主讲老师：Fox

课前须知：
1.本节课是ES专题最后一节课，中间停一节课，然后是周瑜老师Dubbo3.0的讲解
2.ES JAVA Client会在项目实战电商搜索部分讲解

有道云笔记地址

```
文档：6. Logstash与FileBeat详解以及ELK整合...
链接：http://note.youdao.com/noteshare?id=cd88d72a1c76d18efcf7fe767e8c2d20&sub=D7819084A43243FFA52E8A8741795414
```

注意：本节课的命令和配置文件不要再pdf文件中复制，为存在格式问题，保存到有道云笔记后再操作

背景

日志管理的挑战：

- 关注点很多，任何一个点都有可能引起问题
- 日志分散在很多机器，出了问题时，才发现日志被删了
- 很多运维人员是消防员，哪里有问题去哪里

集中化日志管理思路：

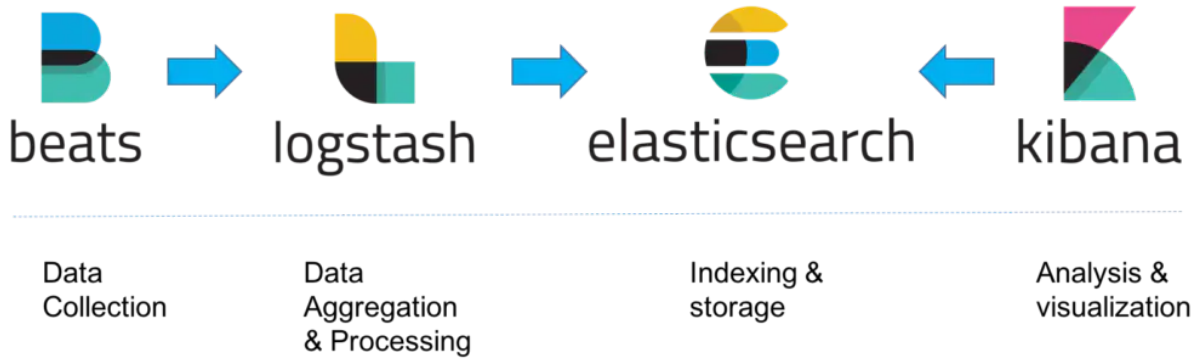
日志收集 ——》格式化分析 ——》检索和可视化 ——》 风险告警

ELK架构

ELK架构分为两种，一种是经典的ELK，另外一种是加上消息队列（Redis或Kafka或RabbitMQ）和Nginx结构。

经典的ELK

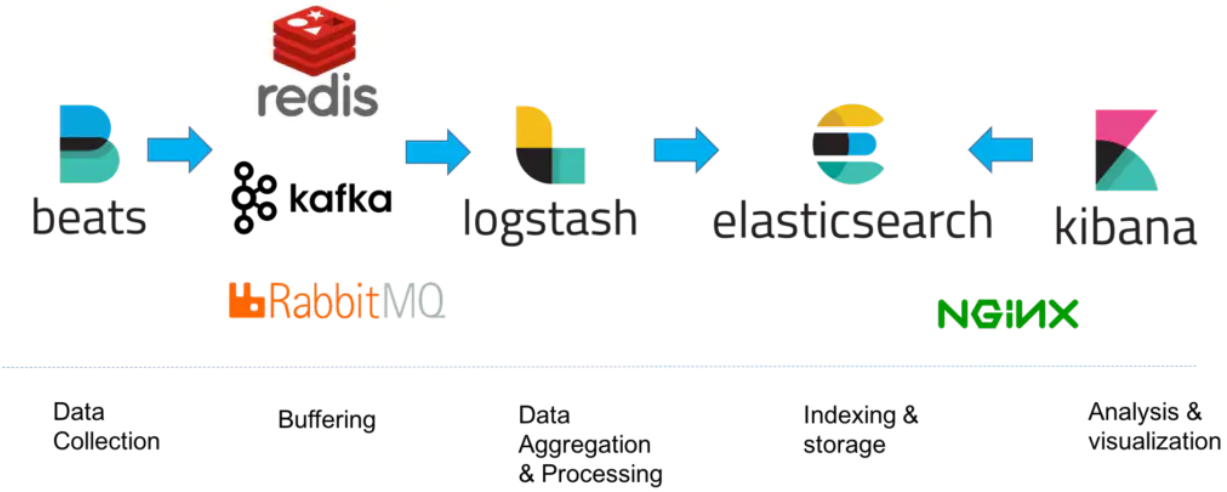
经典的ELK主要是由Filebeat + Logstash + Elasticsearch + Kibana组成，如下图：（早期的ELK只有Logstash + Elasticsearch + Kibana）



