，value为string	必选	默认为空，为空时代表采集所有Container数据；当key非空，value为空时，代表包含label中所有包含此key的container。 说明： 1. 多个键值对间为或关系，即只要容器的label满足任一键值对即可被采集。 2. 此处的label为docker inspect中的label信息。
ExcludeLabel	map类型，其中key为string，value为string	可选	默认为空，为空时不排除任何Container；当key非空，value为空时，代表排除label中所有包含此key的container。 说明： 1. 多个键值对间为或关系，即只要容器的label满足任一键值对即被排除。 2. 此处的label为docker inspect中的label信息。
IncludeEnv	map类型，其中key为string，value为string	可选	默认为空，为空时代表采集所有Container数据；当key非空，value为空时，代表包含environment中所有包含此key的container。 说明： 1. 多个键值对间为或关系，即只要容器的environment满足任一键值对即可被采集。 2. 此处的environment为容器启动中配置的environment信息。
ExcludeEnv	map类型，其中key为string，value为string	可选	默认为空，为空时不排除任何Container；当key非空，value为空时，代表排除Environment中所有包含此key的container。

配置项	类型	是否必选	说明
			说明： 1. 多个键值对间为或关系，即只要容器的environment满足任一键值对即被排除。 2. 此处的environment为容器启动中配置的environment信息。
Stdout	bool	可选	默认为true，为false时不采集stdout数据。
Stderr	bool	可选	默认为true，为false时不采集stderr数据。
BeginLineRegex	string	可选	默认为空，非空时为行首匹配的正则表达式。若该表达式匹配某行，则将该行作为新的一条日志；否则将此行数据连接到上一条日志。
BeginLineTimeoutMs	int	可选	行首匹配超时时间，默认为3000，单位为毫秒。若3秒内没有新日志出现，则将最后一条日志输出。
BeginLineCheckLength	int	可选	行首匹配的长度，默认为10×1024，单位为字节。若行首规则在前N个字节即可体现，可设置此参数，以此提升行首匹配效率。
MaxLogSize	int	可选	日志最大长度，默认为512×1024，单位为字节。若日志超过该项配置，则不继续查找行首，直接上传。

说明：

1. 本文档中IncludeLabel、ExcludeLabel和Kubernetes中定义的label不是同一概念，本文档中的label为docker inspect中的label信息。
2. Kubernetes中的namespace和容器名会映射到docker的label中，分别为`io.kubernetes.pod.namespace`和`io.kubernetes.container.name`。例如您创建的pod所属namespace为backend-prod，容器名为 worker-server，则IncludeLabel中可以配置2个键值对来指定只采集该容器的日志，分别为 `io.kubernetes.pod.namespace : backend-prod`和`io.kubernetes.container.name : worker-server`。
3. Kubernetes不建议使用除`io.kubernetes.pod.namespace`和`io.kubernetes.container.name`之外的其他label。其他情况请使用IncludeEnv/ExcludeEnv。

默认字段

普通Docker

每条日志默认上传以下字段：

字段名	说明
time	数据时间，样例为2018-02-02T02:18:41.979147844Z
source	输入源类型，stdout 或 stderr
_image_name_	镜像名
_container_name_	容器名
_container_ip_	容器IP

Kubernetes

若该集群为Kubernetes，则默认每条日志上传以下字段：

字段名	说明
time	数据时间，样例为2018-02-02T02:18:41.979147844Z
source	输入源类型，stdout 或 stderr
_image_name_	镜像名
_container_name_	容器名
_pod_name_	pod名
namespace	pod所在命名空间
_pod_uid_	pod的唯一标识
_container_id_	pod的IP地址

配置示例

常规配置

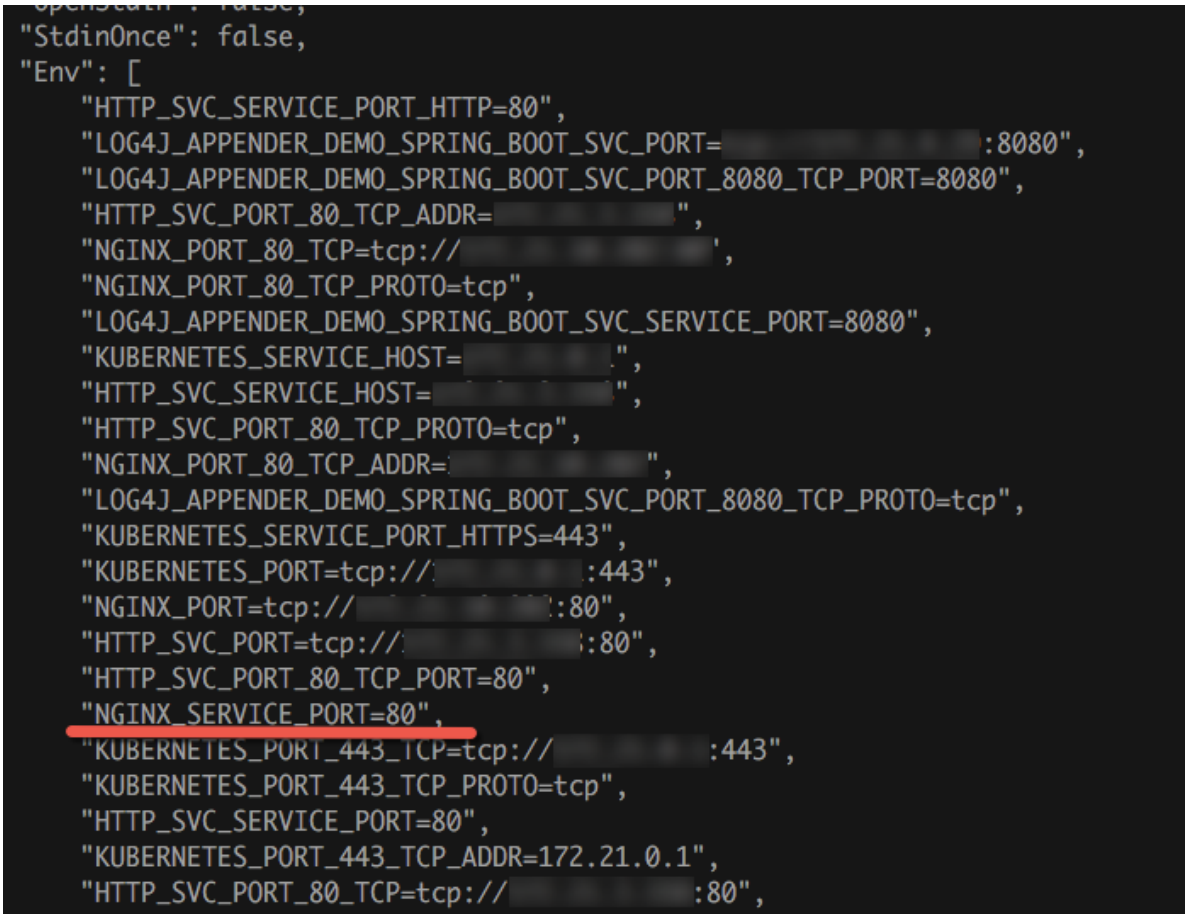
- **environment 配置方式**

  采集environment为NGINX_PORT_80_TCP_PORT=80且environment不为POD_NAMESPACE=kube-system的stdout以及stderr日志：

说明：

此处的environment为容器启动中配置的environment信息。

图 **2-24: environment** 配置方式示例



采集配置：

```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "IncludeEnv": {
                    "NGINX_PORT_80_TCP_PORT": "80"
                },
                "ExcludeEnv": {
                    "POD_NAMESPACE": "kube-system"
                }
            }
        }
    ]
```

```
}
```

- **label 配置方式**

  采集label为`io.kubernetes.container.name=nginx`且label不为`type=pre`的stdout以及stderr日志：

  > 说明：
  > 此处的label为docker inspect中的label信息，并不是Kubernetes配置中的label。

  图 **2-25: label配置方式示例**


```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "IncludeLabel": {
                    "io.kubernetes.container.name": "nginx"
                },
                "ExcludeLabel": {
                    "type": "pre"
                }
            }
        }
    ]
}
```

多行日志采集配置

多行日志采集对于Java异常堆栈输出的采集尤为重要，这里介绍一种标准的Java标准输出日志的采集配置。

- 日志示例

```
2018-02-03 14:18:41.968  INFO [spring-cloud-monitor] [nio-8080-exec-
4] c.g.s.web.controller.DemoController : service start
2018-02-03 14:18:41.969 ERROR [spring-cloud-monitor] [nio-8080-
exec-4] c.g.s.web.controller.DemoController : java.lang.NullPointe
rException
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(
ApplicationFilterChain.java:193)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(
ApplicationFilterChain.java:166)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWr
apperValve.java:199)
at org.apache.catalina.core.StandardContextValve.invoke(StandardCo
ntextValve.java:96)
...
2018-02-03 14:18:41.968  INFO [spring-cloud-monitor] [nio-8080-exec-
4] c.g.s.web.controller.DemoController : service start done
```

- 采集配置

  采集label为`app=monitor`输入的日志，行首为日期类型（为提高匹配效率，这里只判断行首的10个字节）。

```
{
"inputs": [
  {
    "detail": {
      "BeginLineCheckLength": 10,
      "BeginLineRegex": "\\d+-\\d+-\\d+.*",
      "IncludeLabel": {
        "app": "monitor"
      }
    },
    "type": "service_docker_stdout"
  }
]
}
```

采集数据处理

Logtail对于采集到的Docker标准输出，支持通用数据处理方式。建议使用基于上一节的多行日志格式，使用正则表达式将日志解析成time、module、thread、class、info。

- 采集配置：

  采集label为`app=monitor`输入的日志，行首为日期类型（为提高匹配效率，这里只判断行首的10个字节）。

```
{
"inputs": [
  {
    "detail": {
      "BeginLineCheckLength": 10,
      "BeginLineRegex": "\\d+-\\d+-\\d+.*",
```

```
      "IncludeLabel": {
        "app": "monitor"
      }
    },
    "type": "service_docker_stdout"
  }
],
"processors": [
    {
        "type": "processor_regex",
        "detail": {
            "SourceKey": "content",
            "Regex": "(\\d+-\\d+-\\d+ \\d+:\\d+:\\d+\\.\\d+)\\s+(\\w
+)\\s+\\[([^]]+)]\\s+\\[([^]]+)]\\s+:\\s+([\\s\\S]*)",
            "Keys": [
                "time",
                "module",
                "thread",
                "class",
                "info"
            ],
            "NoKeyError": true,
            "NoMatchError": true,
            "KeepSource": false
        }
    }
]
}
```

- 样例输出

  对于日志`2018-02-03 14:18:41.968 INFO [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoController : service start done`处理后的输出为：

```
__tag__:__hostname__:logtail-dfgef
_container_name_:monitor
_image_name_:registry.cn-hangzhou.aliyuncs.xxxxxxxxxxxxxx
_namespace_:default
_pod_name_:monitor-6f54bd5d74-rtzc7
_pod_uid_:7f012b72-04c7-11e8-84aa-00163f00c369
_source_:stdout
_time_:2018-02-02T14:18:41.979147844Z
time:2018-02-02 02:18:41.968
level:INFO
module:spring-cloud-monitor
thread:nio-8080-exec-4
class:c.g.s.web.controller.DemoController
message:service start done
```

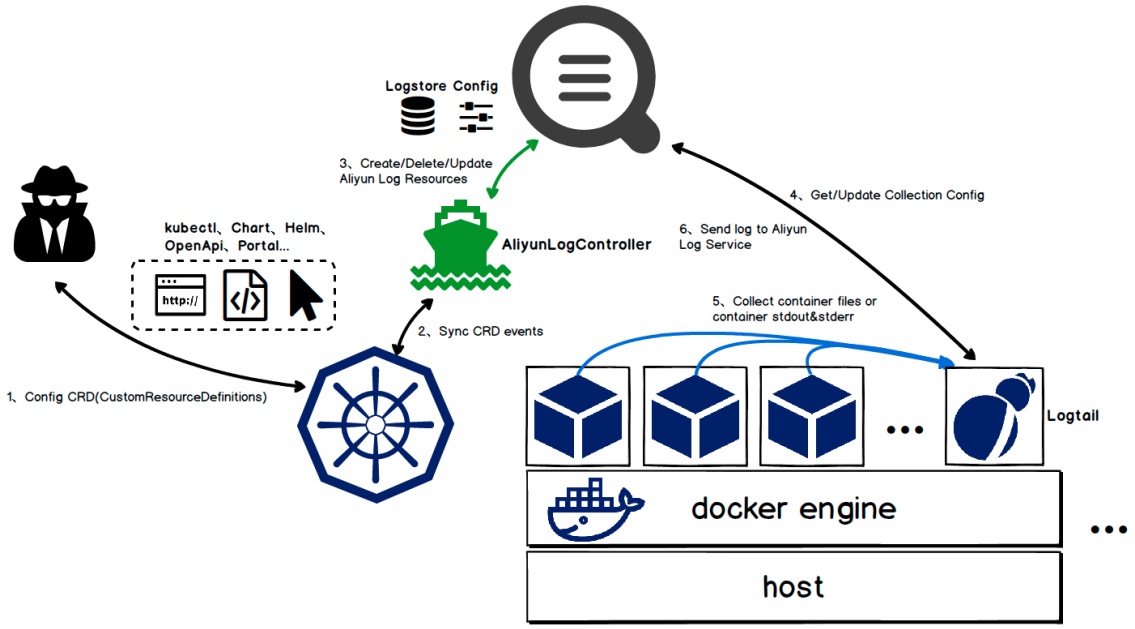
2.3.15 Kubernetes-CRD配置日志采集

配置日志采集默认通过控制台方式配置。针对Kubernetes微服务开发模式，除控制台方式外，日志服务还提供CRD的配置方式日志，您可以直接使用kubectl对配置进行管理。

推荐您使用CRD方式进行采集配置管理，该方式与Kubernetes部署、发布流程的集成更加完善。

实现原理

图 2-26: 实现原理



执行安装命令时会自动安装`alibaba-log-controller`的Helm包。Helm包中主要执行了以下操作：

1. 创建aliyunlogconfigs CRD（CustomResourceDefinition）。
2. 部署alibaba-log-controller的Deployment。
3. 部署Logtail的DaemonSet。

具体配置的内部工作流程如下：

1. 用户使用`kubectl`或其他工具应用aliyunlogconfigs CRD配置。
2. alibaba-log-controller监听到配置更新。
3. alibaba-log-controller根据CRD内容以及服务端状态，自动向日志服务提交logstore创建、配置创建以及应用机器组的请求。
4. 以DaemonSet模式运行的Logtail会定期请求配置服务器，获取新的或已更新的配置并进行热加载。
5. Logtail根据配置信息采集各个容器(POD)上的标准输出或文件。
6. 最终Logtail将处理、聚合好的数据发送到日志服务。

配置方式

说明：

如果您之前使用的DaemonSet方式部署的日志服务Logtail，将无法使用CRD的方式进行配置管理。升级方式参见本文档中**DaemonSet部署方式迁移步骤**。

您只需要定义AliyunLogConfig的CRD即可实现配置的创建，删除对应的CRD资源即可删除对应配置。CRD配置格式如下：

```
apiVersion: log.alibabacloud.com/v1alpha1       ## 默认值，无需改变
kind: AliyunLogConfig                           ## 默认值，无需改变
metadata:
  name: simple-stdout-example                   ## 资源名，必须在集群内唯一
spec:
  logstore: k8s-stdout                          ## Logstore名，不存在时自动创建
  shardCount: 2                                 ## [可选] logstore分区数，默认为2，支持1-10
  lifeCycle: 90                                 ## [可选] 该logstore存储时间，默认为90，支持1-7300，7300天为永久存储
  logtailConfig:                                ## 详细配置
    inputType: plugin                           ## 采集的输入类型，一般为file或plugin
    configName: simple-stdout-example           ## 采集配置名，需要和资源名(metadata.name)一致
    inputDetail:                                ## 详细配置信息，具体请参考示例
      ...
```

配置完成并应用配置后，会自动创建alibaba-log-controller。

查看配置

您可以通过Kubernetes CRD或控制台查看配置。

控制台查看配置参见管理采集配置。

说明：

使用CRD管理方式，若您在控制台更改配置，下一次执行CRD更新配置时，会覆盖控制台的更改内容。

- 使用`kubectl get aliyunlogconfigs`查看当前所有配置。

```
[root@iZbp1dsbiaZ ~]# kubectl get aliyunlogconfigs
NAME                   AGE
regex-file-example     10s
regex-stdout-example   4h
```

```
simple-file-example    5s
```