此架构主要适用于数据量小的开发环境，存在数据丢失的危险。

整合消息队列+Nginx架构

这种架构，主要加上了Redis或Kafka或RabbitMQ做消息队列，保证了消息的不丢失。



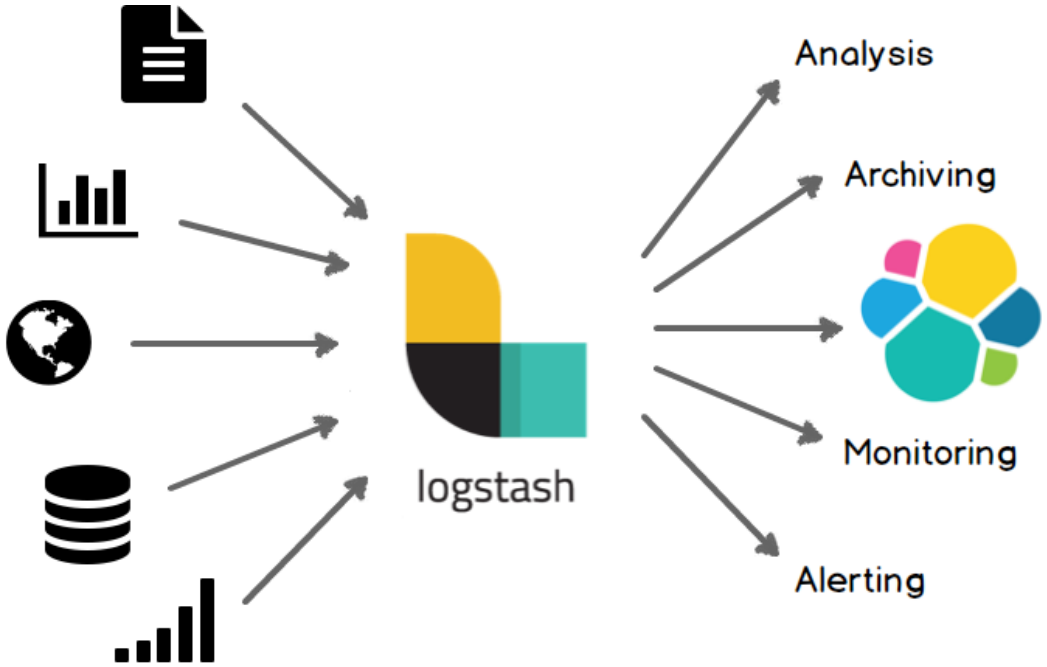
此种架构，主要用在生产环境，可以处理大数据量，并且不会丢失数据。

什么是Logstash

Logstash 是免费且开放的服务器端数据处理管道，能够从多个来源采集数据，转换数据，然后将数据发送到您最喜欢的存储库中。

https://www.elastic.co/cn/logstash/

应用：ETL工具 / 数据采集处理引擎



Logstash核心概念

Pipeline

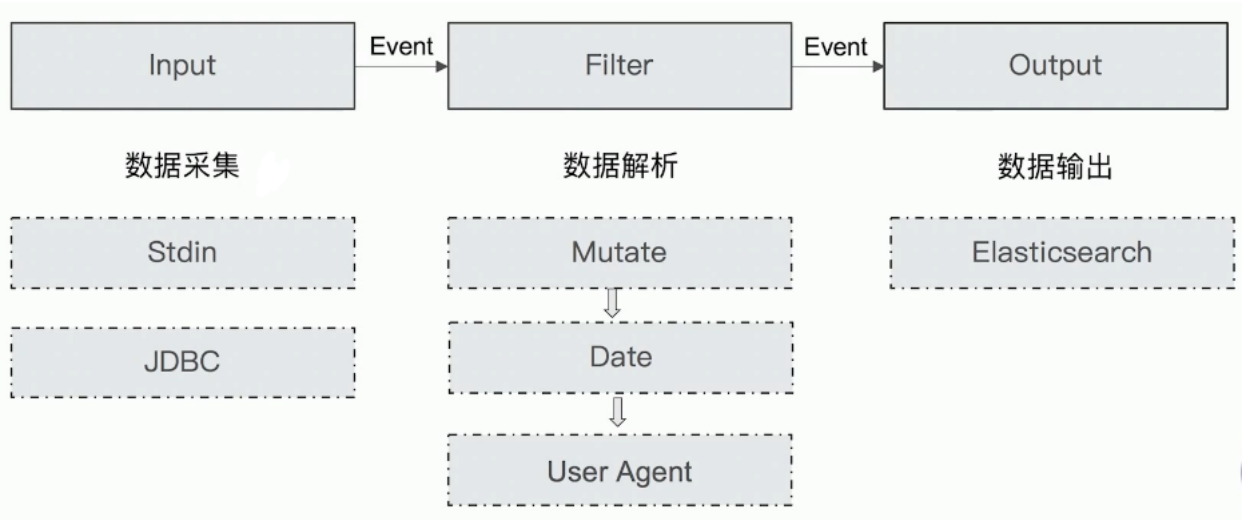
- 包含了input—filter-output三个阶段的处理流程
- 插件生命周期管理
- 队列管理

Logstash Event

- 数据在内部流转时的具体表现形式。数据在input 阶段被转换为Event，在output被转化成目标格式数据
- Event 其实是一个Java Object，在配置文件中，对Event 的属性进行增删改查

Codec (Code / Decode)

将原始数据decode成Event;将Event encode成目标数据



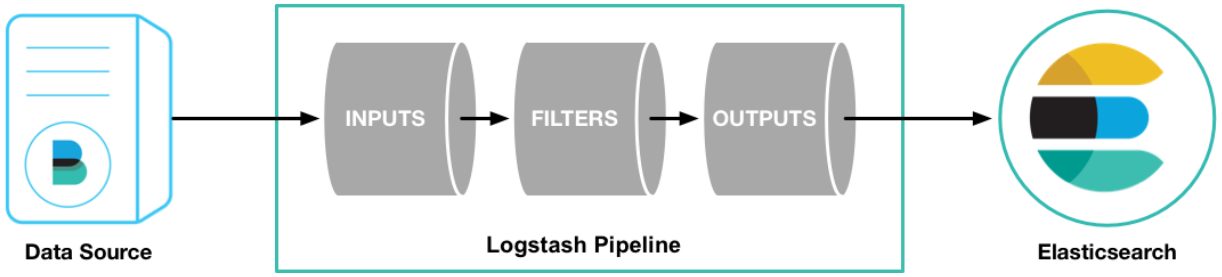
Logstash数据传输原理

1. 数据采集与输入：Logstash支持各种输入选择，能够以连续的流式传输方式，轻松地从日志、指标、Web应用以及数据存储中采集数据。

2. 实时解析和数据转换：通过Logstash过滤器解析各个事件，识别已命名的字段来构建结构，并将它们转换成通用格式，最终将数据从源端传输到存储库中。

3. 存储与数据导出：Logstash提供多种输出选择，可以将数据发送到指定的地方。

Logstash通过管道完成数据的采集与处理，管道配置中包含input、output和filter（可选）插件，input和output用来配置输入和输出数据源、filter用来对数据进行过滤或预处理。