- 使用`kubectl get aliyunlogconfigs ${config_name} -o yaml`查看详细配置和状态。

  配置中`status`字段显示配置执行的结果，若配置应用成功，`status`字段中的`statusCode`为200。若`statusCode`非200，则说明配置应用失败。

```
[root@iZbp1dsbiaZ ~]# kubectl get aliyunlogconfigs simple-file-
example -o yaml
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
annotations:
  kubectl.kubernetes.io/last-applied-configuration: |
    {"apiVersion":"log.alibabacloud.com/v1alpha1","kind":"AliyunLogC
onfig","metadata":{"annotations":{},"name":"simple-file-example","
namespace":"default"},"spec":{"logstore":"k8s-file","logtailConfig
":{"configName":"simple-file-example","inputDetail":{"dockerFile
":true,"dockerIncludeEnv":{"ALIYUN_LOGTAIL_USER_DEFINED_ID":""},"
filePattern":"simple.LOG","logPath":"/usr/local/ilogtail","logType
":"common_reg_log"},"inputType":"file"}}}
clusterName: ""
creationTimestamp: 2018-05-17T08:44:46Z
generation: 0
name: simple-file-example
namespace: default
resourceVersion: "21790443"
selfLink: /apis/log.alibabacloud.com/v1alpha1/namespaces/default/
aliyunlogconfigs/simple-file-example
uid: 8d3a09c4-59ae-11e8-851d-00163f008685
spec:
lifeCycle: null
logstore: k8s-file
logtailConfig:
  configName: simple-file-example
  inputDetail:
    dockerFile: true
    dockerIncludeEnv:
      ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
    filePattern: simple.LOG
    logPath: /usr/local/ilogtail
    logType: common_reg_log
  inputType: file
machineGroups: null
project: ""
shardCount: null
status:
status: OK
```

```
statusCode: 200
```

配置示例

容器标准输出

容器标准输出中，需要将`inputType`设置为`plugin`，并将具体信息填写到`inputDetail`下的`plugin`字段，详细配置字段及其含义请参考容器-标准输出。

- 极简采集方式

  采集除了环境变量中配置`COLLECT_STDOUT_FLAG=false`之外所有容器的标准输出(stdout和stderr)。

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: simple-stdout-example
spec:
  # logstore name to upload log
  logstore: k8s-stdout
  # logtail config detail
  logtailConfig:
    # docker stdout's input type is 'plugin'
    inputType: plugin
    # logtail config name, should be same with [metadata.name]
    configName: simple-stdout-example
    inputDetail:
      plugin:
        inputs:
          -
            # input type
            type: service_docker_stdout
            detail:
              # collect stdout and stderr
              Stdout: true
              Stderr: true
              # collect all container's stdout except containers
with "COLLECT_STDOUT_FLAG:false" in docker env config
              ExcludeEnv:
                COLLECT_STDOUT_FLAG: "false"
```

- 自定义处理采集方式

  采集grafana的access log，并将access log解析成结构化数据。

  grafana的容器配置中包含环境变量为：`GF_INSTALL_PLUGINS=grafana-piechart-....`，通过配置`IncludeEnv`为`GF_INSTALL_PLUGINS: ''`指定Logtail只采集该容器的标准输出。

图 2-27: 自定义处理采集方式

grafana的access log格式如下：

```
t=2018-03-09T07:14:03+0000 lvl=info msg="Request Completed" logger
=context userId=0 orgId=0 uname= method=GET path=/ status=302
remote_addr=172.16.64.154 time_ms=0 size=29 referer=
```

使用正则表达式解析access log，具体配置如下：

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: regex-stdout-example
spec:
  # logstore name to upload log
  logstore: k8s-stdout-regex
  # logtail config detail
  logtailConfig:
    # docker stdout's input type is 'plugin'
    inputType: plugin
    # logtail config name, should be same with [metadata.name]
    configName: regex-stdout-example
    inputDetail:
      plugin:
        inputs:
          -
            # input type
            type: service_docker_stdout
            detail:
              # 只采集stdout，不采集stderr
              Stdout: true
              Stderr: false
              # 只采集容器环境变量中配置key为"GF_INSTALL_PLUGINS"的stdout

              IncludeEnv:
                GF_INSTALL_PLUGINS: ''
        processors:
          -
            # 使用正则表达式处理
            type: processor_regex
            detail:
              # docker 采集的数据默认key为"content"
```

```
                SourceKey: content
                # 正则表达式提取
                Regex: 't=(\d+-\d+-\w+:\d+:\d+\+\d+) lvl=(\w+) msg
="([^"]+)" logger=(\w+) userId=(\w+) orgId=(\w+) uname=(\S*) method
=(\w+) path=(\S+) status=(\d+) remote_addr=(\S+) time_ms=(\d+) size
=(\d+) referer=(\S*).*'
                # 提取出的key
                Keys: ['time', 'level', 'message', 'logger', 'userId
', 'orgId', 'uname', 'method', 'path', 'status', 'remote_addr', '
time_ms', 'size', 'referer']
                # 保留原始字段
                KeepSource: true
                NoKeyError: true
                NoMatchError: true
```

配置应用后，采集到日志服务的数据如下：

图 **2-28:** 采集到的日志数据

容器文件

- 极简文件

  采集环境变量配置中含有key为`ALIYUN_LOGTAIL_USER_DEFINED_ID`的容器内日志文件，文件所处的路径为`/data/logs/app_1`，文件名为`simple.LOG`。

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: simple-file-example
spec:
  # logstore name to upload log
  logstore: k8s-file
  # logtail config detail
  logtailConfig:
    # log file's input type is 'file'
    inputType: file
    # logtail config name, should be same with [metadata.name]
    configName: simple-file-example
    inputDetail:
      # 极简模式日志，logType设置为"common_reg_log"
      logType: common_reg_log
      # 日志文件夹
```

```
      logPath: /data/logs/app_1
      # 文件名, 支持通配符，例如log_*.log
      filePattern: simple.LOG
      # 采集容器内的文件，dockerFile flag设置为true
      dockerFile: true
      # only collect container with "ALIYUN_LOGTAIL_USER_DEFINED_ID
" in docker env config
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
```

- 完整正则模式文件

  对于某Java程序日志样例为：

```
[2018-05-11T20:10:16,000] [INFO] [SessionTracker] [SessionTra
ckerImpl.java:148] Expiring sessions
java.sql.SQLException: Incorrect string value: '\xF0\x9F\x8E\x8F
",...' for column 'data' at row 1
at org.springframework.jdbc.support.AbstractFallbackSQLExceptionTr
anslator.translate(AbstractFallbackSQLExceptionTranslator.java:84)
at org.springframework.jdbc.support.AbstractFallbackSQLException
```

  日志中由于包含错误堆栈信息，可能一条日志会被分解成多行，因此需要设置行首正则表达式；为了提取出各个字段，这里我们使用正则表达式进行提取，具体配置如下：

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: regex-file-example
spec:
  # logstore name to upload log
  logstore: k8s-file
  logtailConfig:
    # log file's input type is 'file'
    inputType: file
    # logtail config name, should be same with [metadata.name]
    configName: regex-file-example
    inputDetail:
      # 对于正则类型的日志，将logType设置为common_reg_log
      logType: common_reg_log
      # 日志文件夹
      logPath: /app/logs
      # 文件名, 支持通配符，例如log_*.log
      filePattern: error.LOG
      # 行首正则表达式
      logBeginRegex: '\[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*'
      # 解析正则
      regex: '\[([^]]+)]\s\[(\w+)]\s\[(\w+)]\s\[([^:]+):(\d+)]\s
(.*)'
      # 提取出的key列表
      key : ["time", "level", "method", "file", "line", "message"]
      # 正则模式日志，时间解析默认从日志中的`time`提取，如果无需提取时间，可不
设置该字段
      timeFormat: '%Y-%m-%dT%H:%M:%S'
      # 采集容器内的文件，dockerFile flag设置为true
      dockerFile: true
```

```
      # only collect container with "ALIYUN_LOGTAIL_USER_DEFINED_ID
" in docker env config
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
```

配置应用后，采集到日志服务的数据如下：

图 **2-29:** 采集到的日志数据

- 分隔符模式文件

  Logtail同时支持分隔符模式的日志解析，示例如下：

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: delimiter-file-example
spec:
  # logstore name to upload log
  logstore: k8s-file
  logtailConfig:
    # log file's input type is 'file'
    inputType: file
    configName: delimiter-file-example
    # logtail config name, should be same with [metadata.name]
    inputDetail:
      # 对于分隔符类型的日志，logType设置为delimiter_log
      logType: delimiter_log
      # 日志文件夹
      logPath: /usr/local/ilogtail
      # 文件名，支持通配符，例如log_*.log
      filePattern: delimiter_log.LOG
      # 使用多字符分隔符
      separator: '|&|'
      # 提取的key列表
      key : ['time', 'level', 'method', 'file', 'line', 'message']
      # 用作解析时间的key，如无需求可不填
      timeKey: 'time'
      # 时间解析方式，如无需求可不填
      timeFormat: '%Y-%m-%dT%H:%M:%S'
      # 采集容器内的文件，dockerFile flag设置为true
      dockerFile: true
      # only collect container with "ALIYUN_LOGTAIL_USER_DEFINED_ID
" in docker env config
      dockerIncludeEnv:
```

```
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ''
```

- **JSON模式文件**

  若文件中每行数据为一个JSON object，可以使用JSON方式进行解析，示例如下：

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: json-file-example
spec:
  # logstore name to upload log
  logstore: k8s-file
  logtailConfig:
    # log file's input type is 'file'
    inputType: file
    # logtail config name, should be same with [metadata.name]
    configName: json-file-example
    inputDetail:
      # 对于分隔符类型的日志，logType设置为json_log
      logType: json_log
      # 日志文件夹
      logPath: /usr/local/ilogtail
      # 文件名, 支持通配符，例如log_*.log
      filePattern: json_log.LOG
      # 用作解析时间的key，如无需求可不填
      timeKey: 'time'
      # 时间解析方式，如无需求可不填
      timeFormat: '%Y-%m-%dT%H:%M:%S'
      # 采集容器内的文件，dockerFile flag设置为true
      dockerFile: true
      # only collect container with "ALIYUN_LOGTAIL_USER_DEFINED_ID
" in docker env config
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
```

2.4 相关限制说明

表 2-1: 文件采集限制

分类	限制说明
文件编码	支持UTF8/GBK编码日志文件，建议使用UTF8编码以获得更好的处理性能。如果日志文件为其它编码格式则会出现乱码、数据丢失等错误。
日志文件大小	无限制。
日志文件轮转	支持，流转文件名支持配置为`.log*`或者`.log`。
日志解析阻塞时采集行为	日志解析阻塞时，Logtail会将该日志文件FD保持打开状态；若解析阻塞期间出现多次日志文件

分类	限制说明
	轮转，Logtail会尽可能保持各个轮转日志解析顺序。若未解析的日志轮转超过20个，则后续文件不被处理。更多内容请参考相关技术文章。
软链接	支持监控目录为软链接。
单条日志大小	单条日志大小限制为512KB。多行日志按行首正则表达式划分后，每条日志大小限制仍为512KB。若日志超过512KB后，会强制拆分多块进行采集。例如：日志单条1025KB，则第一次处理前512KB，第二次处理512KB，第三次处理1KB。
正则表达式	正则表达式类型支持Perl兼容正则表达式。
同一文件对应多个采集配置	不支持，建议文件采集到一个Logstore，可以配置多份订阅。若有相关需求，可通过为文件配置软连接的方式绕过该限制。
文件打开行为	Logtail会保持被采集文件处于打开状态，若该文件超过5分钟未修改，则会关闭该文件（未发生轮转情况下）。
首次日志采集行为	Logtail只采集增量的日志文件，首次发现文件修改后，若文件大小超过1M，则从最后1M处开始采集，否则从开始位置采集；若配置下发后日志文件一直无修改，则不采集该文件。
非标准文本日志	对于日志中包含'\0'的行，该条日志会被截断到第一个'\0'处。

表 2-2: Checkpoint管理

项目	能力与限制
Checkpoint超时时间	若文件超过30天未修改，则会删除该Checkpoint。
Checkpoint保存策略	定期保存（15分钟），程序退出时会自动保存。
Checkpoint保存位置	保存路径默认为`/tmp/logtail_checkpoint`，可根据 配置启动参数 调整参数。

表 2-3: 配置限制

项目	能力与限制
配置更新	用户的配置更新生效的延时约30秒。
配置动态加载	支持，且其中某一配置更新不影响其他采集。
配置数	理论无限制，建议一台服务器采集配置数不超过100。
多租户隔离	各个采集配置间隔离。详细内容请参考相关技术文章。

表 2-4: 资源、性能限制

项目	能力与限制
日志处理吞吐能力	原始日志流量默认限制为2MB/s（数据会编码压缩后上传，一般压缩率为5-10倍）。超过该日志流量则有可能丢失日志，可根据 配置启动参数 调整参数。
最大性能	单核能力：极简模式日志最大处理能力为100MB/s，正则默认最大处理能力为20MB/s（和正则复杂度有关），分隔符日志最大处理能力为40MB/s，JSON日志最大处理能力为30MB/s；开启多个处理线程性能可提高1.5-3倍左右
监控目录数	主动限制监控的目录层深，避免出现过多消耗用户资源。如果监控上限已到，则放弃监控更多目录和日志文件。限制最多3000个目录（含子目录）。
默认资源限制	默认Logtail最多会占用40%CPU、256MB内存，如日志产生速率较高，可根据 配置启动参数 调整参数。
资源超限处理策略	若3分钟内Logtail占用的相关资源超过最大限制，则Logtail会强制重启，此时数据可能会丢失或重复。

表 2-5: 错误处理限制

项目	能力与限制
网络错误处理	在出现网络异常时会主动重试并自动调整重试间隔。
资源配额超限处理	若数据发送速率超出Logstore最大配额，Logtail会阻塞采集并自动重试。详细内容请参考相关技术文章。。

项目	能力与限制
超时最大尝试时间	若数据持续发送失败超过6小时，则丢弃该数据。
状态自检	支持异常情况下自动重启，例如程序异常退出及使用资源超限等。

表 2-6: 其他限制

项目	能力与限制
日志采集延迟	正常情况下从日志flush磁盘到Logtail采集改日志延迟不超过1秒（阻塞状态下除外）。
日志上传策略	Logtail会将同一文件的日志自动聚合上传，聚合条件为：日志超过2000条、日志总大小超过2M或者日志采集时间超过3秒，任一条件满足则触发上传行为。

3 索引与查询

3.1 简介

日志服务提供大规模、日志实时查询与分析能力（LogSearch/Analytics），可以通过开启索引+字段统计来支持该功能。

功能优势

- 实时：写入后可以立即被分析。
- 速度快：
  - 查询：一秒内查询（5个条件）可处理10亿级数据。
  - 分析：一秒内分析（5个维度聚合+GroupBy）可聚合亿级别数据。
- 灵活：可以改变任意查询和分析条件，实时获取结果。
- 生态：除控制台提供的报表、仪表盘、快速分析等功能外，还可以和无缝与Grafana、DataV、Jaeger等产品对接，并支持Restful API，JDBC等协议。

基本概念

在不开启查询分析（索引）时，原始数据可以根据Shard进行顺序消费（类似Kafka），在开启后除了支持顺序消费外，还可以对日志数据进行统计与查询。

开启索引

1. 登录日志服务控制台。选择目标项目，单击项目名称。
2. 选择目标日志库并单击日志索引列下的**查询**。单击右上角的**开启索引**。 如您之前已开启索引，请直接单击**设置查询分析 > 设置**。
   - 开启查询、统计后意味着数据将会在后台被索引，会产生索引的流量，以及索引对应存储的空间。
   - 如果不需要查询分析功能，可以单击**关闭索引**。
3. 进入设置菜单进行配置。

数据类型

对一条日志而言，每个Key都可以设置类型（全文索引本身也是一个特殊的Key，把整条日志作为Value），目前支持类型如下：

类别	类型	说明	查询示例
基础	*TEXT*	文本类型，可以进行关键词+模糊匹配，支持中文分词。	`uri:"login*" method:"post"`
基础	*Long*	数值类型，支持区间查询。	`status>200, status in [200, 500]`
基础	*Double*	带浮点数数值类型。	`price>28.95, t in [20.0, 37]`
组合	*JSON*	内容为JSON字段，默认为Text类型，支持嵌套模式。可以通过 a.b等路径格式给a层下b元素设置（Text、Long、Double）类型索引，设置后的字段类型以设置为主。	`level0.key>29.95 level0.key2:"action"`
组合	文本	整条日志当做文本进行查询。	`error and "login fail"`

查询分析语法

实时分析查询（Query）：由（Search、Analytics）两个部分组成，中间通过|进行分割：`$Search |$Analytics`

- 查询（Search）：查询条件，可以由关键词、模糊、数值等、区间范围和组合条件等产生。如果为空或”*”，则代表所有数据。
- 分析（Analytics）：对查询结果或全量数据进行计算和统计。

说明：

两部分均为可选，当Search部分为空时代表针对该时间段所有数据不过滤任何条件，直接对结果进行统计。当Analysis部分为空时，代表只返回查询结果，不需要做统计。

说明：

更多请细节参见： 查询语法、实时分析简介。

查询示例

以下一条日志除时间外，还包含4个键值：

序号	**Key**	类型
0	time	-

序号	Key	类型
1	class	text
2	status	long
3	latency	double
4	message	json

```
0. time:2018-01-01 12:00:00
  1. class:central-log
  2. status:200
  3. latency:68.75
  4. message:
  {
      "methodName": "getProjectInfo",
      "success": true,
      "remoteAddress": "1.1.1.1:11111",
      "usedTime": 48,
      "param": {
              "projectName": "ali-log-test-project",
              "requestId": "d3f0c96a-51b0-4166-a850-f4175dde7323"
      },
      "result": {
          "message": "successful",
          "code": "200",
          "data": {
              "clusterRegion": "ap-southeast-1",
              "ProjectName": "ali-log-test-project",
              "CreateTime": "2017-06-08 20:22:41"
          },
          "success": true
      }
  }
```