Logstash配置文件结构

参考：https://www.elastic.co/guide/en/logstash/7.17/configuration.html

Logstash的管道配置文件对每种类型的插件都提供了一个单独的配置部分，用于处理管道事件。

```
input {
 stdin { }
}

filter {
 grok {
 match => { "message" => "%{COMBINEDAPACHELOG}" }
 }
 date {
  match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}

output {
 elasticsearch { hosts => ["localhost:9200"]}
 stdout { codec => rubydebug }
}
```

每个配置部分可以包含一个或多个插件。例如，指定多个filter插件，Logstash会按照它们在配置文件中出现的顺序进行处理。

```
#运行
bin/logstash -f logstash-demo.conf
```

Input Plugins

https://www.elastic.co/guide/en/logstash/7.17/input-plugins.html

一个 Pipeline可以有多个input插件

- Stdin / File
- Beats / Log4J /Elasticsearch / JDBC / Kafka /Rabbitmq /Redis
- JMX/ HTTP / Websocket / UDP / TCP
- Google Cloud Storage / S3
- Github / Twitter

Output Plugins

https://www.elastic.co/guide/en/logstash/7.17/output-plugins.html

将Event发送到特定的目的地，是 Pipeline 的最后一个阶段。

常见 Output Plugins:

- Elasticsearch
- Email / Pageduty
- Influxdb / Kafka / Mongodb / Opentsdb / Zabbix
- Http / TCP / Websocket

Filter Plugins

https://www.elastic.co/guide/en/logstash/7.17/filter-plugins.html

处理Event

内置的Filter Plugins:

- Mutate 一操作Event的字段
- Metrics — Aggregate metrics
- Ruby 一执行Ruby 代码

Codec Plugins

https://www.elastic.co/guide/en/logstash/7.17/codec-plugins.html

将原始数据decode成Event;将Event encode成目标数据

内置的Codec Plugins:

- Line / Multiline

- JSON / Avro / Cef (ArcSight Common Event Format)
- Dots / Rubydebug

Logstash Queue

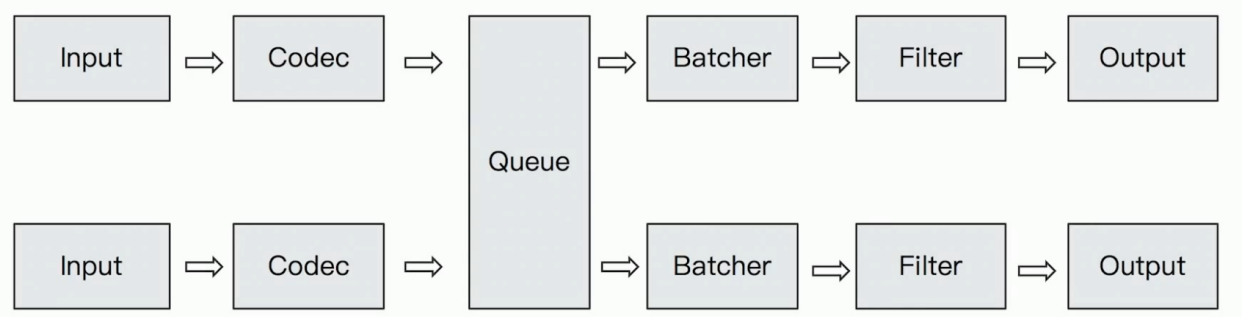
- In Memory Queue

进程Crash，机器宕机，都会引起数据的丢失

- Persistent Queue

机器宕机，数据也不会丢失; 数据保证会被消费; 可以替代 Kafka等消息队列缓冲区的作用

```
queue.type: persisted (默认是memory)
queue.max_bytes: 4gb
```



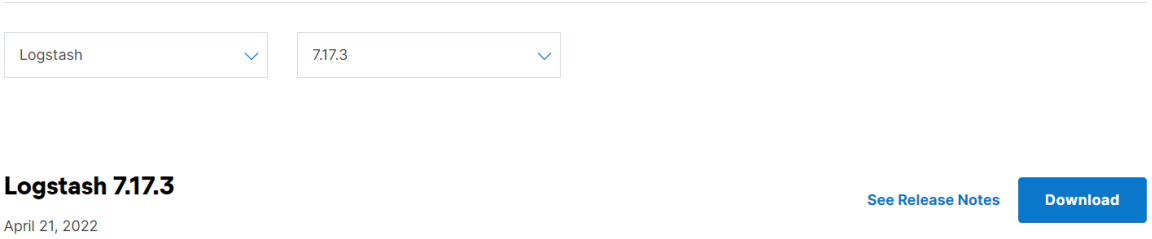
Logstash安装

logstash官方文档: https://www.elastic.co/guide/en/logstash/7.17/installing-logstash.html

1）下载并解压logstash

下载地址： https://www.elastic.co/cn/downloads/past-releases#logstash

选择版本：7.17.3



2）测试：运行最基本的logstash管道

```
cd logstash-7.17.3
#linux
#-e选项表示，直接把配置放在命令中，这样可以有效快速进行测试
```

```
bin/logstash -e 'input { stdin { } } output { stdout {} }'
#windows
.\bin\logstash.bat -e "input { stdin { } } output { stdout {} }"
```

测试结果：

```
non_running_pipelines=>[]}
hello
{
       "message" => "hello\r",
    "@timestamp" => 2022-06-10T05:36:47.382Z,
      "@version" => "1",
          "host" => "LAPTOP-UTD9471P"
}
hello fox
{
       "message" => "hello fox\r",
    "@timestamp" => 2022-06-10T05:37:03.789Z,
      "@version" => "1",
          "host" => "LAPTOP-UTD9471P"
}
```

window版本的logstash-7.17.3的bug:

windows出现错误提示could not find java; set JAVA_HOME or ensure java is in PATH

```
D:\apache\logstash-7.17.3\bin>logstash.bat -e "input { stdin { } } output { stdout {} }"
Using JAVA_HOME defined java: C:\Program Files\Java\jdk1.8.0_251
WARNING: Using JAVA_HOME while Logstash distribution comes with a bundled JDK.
DEPRECATION: The use of JAVA_HOME is now deprecated and will be removed starting from 8.0. Please configure LS_JAVA_HOME
 instead.
系统找不到指定的路径。
could not find java; set JAVA_HOME or ensure java is in PATH
```

修改setup.bat

```
if defined LS_JAVA_HOME (
  set JAVACMD=%LS_JAVA_HOME%\bin\java.exe
  echo Using LS_JAVA_HOME defined java: %LS_JAVA_HOME%
  if exist "%LS_HOME%\jdk" (
    echo WARNING: Using LS_JAVA_HOME while Logstash distribution comes wit
  )
) else if defined JAVA_HOME (
  set JAVACMD="%JAVA_HOME%\bin\java.exe"
  echo Using JAVA_HOME defined java: %JAVA_HOME%
  if exist "%LS_HOME%\jdk" (
    echo WARNING: Using JAVA_HOME while Logstash distribution comes with a
  )
  echo DEPRECATION: The use of JAVA_HOME is now deprecated and will be rem
) else (
  if exist "%LS_HOME%\jdk" (
```

脚本问题，去掉双引号

```
if defined LS_JAVA_HOME (
  set JAVACMD=%LS_JAVA_HOME%\bin\java.exe
  echo Using LS_JAVA_HOME defined java: %LS_JAVA_HOME%
  if exist "%LS_HOME%\jdk" (
    echo WARNING: Using LS_JAVA_HOME while Logstash distribution comes with a bundled JDK.
  )
) else if defined JAVA_HOME (
  set JAVACMD="%JAVA_HOME%\bin\java.exe"
  echo Using JAVA_HOME defined java: %JAVA_HOME%
  if exist "%LS_HOME%\jdk" (
    echo WARNING: Using JAVA_HOME while Logstash distribution comes with a bundled JDK.
  )
  echo DEPRECATION: The use of JAVA_HOME is now deprecated and will be removed starting from 8.0. Please configure LS_JAVA_HOME
) else (
  if exist "%LS_HOME%\jdk" (
    set JAVACMD=%LS_HOME%\jdk\bin\java.exe
    echo "Using bundled JDK: !JAVACMD!"
  ) else (
    for %%I in (java.exe) do set JAVACMD="%%~$PATH:I"
    echo "Using system java: !JAVACMD!"
  )
)

if not exist "%JAVACMD%"
  echo could not find java; set JAVA_HOME or ensure java is in PATH 1>&2
  exit /b 1
)
```

The double quotes here cause the JAVACMD to be empty

Codec Plugin测试

```
# single line
bin/logstash -e "input{stdin{codec=>line}}output{stdout{codec=> rubydebug}}"
bin/logstash -e "input{stdin{codec=>json}}output{stdout{codec=> rubydebug}}"
```

Codec Plugin —— Multiline

设置参数:

- pattern: 设置行匹配的正则表达式
- what : 如果匹配成功，那么匹配行属于上一个事件还是下一个事件
  - previous / next
- negate : 是否对pattern结果取反
  - true / false

```
# 多行数据，异常
Exception in thread "main" java.lang.NullPointerException
 at com.example.myproject.Book.getTitle(Book.java:16)
 at com.example.myproject.Author.getBookTitles(Author.java:25)
 at com.example.myproject.Bootstrap.main(Bootstrap.java:14)

```

```
# multiline-exception.conf
input {
 stdin {
  codec => multiline {
  pattern => "^\s"
  what => "previous"
  }
  }
}

filter {}

output {
 stdout { codec => rubydebug }
}

#执行管道
bin/logstash -f multiline-exception.conf
```

Input Plugin —— File

- 支持从文件中读取数据，如日志文件
- 文件读取需要解决的问题：只被读取一次。重启后需要从上次读取的位置继续(通过sincedb 实现)
- 读取到文件新内容，发现新文件
- 文件发生归档操作(文档位置发生变化，日志rotation)，不能影响当前的内容读取

Filter Plugin

Filter Plugin可以对Logstash Event进行各种处理，例如解析，删除字段，类型转换

- Date: 日期解析
- Dissect: 分割符解析
- Grok: 正则匹配解析
- Mutate: 处理字段。重命名，删除，替换
- Ruby: 利用Ruby 代码来动态修改Event

Filter Plugin - Mutate

对字段做各种操作:

- Convert : 类型转换
- Gsub : 字符串替换
- Split / Join /Merge: 字符串切割，数组合并字符串，数组合并数组
- Rename: 字段重命名
- Update / Replace: 字段内容更新替换
- Remove_field: 字段删除

Logstash导入数据到ES

1）测试数据集下载：https://grouplens.org/datasets/movielens/

A	B	C
movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller

2）准备logstash-movie.conf配置文件

```
input {
 file {
 path => "/home/es/logstash-7.17.3/dataset/movies.csv"
 start_position => "beginning"
 sincedb_path => "/dev/null"
 }
}
filter {
 csv {
  separator => ","
  columns => ["id","content","genre"]
  }

  mutate {
  split => { "genre" => "|" }
  remove_field => ["path", "host","@timestamp","message"]
  }

  mutate {

```

```
  split => ["content", "("]
  add_field => { "title" => "%{[content][0]}"}
  add_field => { "year" => "%{[content][1]}"}
  }

  mutate {
  convert => {
  "year" => "integer"
  }
  strip => ["title"]
  remove_field => ["path", "host","@timestamp","message","content"]
  }

}
output {
  elasticsearch {
  hosts => "http://localhost:9200"
  index => "movies"
  document_id => "%{id}"
  user => "elastic"
  password => "123456"
  }
  stdout {}
}
```

3）运行logstash

```
# linux
bin/logstash -f logstash-movie.conf
```

同步数据库数据到Elasticsearch

需求: 将数据库中的数据同步到ES，借助ES的全文搜索,提高搜索速度

- 需要把新增用户信息同步到Elasticsearch中
- 用户信息Update 后，需要能被更新到Elasticsearch
- 支持增量更新
- 用户注销后，不能被ES所搜索到