设置如下：

图 3-1: 索引设置

其中：

- ①表示可查询json字段中所有string和bool数据。
- ②表示可查询long类型数据。
- ③表示配置的字段可进行SQL分析。

示例1：查询string、bool类型

```
class : cental*
message.traceInfo.requestId : 92.137_1518139699935_5599
message.param.projectName : ali-log-test-project
message.success : true
```

说明：

- json内字段无需配置。
- json map、array 自动展开，支持多层嵌套，每一层以”.”进行分割。

示例2：查询Double、Long类型

```
latency>40
message.usedTime > 40
```

说明：

需要对json内字段独立配置，字段必须不在array。

示例3：组合查询

```
class : cental* and message.usedTime > 40 not message.param.projectNam
e:ali-log-test-project
```

其他

如果您查询需要检索的日志数据量很大时（如查询时间跨度非常长，其中数据量在百亿以上等），则一次查询请求无法检索完所有数据。在这种情况下，日志服务会把已有的数据返回给您，并在返回结果中告知您该查询结果并不完整。

如此同时，服务端会缓存 15 分钟内的查询结果。当查询请求的结果有部分被缓存命中，则服务端会在这次请求中继续扫描未被缓存命中的日志数据。为了减少您合并多次查询结果的工作量，日志服务会把缓存命中的查询结果与本次查询新命中的结果合并返回给您。

因此日志服务可以让您通过以相同参数反复调用该接口来获取最终完整结果。

3.2 索引数据类型

3.2.1 文本类型

和搜索引擎类似，文本类（Text）数据查询基于词（Term）的命中，因此需要配置分词符、大小写敏感，包含中文（中文分词）选项。

配置说明

大小写敏感

原始日志查询时是否区分大小写。例如原始日志为“internalError”：

- **false**（不区分），即查询关键字“INTERNALERROR”和“internalerror”都能查询到样例日志。
- **true**（区分），只能通过关键字“internalError”查询到样例日志。

分词符

原始日志内容根据分词符可以将日志内容切分成多个关键词。

例如我们要查询如下日志内容：

```
/url/pic/abc.gif
```

- 不设置任何分词符，整个字符串会作为一个独立单词`/url/pic/abc.gif`，只有通过该完整字符串，或通过模糊查询`/url/pic/*`才能找到。
- 如果设置分词符为`/`，则原始日志被切分为`url`、`pic`和`abc.gif`三个单词，可以使用任意一个单词或单词模糊查询都可以找到该日志，例如`url`、`abc.gif`或`pi*`，也可以使用`/url/pic/abc.gif`进行查询(查询时会被拆分为`url and pic and abc.gif`三个条件)。
- 如果设置分词符为`/.`，则原始日志被切分为`url`、`pic`、`abc`和`gif`四个单词。

说明：
通过设置合理的分词符，可以放宽查询的范围。

包含中文

如果日志中包含中文，需要打开中文分词。例如对如下日志内容：

```
buyer:用户小李飞刀lee
```

默认分词符为”:”，则原始日志会被拆分为`buyer`、`用户小李飞刀lee`这两个单词，如果搜索`用户`，则不会返回`lee`，如果开启**包含中文**选项后，日志服务后台分词器会智能去理解中文含义，并将日志拆分为`buyer`、`用户`、`小李`、`飞刀`和`lee`五个单词，无论使用`飞刀` 或`小李飞刀`（会被解析为：`小李 and 飞刀`) 都可以查找到日志。

说明：

中文分词对写入速度会有一定影响，请根据需求谨慎设置。

全文索引

全文查询（索引）默认会将整条日志（除Time以外所有字段、包括Key）作为文本类型，全文查询默认不需要指定key。例如对以下由4个字段组成的日志（时间/status/level/message）。

```
[20180102 12:00:00] 200,error,some thing is error in this field
```

- time:2018-01-02 12:00:00
- level:”error”
- status:200
- message:”some thing is error in this field”

当打开全文索引时，整条日志中会根据Key:Value + “空格”模式组装成一条文本数据，例如：

```
status:200 level:error message:"some thing is error in this field"
```

说明：

- 全文检索时不需要输入前缀，在检索过程中搜索error时（level和message两个字段中error都会被命中）。
- 全文检索需要设置分词符，例如当设置分词符为“ ”时，可以“status:200”作为一个短语；如果分词符为“ :“时，“status”和“200”分别会作为2个独立短语。
- 数值类会被作为文本处理，例如200可以检索到该日志，时间字段（time) 不会被作为文本处理。
- 当输入Key时整条日志也会被命中，例如“status”。

3.2.2 Json类型

Json是由文本、布尔、数值、数组（Array）和图（Map）构成的组合类型数据。

配置说明

文本类数据

对JSON类型字段，会自动识别text/bool类型字段。

例如，以下jsonkey可以通过`jsonkey.key1:"text_value"`、`jsonkey.key2:true`等条件进行查询。

```
jsonkey: {
   key1:text_value,
   key2:true,
   key3:3.14
}
```

数值类数据

非JSON array中的double、long类型数据，可通过设置类型，并指定路径后进行查询。

例如，jsonkey.key3这个字段的类型为double，则查询语句为：

```
 jsonkey.key3 > 3
```

非完全合法JSON

对于非完全合法JSON数据，会尽可能解析有效内容，直到遇到非法部分结束。

例如：

```
"json_string":
{
     "key_1" :  "value_1",
     "key_map" :
      {
             "key_2" : "value_2",
             "key_3" : "valu
```

key_3之后的数据被截断丢失，对于这种缺失的日志可正确解析到`json_string.key_map.key_2`这个字段。

注意事项

- 不支持json object、json array类型。
- 字段不能在json array中。
- bool类型字段可以转成text类型。

查询语法

指定Key查询需要加上JSON中父路径的前缀，文本、数值类查询语法与其他类型相同，详情请参见查询语法。

3.2.3 数值类型

在配置索引时，您可以将字段配置为数值类型，并通过数值范围查询键值。

配置说明

支持类型：`long`（长整数）或者`double`（小数），当设置为数值类型后对于该键的查询只能通过数值范围。

查询示例

查询键值范围为(1000 2000]的longkey，可以使用以下查询方式：

- 数值类查询语法，例如：

```
longKey > 1000 and longKey <= 2000
```

- 也可以使用区间查询语法，例如：

```
longKey in (1000 2000]
```

更多语法请参见查询语法。

3.3 查询语法

为了能够帮助您更有效地查询日志，Log Service 提供一套查询语法用以表达查询条件。您可以通过 Log Service API 中的 *GetLogs* 和 *GetHistograms* 接口或者在 Log Service 控制台的查询页面指定查询条件。本文档详细说明该查询条件的语法。

索引类型

日志服务支持通过两种模式建立对日志库索引：

- 全文索引：将整行日志作为整体进行查询，既不区分键与数值（Key，Value）。
- 键值索引：指定键（Key）情况下进行查询，例如 *`FILE:app`*、*`Type:action`*。在该键下被包含字符串都会命中。

语法关键词

LogSearch 查询条件支持如下关键字：

名称	语义
and	双目运算符。形式为`query1 and query2`，表示`query1`和`query2`查询结果的交集。如果多个单词间没有语法关键词，默认是`and`的关系。
or	双目运算符。形式为`query1 or query2`，表示`query1`和`query2`查询结果的并集。