实现思路

- 基于canal同步数据（项目实战中讲解）
- 借助JDBC Input Plugin将数据从数据库读到Logstash

- 需要自己提供所需的 JDBC Driver；
- JDBC Input Plugin 支持定时任务 Scheduling，其语法来自 Rufus-scheduler，其扩展了 Cron，使用 Cron 的语法可以完成任务的触发；
- JDBC Input Plugin 支持通过 Tracking_column / sql_last_value 的方式记录 State，最终实现增量的更新；
- https://www.elastic.co/cn/blog/logstash-jdbc-input-plugin

JDBC Input Plugin实现步骤

1）拷贝jdbc依赖到logstash-7.17.3/drivers目录下

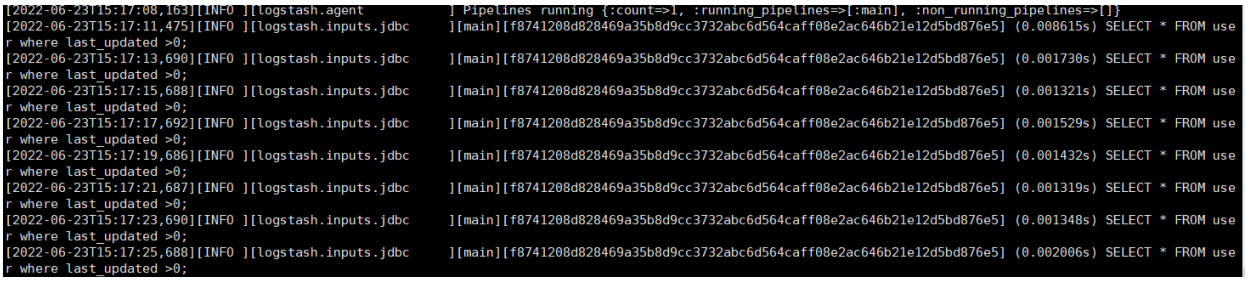
2）准备mysql-demo.conf配置文件

```
input {
 jdbc {
 jdbc_driver_library => "/home/es/logstash-7.17.3/drivers/mysql-connector-java-5.1.49.jar"
 jdbc_driver_class => "com.mysql.jdbc.Driver"
 jdbc_connection_string => "jdbc:mysql://localhost:3306/test?useSSL=false"
 jdbc_user => "root"
 jdbc_password => "123456"
 #启用追踪，如果为true，则需要指定tracking_column
 use_column_value => true
  #指定追踪的字段，
  tracking_column => "last_updated"
  #追踪字段的类型，目前只有数字(numeric)和时间类型(timestamp)，默认是数字类型
  tracking_column_type => "numeric"
  #记录最后一次运行的结果
  record_last_run => true
  #上面运行结果的保存位置
  last_run_metadata_path => "jdbc-position.txt"
  statement => "SELECT * FROM user where last_updated >:sql_last_value;"
  schedule => " * * * * * * *"
  }
 }
 output {
  elasticsearch {
  document_id => "%{id}"
```

```
  document_type => "_doc"
  index => "users"
  hosts => ["http://localhost:9200"]
  user => "elastic"
  password => "123456"
  }
  stdout{
  codec => rubydebug
  }
}
```

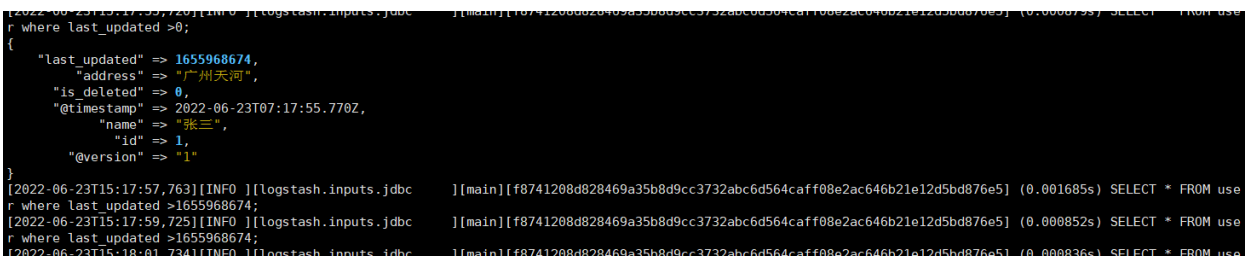
3）运行logstash

```
bin/logstash -f mysql-demo.conf
```