名称	语义
not	双目运算符。形式为`query1 not query2`，表示符合`query1`并且不符合`query2`的结果，相当于`query1 - query2`。如果只有`not query1`，那么表示从全部日志中选取不包含`query1`的结果。
(,)	左右括号用于把一个或多个子 query 合并成一个 query，用于提高括号内 query 的优先级。
:	用于 key-value 对的查询。`term1:term2`构成一个 key-value 对。如果 key 或者 value 内有空格、冒号`:`等保留字符，需要用双引号`""`把整个 key 或者 value 包括起来。
“	把一个关键词转换成普通的查询字符。左右引号内部的任何一个 term 都会被查询，而不会当成语法关键词。或者在 key-value 查询中把左右引号内的所有 term 当成一个整体。
\	转义符。用于转义引号，转义后的引号表示符号本身，不会当成转义字符，例如`"\""`。
|	管道运算符，表示前一个计算的基础上进行更多计算，例如 query1 | timeslice 1h | count。
timeslice	时间分片运算符，表示多长时间的数据作为一个整体进行计算，使用方式有 timeslice 1h，timeslice 1m，timeslice 1s 分别表示以 1 小时，1 分钟，1s 作为一个整体。例如 query1 | timeslice 1h | count 表示查询 query 这个条件，并且返回以 1 小时为时间分片的总次数。
count	计数运算符，表示日志条数。
*	模糊查询关键字，用于替代 0 个或多个字符，例如：`que*`，会返回`que` 开头的所有命中词。 **说明：** 模糊查询最多返回100个包含符合关键词的日志。
?	模糊查询关键字，用于替代一个字符，比如`qu?ry`，会返回以`qu`开头，以`ry`结尾，并且中间还有一个字符的所有命中词。
`__topic__`	查询某个 topic 下数据，新的语法下，可以在 query 中查询 0 个或多个 topic 的数据，例如`__topic__:mytopicname`。
`__tag__`	查询某个 tag key 下某个 tag value，例如`__tag__:tagkey:tagvalue`。

名称	语义
source	查询某个 IP 的数据，例如`source:127.0.0.1`。
>	查询某个字段下大于某个数值的日志，例如`latency > 100`。
>=	查询某个字段下大于或等于某个数值的日志，例如`latency >= 100`。
<	查询某个字段下小于某个数值的日志，例如`latency < 100`。
<=	查询某个字段下小于或等于某个数值的日志，例如`latency <= 100`。
=	查询某个字段下等于某个数值的日志，例如`latency = 100`。
in	查询某个字段处于某个范围内的日志，使用中括号表示闭区间，使用小括号表示开区间，括号中间使用两个数字，数字中间为若干个空格。例如`latency in [100 200]`或`latency in (100 200]`。

说明：

- 语法关键词不区分大小写。
- 语法关键字的优先级由高到底排序为：`> " > ( ) > and not > or`。
- Log Service 还保留以下关键字的使用权，如果您需要使用以下关键字，请使用双引号包含起来：`sort asc desc group by avg sum min max limit` 。
- 同时配置全文索引和键值索引时，如果两者的分词字符不一样，那么使用全文查询方式时数据无法查出。
- 使用数值查询的前提条件是给该列设置类型为double或long，如果您没有设置类型，或者数值范围查询的语法不正确，日志服务会将该查询条件解释成全文索引，可能与您的期望的结果不同。
- 如果您之前把某列配置为文本类型，现在改成数值类型，那么之前的数据只支持=查询。

查询示例

1. 同时包含 a 和 b 的日志： `a and b` 或者 `a b`
2. 包含 a 或者包含 b 的日志：`a or b`
3. 包含 a 但是不包含 b 的日志：`a not b`
4. 所有日志中不包含 a 的日志：`not a`

5. 查询包含 a 而且包含 b，但是不包括 c 的日志：`a and b not c`
6. 包含 a 或者包含 b，而且一定包含 c 的日志：`(a or b ) and c`
7. 包含 a 或者包含 b，但不包括 c 的日志：`(a or b ) not c`
8. 包含 a 而且包含 b，可能包含 c 的日志：`a and b or c`
9. FILE 字段包含 apsara的日志： `FILE:apsara`
10. FILE 字段包含 apsara 和 shennong 的日志：`FILE:"apsara shennong"` 或者 `FILE: apsara FILE: shennong` 或者 `FILE:apsara and FILE:shennong`
11. 包含 and 的日志：`and`
12. FILE 字段包含 apsara 或者 shennong 的日志：`FILE:apsara or FILE:shennong`
13. file info 字段包含 apsara 的日志：`"file info":apsara`
14. 包括引号的日志：`\"`
15. 查询以 shen 开头的所有日志：`shen*`
16. 查询 FILE 字段下，以 shen 开头的所有日志：`FILE:shen*`
17. 查询以 shen 开头，以 ong 结尾，中间还有一个字符的日志：`shen?ong`
18. 查询包括以 shen 开头，并且包括以 aps 开头的日志：`shen* and aps*`
19. 查询以 shen 开头的日志的分布，时间片为 20 分钟：`shen*| timeslice 20m | count`
20. 查询 topic1 和 topic2 下的所有数据： `__topic__:topic1 or __topic__ : topic2`
21. 查询 tagkey1 下 tagvalue2 的所有数据：`__tag__ : tagkey1 : tagvalue2`
22. 查询latency大于等于100，并且小于200的所有数据，有两种写法：`latency >=100 and latency < 200`或`latency in [100 200)`。
23. 查询latency 大于100的所有请求，只有一种写法： `latency > 100`。
24. 查询不包含爬虫的日志，并且http_referer中不包含opx的日志： `not spider not bot not http_referer:opx`。
25. 查询cdnIP字段为空的日志： `not cdnIP:""`。
26. 查询cdnIP字段不存在的日志：`not cdnIP:*`。
27. 查询存在cdnIP字段的日志：`cdnIP:*`。

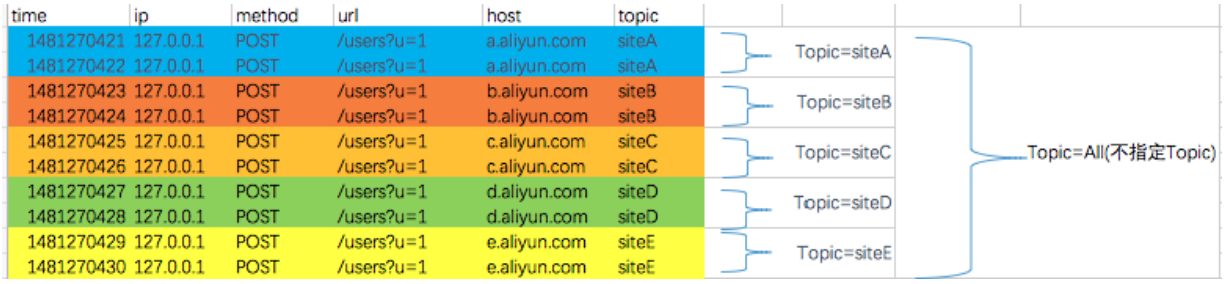
指定或跨 Topic 查询

每个 Logstore 根据 Topic 可以划分成一个或多个子空间，当进行查询时，指定 topic 可以限定查询范围，达到更快速度。因此我们推荐对 logstore 有二级分类需求的用户使用 topic 进行划分。

当指定一个或多个 topic 进行查询时，仅从符合条件的 topic 中进行查询。但不输入 topic，默认查询所有 topic 下的数据。

例如，使用Topic来划分不同域名下日志：

图 3-2: 日志Topic



Topic 查询语法：

- 支持查询所有 topic 下的数据，在查询语法和参数中都不指定 topic 意味着查询所有 topic 的数据。
- 支持在 query 中查询 topic，查询语法为 `__topic__:topicName`。同时仍然支持旧的模式，在 url 参数中指定 topic。
- 支持查询多个 topic，例如 `__topic__:topic1 or __topic__:topic2` 表示查询 topic1 和 topic2 下的数据的并集。

模糊查询

日志服务支持单词模糊查询，指定一个64个字符以内的词，在词的中间或者末尾加上模糊查询关键字，即`*`和`?`，日志服务会在所有日志中为您查询到符合条件的100个词，并返回包含这100个词并满足查询条件的所有日志。

限制说明：

- 查询时必须指定前缀，即`*`和`?`不能出现在词的开头。
- 指定的词越精确，查询结果越精确。
- 查询的词超过64个字符，无法使用模糊查询。建议您把查询的词长度缩小到64个字符以下。

3.4 上下文查询

当您展开一份日志文件，每一条日志都记录一个事件，并且往往不是孤立存在的，连续的若干条日志可以回放整个事件序列的发生过程。

日志上下文查询是指定日志来源（机器 + 文件）和其中一条日志，将该日志在原始文件中的前若干条（上文）或后若干条日志（下文）也查找出来，尤其是在 DevOps 场景下对于理清问题来龙去脉来说可谓是一把利器。

日志服务控制台提供专门的查询页面，您可以在控制台查看指定日志在原始文件中的上下文信息，体验类似于在原始日志文件中向上或向下翻页功能。通过查看指定日志的上下文信息，您可以在业务故障排查中快速查找相关故障信息，方便定位问题。

应用场景

例如，O2O 外卖网站在服务器上的程序日志里会记录一次订单成交的轨迹：

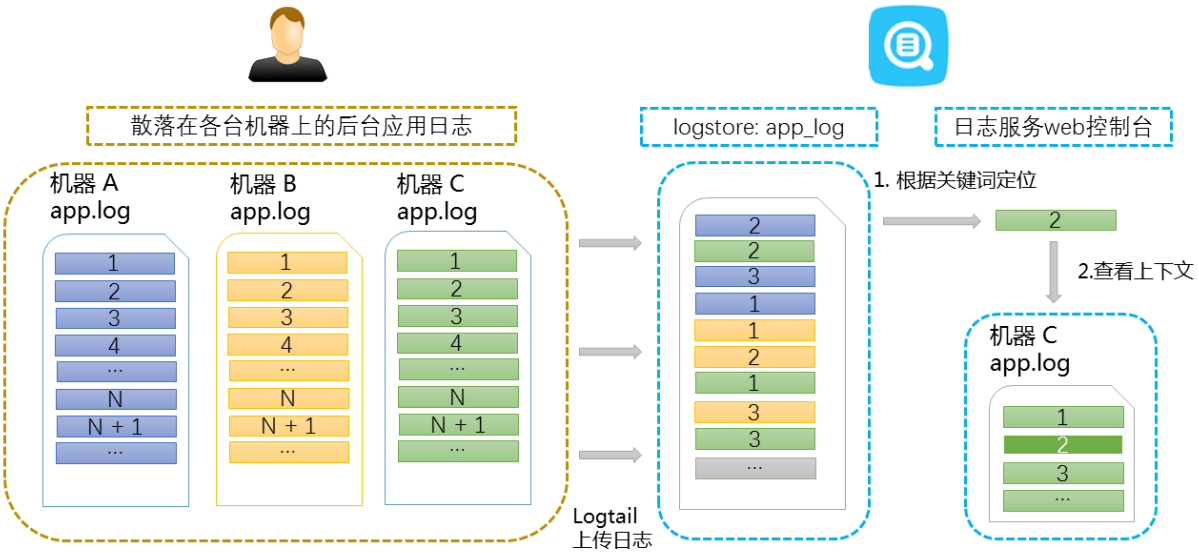
用户登录 > 浏览商品 > 点击物品 > 加入购物车 > 下单 > 订单支付 > 支付扣款 > 生成订单

如果用户下单失败了，运维人员需要快速定位问题原因。传统的上下文查询中，需要管理员相关人员添加机器登录权限，然后调查者依次登录应用所部署的每一台机器，以订单 ID 为关键词搜索应用程序日志文件，帮助判断下单失败原因。

在日志服务中，可以按照以下步骤排查：

1. 到服务器上安装日志采集客户端 Logtail，并到控制台上添加机器组、日志采集配置，然后 Logtail 开始上传增量日志。
2. 到日志服务供控制台日志查询页面，指定时间段根据订单 ID 找到订单失败日志。
3. 以查到的错误日志为基准，向上翻页直到发现与之相关的其它日志信息（例如：信用卡扣款失败）。

图 3-3: 应用场景



功能优势

- 不侵入应用程序，日志文件格式无需改动。
- 在日志服务控制台上可以查看任意机器、文件的指定日志上下文信息，解放了过去需要登录每台机器查看日志文件的痛苦。
- 结合事件发生的时间线索，在日志服务控制台指定时间段快速定位可疑日志后再进行上下文查询，往往可以事半功倍。
- 不用担心服务器存储空间不足或日志文件轮转（rotate）造成的数据丢失，到日志服务控制台上随时可以查看过往的数据。

前提条件

- 使用 *Logtail* 采集日志 上传数据到日志库，除创建机器组、采集配置以外无需其他配置。
  或使用Producer相关的SDK上传，例如 Producer Library、Log4J、LogBack、C-Producer Library等。
- 开启索引。

> **说明：**
> 上下文查询功能暂不支持syslog日志。

操作步骤

1. 进入日志服务控制台。
2. 选择需要的项目，单击项目名称。
3. 在**Logstore**列表页面，选择所需的日志库并单击日志索引列下的**查询**进入查询界面。
4. 输入您的查询分析语句，选择查询时段并单击**搜索**。

   查询结果页中任一条日志的左侧有**上下文浏览**按钮，表明该日志支持上下文查看功能。

   图 3-4: 查询日志

5. 选中一条日志，单击**上下文浏览**。在右侧弹出页面中查看目标日志的上下文日志。
6. 使用鼠标在当前页面上下滚动查看选中日志周边的日志信息。如需要继续查看上文和下文，单击**更早** 或 **更新** 进行翻页浏览。

图 3-5: 查询日志

3.5 其他功能

日志服务查询分析功能除了提供日志内容的各种语句查询能力以外，还提供以下多种扩展功能优化您的查询。

- 原始日志
- 统计图表
- 上下文查询
- 快速分析
- 快速查询
- 标签

- 仪表盘
- 另存为告警

原始日志

开启索引后，在检索框输入关键字、选择查询时段，单击**搜索**后，即可看到日志数量的直方图、原始日志及统计图表。

日志数量的直方图即日志检索的命中数量在时间上的分布，您可以通过直方图查看某个时间段的日志数量变化，单击长方形区域可以缩小时间范围，查看长方形区域表示的时间范围内的日志命中情况，为您的日志检索结果提供更精细的展示方式。

在原始日志页签中，您可以按时间排序，查看被命中的日志内容。

- 单击列名**时间**旁的三角符号，可切换**时间正序**或**时间倒序**。
- 单击列名**内容**旁的三角符号，可以选择日志内容**换行显示**或者**整行显示**。
- 单击日志内容中的value关键字，可以查看包含该关键字的所有日志内容。
- 单击原始日志页签右上角的**下载**图标，可下载CSV格式的查询结果，单击**设置**图标，可在原始日志显示结果中增加字段为列名的显示列，您可以更直观地在新增列中查看每条原始日志的目标字段内容。
- 单击上下文查看该日志的前后各15条日志。更多信息请参考上下文查询。

> **说明：**
> 上下文查询功能目前仅支持使用Logtail上传的数据。

图 3-6: 原始日志

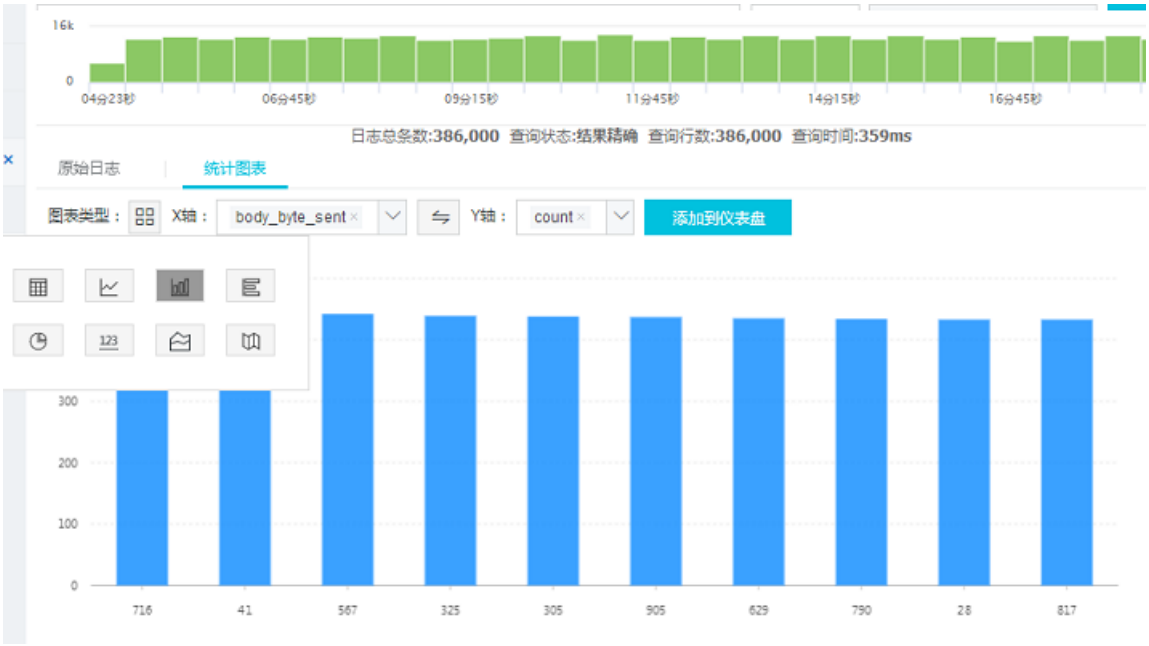
统计图表

开启索引并输入查询和分析语句后，您可以在统计图表页签中查看日志的统计结果。

- 提供多种数据显示方式：表格、折线图、柱状图、条形图、饼图、数字、面积图、地图。

  您可以根据统计分析的需要选择统计图表类型。
- 您可以调整X轴和Y轴的显示内容，以获取更为符合需求的显示结果。
- 将分析结果添加到仪表盘，详细内容请参考仪表盘。