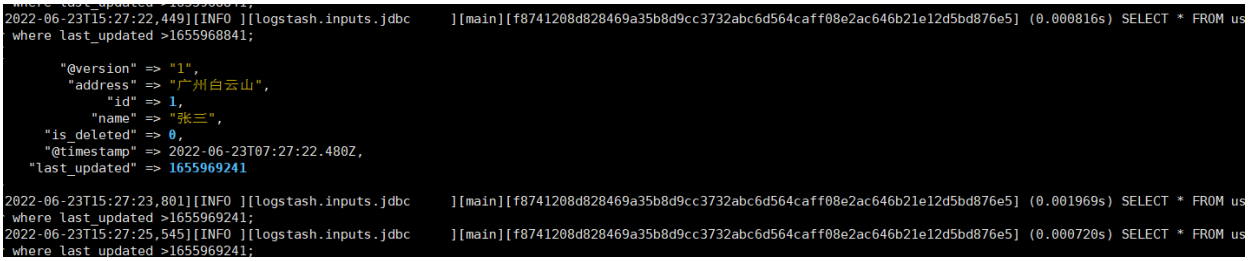


测试

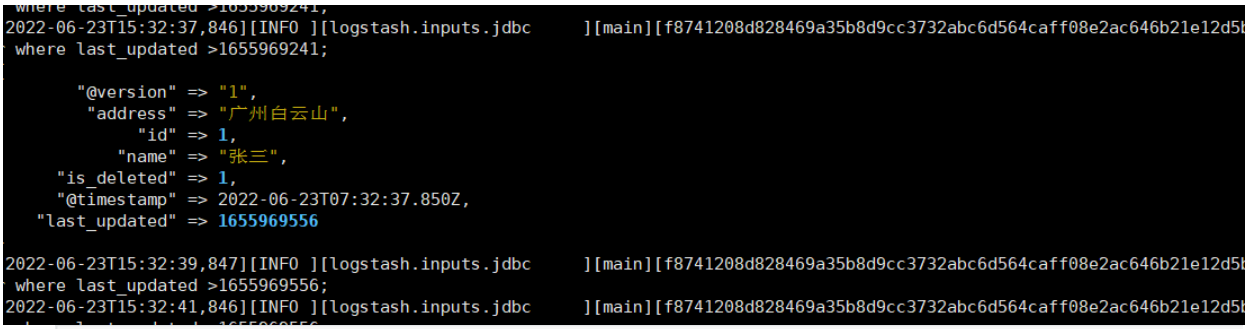
```
#user表
CREATE TABLE `user` (
 `id` int NOT NULL AUTO_INCREMENT,
 `name` varchar(50) DEFAULT NULL,
 `address` varchar(50) CHARACTER DEFAULT NULL,
 `last_updated` bigint DEFAULT NULL,
 `is_deleted` int DEFAULT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_
0900_ai_ci;
#插入数据
INSERT INTO user(name,address,last_updated,is_deleted) VALUES("张三","广
州天河",unix_timestamp(NOW()),0)
```



```
# 更新
update user set address="广州白云山",last_updated=unix_timestamp(NOW()) where name="张三"
```




```
#删除
update user set is_deleted=1,last_updated=unix_timestamp(NOW()) where name="张三"
```




```
#ES中查询
# 创建 alias，只显示没有被标记 deleted的用户
POST /_aliases
{
 "actions": [
 {
 "add": {
 "index": "users",
 "alias": "view_users",
 "filter" : { "term" : { "is_deleted" : 0} }
 }
 }
 ]
}

# 通过 Alias查询，查不到被标记成 deleted的用户
POST view_users/_search

POST view_users/_search
```

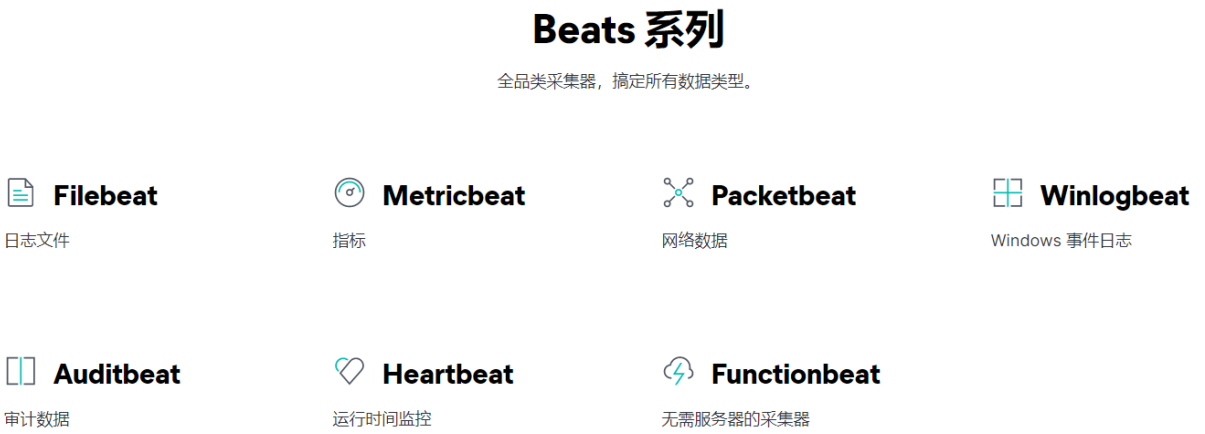
```
{
 "query": {
 "term": {
 "name.keyword": {
 "value": "张三"
 }
 }
 }
}
```

什么是Beats

轻量型数据采集器，文档地址：

https://www.elastic.co/guide/en/beats/libbeat/7.17/index.html

Beats 是一个免费且开放的平台，集合了多种单一用途的数据采集器。它们从成百上千或成千上万台机器和系统向 Logstash 或 Elasticsearch 发送数据。

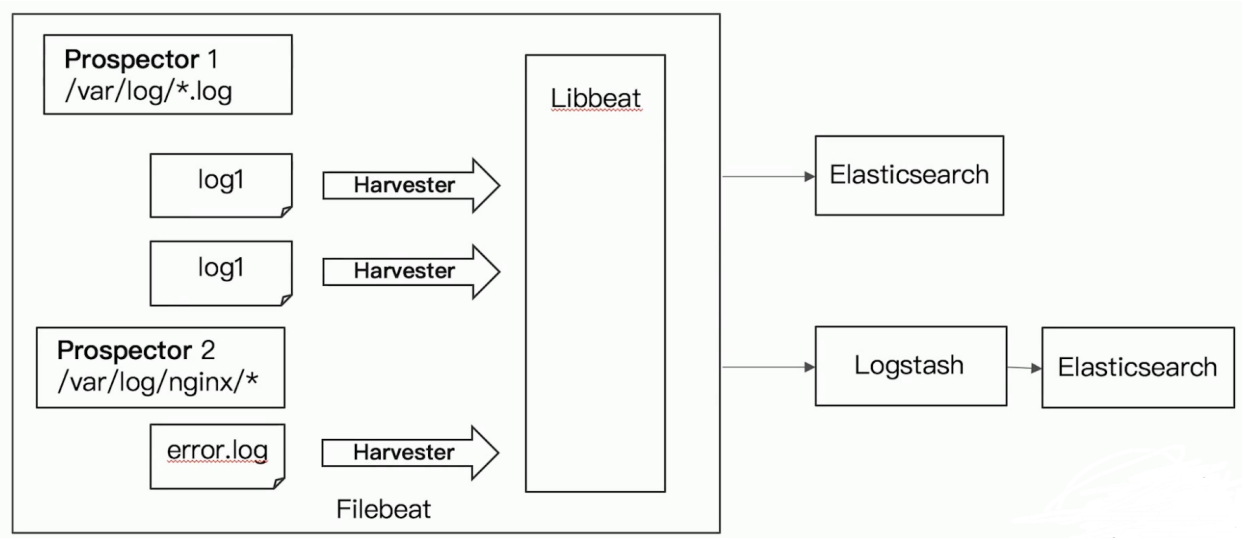


FileBeat简介

FileBeat专门用于转发和收集日志数据的轻量级采集工具。它可以作为代理安装在服务器上，FileBeat监视指定路径的日志文件，收集日志数据，并将收集到的日志转发到Elasticsearch或者Logstash。