图 3-7: 仪表盘



上下文查询

日志服务控制台提供专门的查询页面，您可以在控制台查看指定日志在原始文件中的上下文信息，体验类似于在原始日志文件中向上或向下翻页功能。通过查看指定日志的上下文信息，您可以在业务故障排查中快速查找相关故障信息，方便定位问题。更多信息请参考上下文查询。

快速分析

日志服务（Log Service）快速分析功能提供给用户一键交互式查询体验，意图帮助用户快速分析某一字段在一段时间内的分布情况，减少用户索引关键数据的成本。详细说明请参考快速分析。

快速查询

单击查询界面右上角的**快速查询**，可以将您当前的查询动作保存为快速查询，下次进行该查询动作时，不需要手动输入查询语句，只需在左侧**快速查询**页签，即可再次进行该项查询动作。

您也可以在报警规则中使用该快速查询条件。如您已将该快速查询添加到**标签**中，也可以在标签中直接进入该快速查询。

标签

日志服务查询界面首页左侧为您提供了标签列表，您可以将以下三种数据页面添加至标签列表中：

- Logstore
- 快速查询
- 仪表盘

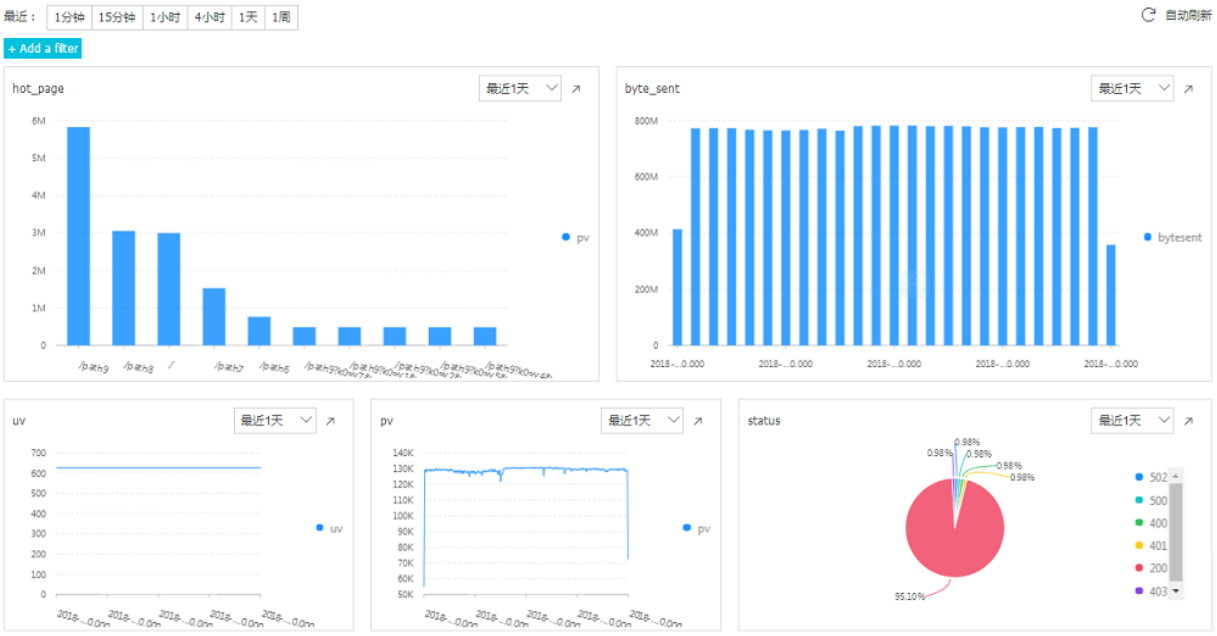
标签列表为您提供了简单便捷的页面打开方式，您可以直接在标签列表中单击打开Logstore、已保存的快速查询和仪表盘。在标签列表处单击**添加标签**，在页面右侧的弹出菜单中选择您想要添加为标签的Logstore、快速查询和仪表盘。需要删除标签时，单击标签列表中需要删除的标签名称右侧的X号即可删除。

图 3-8: 标签

仪表盘

日志服务提供仪表盘功能，支持将查询分析语句进行可视化展示。详细信息请参考仪表盘。

图 3-9: 仪表盘



另存为告警

日志服务支持基于您的日志查询结果进行告警，您可以通过配置规则将具体告警内容以站内通知或者钉钉发送给您。

配置流程为：

1. 配置快速查询。
2. 创建告警规则。
3. 配置通知方式。
4. 查看告警记录。

详细信息请参考设置告警。

3.6 快速查询

快速查询是日志服务提供的一键查询分析功能。

前提条件

已开启并设置索引。

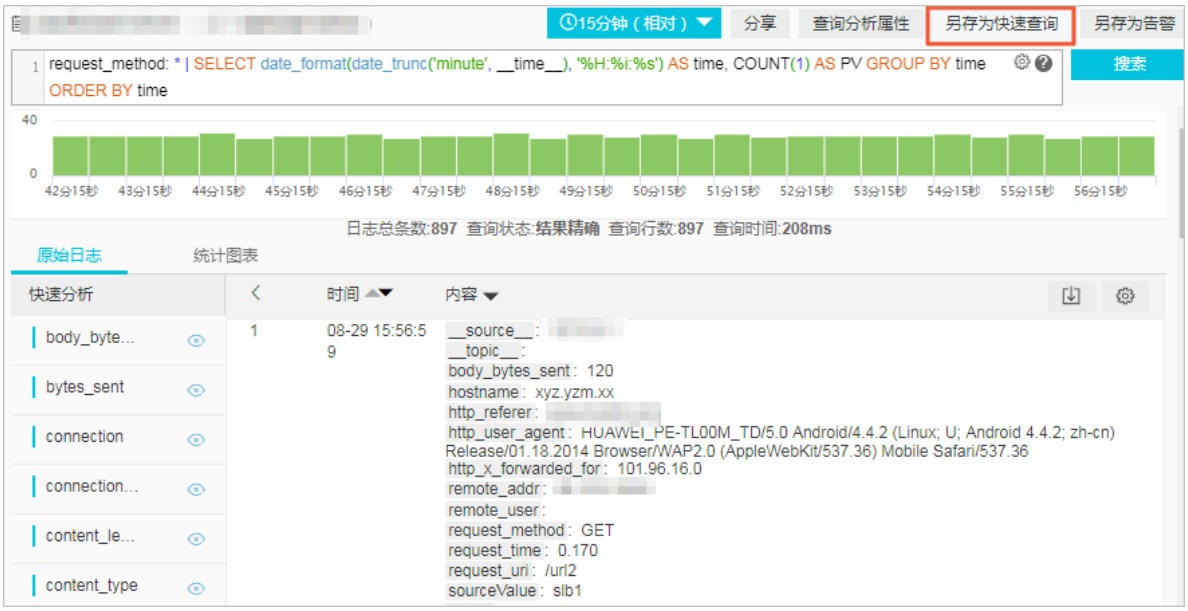
背景信息

当您需要经常查看某一查询分析语句的结果时，可以将其另存为快速查询，下次进行该查询动作时，不需要手动输入查询语句，只需在查询页面左侧单击该快速查询的名称，即可再次进行该项查询动作。您也可以在告警规则中使用该快速查询条件。日志服务会定期执行该快速查询语句，并在查询结果满足预设条件时发送告警信息。

设置下钻分析时，如果需要将下钻事件设置**跳转到快速查询**，必须提前配置快速查询，并在查询语句中设置**占位符**。

操作步骤

1. 登录日志服务控制台，单击Project名称。
2. 在**Logstore**列表单击**查询分析**列下的**查询**。
3. 输入您的查询分析语句，设置时间范围，并单击**搜索**。
4. 单击页面右上角的**另存为快速查询**。



5. 设置快速查询属性。

   a) 设置**快速查询名称**。

   - 名称仅支持小写字母、数字、连字符（-）和下划线（_）。
   - 必须以小写字母或数字开头和结尾。
   - 名称长度为3~63个字符。

   b) 确认**日志库**、**日志主题**和**查询语句**。

   若**日志库**和**日志主题**不符合您的需求，请返回至查询页面进入正确的日志库并输入查询语句，再次单击**另存为快速查询**。

c) （可选）划选查询语句的部分内容，并单击生成变量。

生成的变量为占位符变量。您可以在变量名中为占位符变量命名。默认值是您划选时选中的词。

> **说明：**
> 如果其他图表的下钻事件为跳转到这个快速查询，且图表配置的变量和快速查询的变量名相同，单击其他图表时会执行跳转，占位符变量的默认值替换为触发下钻事件的图表值，并以替换变量后的查询语句执行查询。详细信息请查看下钻分析。

6. 单击确定，结束配置。