FileBeat的工作原理

启动FileBeat时，会启动一个或者多个输入（Input），这些Input监控指定的日志数据位置。FileBeat会针对每一个文件启动一个Harvester（收割机）。Harvester读取每一个文件的日志，将新的日志发送到libbeat，libbeat将数据收集到一起，并将数据发送给输出（Output）。



logstash vs FileBeat

- Logstash是在jvm上运行的，资源消耗比较大。而FileBeat是基于golang编写的，功能较少但资源消耗也比较小，更轻量级。
- Logstash 和Filebeat都具有日志收集功能，Filebeat更轻量，占用资源更少
- Logstash 具有Filter功能，能过滤分析日志
- 一般结构都是Filebeat采集日志，然后发送到消息队列、Redis、MQ中，然后Logstash去获取，利用Filter功能过滤分析，然后存储到Elasticsearch中
- FileBeat和Logstash配合，实现背压机制。当将数据发送到Logstash或Elasticsearch时，Filebeat使用背压敏感协议，以应对更多的数据量。如果Logstash正在忙于处理数据，则会告诉Filebeat 减慢读取速度。一旦拥堵得到解决，Filebeat就会恢复到原来的步伐并继续传输数据。

Filebeat安装

https://www.elastic.co/guide/en/beats/filebeat/7.17/filebeat-installation-configuration.html

1）下载并解压Filebeat

下载地址：https://www.elastic.co/cn/downloads/past-releases#filebeat

选择版本：7.17.3

2）编辑配置

修改 filebeat.yml 以设置连接信息：

```
output.elasticsearch:
  hosts: ["192.168.65.174:9200","192.168.65.192:9200","192.168.65.204:9200"]
  username: "elastic"
  password: "123456"
setup.kibana:
  host: "192.168.65.174:5601"
```

3) 启用和配置数据收集模块

从安装目录中，运行：

```
# 查看可以模块列表
./filebeat modules list

#启用nginx模块
./filebeat modules enable nginx
#如果需要更改nginx日志路径,修改modules.d/nginx.yml
- module: nginx
  access:
  var.paths: ["/var/log/nginx/access.log*"]

#启用 Logstash 模块
./filebeat modules enable logstash
#在 modules.d/logstash.yml 文件中修改设置
- module: logstash
  log:
  enabled: true
  var.paths: ["/home/es/logstash-7.17.3/logs/*.log"]

```

4）启动 Filebeat

```
# setup命令加载Kibana仪表板。 如果仪表板已经设置，则忽略此命令。
./filebeat setup
# 启动Filebeat
./filebeat -e
```

ELK整合实战

案例：采集tomcat服务器日志

Tomcat服务器运行过程中产生很多日志信息，通过Logstash采集并存储日志信息至ElasticSearch中

使用FileBeats将日志发送到Logstash

1）创建配置文件filebeat-logstash.yml，配置FileBeats将数据发送到Logstash

```
vim filebeat-logstash.yml
chmod 644 filebeat-logstash.yml
#因为Tomcat的web log日志都是以IP地址开头的，所以我们需要修改下匹配字段。
# 不以ip地址开头的行追加到上一行
filebeat.inputs:
- type: log
 enabled: true
 paths:
 - /home/es/apache-tomcat-8.5.33/logs/*access*.*
  multiline.pattern: '^\\d+\\.\\d+\\.\\d+\\.\\d+ '
  multiline.negate: true
  multiline.match: after

output.logstash:
 enabled: true
 hosts: ["192.168.65.204:5044"]

```

- pattern：正则表达式
- negate：true 或 false；默认是false，匹配pattern的行合并到上一行；true，不匹配pattern的行合并到上一行
- match：after 或 before，合并到上一行的末尾或开头

2）启动FileBeat，并指定使用指定的配置文件

```
./filebeat -e -c filebeat-logstash.yml
```

可能出现的异常：

异常1：Exiting: error loading config file: config file ("filebeat-logstash.yml") can only be writable by the owner but the permissions are "-rw-rw-r--" (to fix the permissions use: 'chmod go-w /home/es/filebeat-7.17.3-linux-x86_64/filebeat-logstash.yml')

因为安全原因不要其他用户写的权限，去掉写的权限就可以了

```
chmod 644 filebeat-logstash.yml
```

异常2：Failed to connect to backoff(async(tcp://192.168.65.204:5044)): dial tcp 192.168.65.204:5044: connect: connection refused

FileBeat将尝试建立与Logstash监听的IP和端口号进行连接。但此时，我们并没有开启并配置Logstash，所以FileBeat是无法连接到Logstash的。

配置Logstash接收FileBeat收集的数据并打印

```
vim config/filebeat-console.conf
# 配置从FileBeat接收数据
input {
 beats {
 port => 5044
 }
}

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

测试logstash配置是否正确

```
bin/logstash -f config/filebeat-console.conf --config.test_and_exit
```

启动logstash

```
# reload.automatic：修改配置文件时自动重新加载
bin/logstash -f config/filebeat-console.conf --config.reload.automatic
```

测试访问tomcat，logstash是否接收到了Filebeat传过来的tomcat日志

Logstash输出数据到Elasticsearch

如果我们需要将数据输出值ES而不是控制台的话，我们修改Logstash的output配置。

```
vim config/filebeat-elasticSearch.conf
input {
 beats {
 port => 5044
 }
}

```

```
output {
 elasticsearch {
  hosts => ["http://localhost:9200"]
  user => "elastic"
  password => "123456"
  }
  stdout{
  codec => rubydebug
  }
}
```

启动logstash

```
bin/logstash -f config/filebeat-elasticSearch.conf --config.reload.automatic
```

ES中会生成一个以logstash开头的索引，测试日志是否保存到了ES。

思考：日志信息都保证在message字段中，是否可以把日志进行解析一个个的字段？例如：IP字段、时间、请求方式、请求URL、响应结果。

利用Logstash过滤器解析日志

从日志文件中收集到的数据包含了很多有效信息，比如IP、时间等，在Logstash中可以配置过滤器Filter对采集到的数据进行过滤处理，Logstash中有大量的插件可以供我们使用。

```
查看Logstash已经安装的插件
bin/logstash-plugin list
```

Grok插件

Grok是一种将非结构化日志解析为结构化的插件。这个工具非常适合用来解析系统日志、Web服务器日志、MySQL或者是任意其他的日志格式。

https://www.elastic.co/guide/en/logstash/7.17/plugins-filters-grok.html

Grok语法

Grok是通过模式匹配的方式来识别日志中的数据,可以把Grok插件简单理解为升级版本的正则表达式。它拥有更多的模式，默认Logstash拥有120个模式。如果这些模式不满足我们解析日志的需求，我们可以直接使用正则表达式来进行匹配。

grok模式的语法是：

```
%{SYNTAX:SEMANTIC}
```

SYNTAX（语法）指的是Grok模式名称，SEMANTIC（语义）是给模式匹配到的文本字段名。例如：

```
%{NUMBER:duration} %{IP:client}
duration表示：匹配一个数字，client表示匹配一个IP地址。
```

默认在Grok中，所有匹配到的的数据类型都是字符串，如果要转换成int类型（目前只支持int和float），可以这样：%{NUMBER:duration:int} %{IP:client}

常用的Grok模式

https://help.aliyun.com/document_detail/129387.html?scm=20140722.184.2.173

用法

```
filter {
 grok {
 match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }
 }
}
```

比如，tomacat日志

```
192.168.65.103 - - [23/Jun/2022:22:37:23 +0800] "GET /docs/images/docs-stylesheet.css HTTP/1.1" 200 5780
```

解析后的字段

字段名	说明
client IP	浏览器端IP
timestamp	请求的时间戳
method	请求方式（GET/POST）
uri	请求的链接地址
status	服务器端响应状态
length	响应的数据长度

grok模式

```
%{IP:ip} - - \[%{HTTPDATE:date}\] \"%{WORD:method} %{PATH:uri} %{DATA:protocol}\" %{INT:status} %{INT:length}
```

为了方便测试，我们可以使用Kibana来进行Grok开发：

修改Logstash配置文件

```
vim config/filebeat-console.conf

input {
 beats {
 port => 5044
 }
}

filter {
  grok {
  match => {
  "message" => "%{IP:ip} - - \[%{HTTPDATE:date}\] \"%{WORD:method} %{PATH:uri} %{DATA:protocol}\" %{INT:status:int} %{INT:length:int}"
  }
 }
 }

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

启动logstash测试

```
bin/logstash -f config/filebeat-console.conf --config.reload.automatic
```

使用mutate插件过滤掉不需要的字段

```
mutate {
 enable_metric => "false"
 remove_field => ["message", "log", "tags", "input", "agent", "host", "ecs", "@version"]
}
```

要将日期格式进行转换，我们可以使用Date插件来实现。该插件专门用来解析字段中的日期，官方说明文档：https://www.elastic.co/guide/en/logstash/7.17/plugins-filters-date.html
用法如下：

```
filter {
  date {
    match => [ "logdate", "MMM dd yyyy HH:mm:ss" ]
  }
}
```

将date字段转换为「年月日 时分秒」格式。默认字段经过date插件处理后，会输出到@timestamp字段，所以，我们可以通过修改target属性来重新定义输出字段。

```
date {
 match => ["date","dd/MMM/yyyy:HH:mm:ss Z","yyyy-MM-dd HH:mm:ss"]
 target => "date"
}
```

输出到Elasticsearch指定索引

index来指定索引名称，默认输出的index名称为：logstash-%{+yyyy.MM.dd}。但注意，要在index中使用时间格式化，filter的输出必须包含 @timestamp字段，否则将无法解析日期。

```
output {
 elasticsearch {
 index => "tomcat_web_log_%{+YYYY-MM}"
 hosts => ["http://localhost:9200"]
 user => "elastic"
 password => "123456"
 }
 stdout{
```

```
 codec => rubydebug
  }
}
```

注意：index名称中，不能出现大写字符

完整的Logstash配置文件

```
vim config/filebeat-filter-es.conf

input {
 beats {
 port => 5044
 }
}

filter {
  grok {
  match => {
  "message" => "%{IP:ip} - - \[%{HTTPDATE:date}\] \"%{WORD:method} %{PATH:uri} %{DATA:protocol}\" %{INT:status:int} %{INT:length:int}"
  }
 }
 mutate {
  enable_metric => "false"
  remove_field => ["message", "log", "tags", "input", "agent", "host", "ecs", "@version"]
 }
 date {
  match => ["date","dd/MMM/yyyy:HH:mm:ss Z","yyyy-MM-dd HH:mm:ss"]
  target => "date"
  }
 }

 output {
  stdout {
  codec => rubydebug
 }
 elasticsearch {
  index => "tomcat_web_log_%{+YYYY-MM}"
  hosts => ["http://localhost:9200"]
  user => "elastic"
  password => "123456"
```

```
34   }
35 }
```

启动logstash

```
1 bin/logstash -f config/filebeat-filter-es.conf --config.reload.automatic
```