

1. 为什么要使用 ELK

随着企业信息化进程的加速, 日志数据量急剧增加且来源多样、格式复杂, 传统的日志管理方式已难以满足需求。



ELK (Elasticsearch、Logstash、Kibana) 的引入, 正是为了应对这些挑战。ELK通过其强大的分布式搜索能力 (Elasticsearch)、灵活的数据采集与处理功能 (Logstash)、以及直观的数据可视化界面 (Kibana), 提供了高效、实时、可扩展且易用的日志管理解决方案, 帮助企业和开发人员更有效地管理和分析日志数据, 从而提高工作效率和问题解决速度。

以下是使用ELK的主要原因:

1. 集中化管理与高效检索日志:

- 在大型分布式系统中, ELK通过构建集中式日志系统, 实现所有节点上日志的统一收集、管理和访问, 提高定位问题的效率。
- Elasticsearch提供强大的检索特性, 能够快速查询问题日志, 显著提升运维人员的工作效率。

2. 全面的日志分析与系统监控:

- ELK能够管理和分析包括系统日志、应用程序日志和安全日志在内的多种日志, 帮助系统运维和开发人员了解服务器软硬件信息、检查配置错误及其原因。

- 通过分析和监控日志，可以及时了解服务器的负荷、性能和安全性，从而及时采取措施纠正错误。

3. 直观的数据可视化与理解：

- Kibana为Elasticsearch提供Web可视化界面，可以生成各种维度表格、图形，使复杂的日志数据可视化。
- 可视化界面帮助用户更直观地理解和分析数据，进一步提升日志分析和系统监控的效果。

2. ELK的整体架构分析

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。

2.1 经典的ELK

组成：经典的ELK架构主要由Filebeat + Logstash + Elasticsearch + Kibana组成。在早期，ELK架构可能仅包含Logstash + Elasticsearch + Kibana，但随着技术的发展，Filebeat因其轻量级和高效性逐渐被引入作为日志收集工具。

特点：

- 日志收集：Filebeat作为轻量级的日志收集代理，部署在客户端上，消耗资源少，能够高效地收集日志数据。
- 数据处理：Logstash作为数据处理管道，负责将Filebeat收集的日志数据进行过滤、转换等操作，然后发送到Elasticsearch进行存储。
- 存储与搜索：Elasticsearch是一个基于Lucene的分布式搜索和分析引擎，提供强大的数据存储和搜索能力。
- 可视化：Kibana为Elasticsearch提供Web可视化界面，允许用户通过图表、仪表盘等方式直观地查看和分析日志数据。

适用场景：经典的ELK架构主要适用于数据量较小的开发环境。然而，由于缺少消息队列的缓冲机制，当Logstash或Elasticsearch出现故障时，可能存在数据丢失的风险。

2.2 整合消息队列+Nginx的ELK架构

组成：在经典的ELK架构基础上，整合消息队列（如Redis、Kafka、RabbitMQ）和Nginx，形成更为复杂的架构。

特点：

- 消息队列：引入消息队列作为缓冲机制，确保即使在Logstash或Elasticsearch出现故障时，日志数据也不会丢失。消息队列能够均衡网络传输，降低数据丢失的可能性。
- Nginx：Nginx作为高性能的Web和反向代理服务器，可以进一步优化整个系统的性能和可用性。它可以在负载均衡、缓存等方面发挥作用，提升用户访问体验。
- 扩展性：由于引入了消息队列和Nginx等组件，整个架构的扩展性得到增强。可以根据实际需求动态调整各组件的资源分配和部署规模。

适用场景：整合消息队列+Nginx的架构主要适用于生产环境，特别是需要处理大数据量的场景。它能够确保数据的安全性和完整性，同时提供高性能的日志处理和可视化分析服务。

3. 数据处理管道Logstash详解

Logstash的概述

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/>

应用场景：**ETL工具 / 数据采集处理引擎**

Logstash的工作原理分析

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，可以对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的安装与配置

Logstash安装

logstash官方文档: <https://www.elastic.co/guide/en/logstash/8.14/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: 8.14.3

```
1 #下载Logstash
2 #windows
3 https://artifacts.elastic.co/downloads/logstash/logstash-8.14.3-windows-x86_64.zip
4 #linux
5 https://artifacts.elastic.co/downloads/logstash/logstash-8.14.3-linux-x86_64.tar.gz
```

2) 测试: 运行最基本的logstash管道

```
1 cd logstash-8.14.3
2 #linux
3 #-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
4 bin/logstash -e 'input { stdin { } } output { stdout { } }'
5 #windows
6 .\bin\logstash.bat -e "input { stdin { } } output { stdout { } }"
```

测试结果:

Logstash的配置

参考: <https://www.elastic.co/guide/en/logstash/8.14/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。

```

1 input {
2   stdin { }
3 }
4
5 filter {
6   grok {
7     match => { "message" => "%{COMBINEDAPACHELOG}" }
8   }
9   date {
10    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
11  }
12 }
13
14 output {
15   elasticsearch {
16     index => "logstash-demo"
17     hosts => ["localhost:9200"]
18   }
19   stdout { codec => rubydebug }
20 }

```

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。

```

1 #运行
2 bin/logstash -f logstash-demo.conf

```

测试效果

Loginstash插件

Input Plugins

<https://www.elastic.co/guide/en/logstash/8.14/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

Filter Plugins

<https://www.elastic.co/guide/en/logstash/8.14/filter-plugins.html>

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: 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
- Ruby: 利用Ruby 代码来动态修改Event

Output Plugins

<https://www.elastic.co/guide/en/logstash/8.14/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

Codec Plugins

<https://www.elastic.co/guide/en/logstash/8.14/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

Codec Plugin测试

```
1 # single line
2 bin/logstash -e "input{stdin{codec=>line}}output{stdout{codec=> rubydebug}}"
3
```

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

```

1 # 多行数据, 异常
2 Exception in thread "main" java.lang.NullPointerException
3     at com.example.myproject.Book.getTitle(Book.java:16)
4     at com.example.myproject.Author.getBookTitles(Author.java:25)
5     at com.example.myproject.Bootstrap.main(Bootstrap.java:14)
6
7
8 #vim multiline-exception.conf
9 input {
10     stdin {
11         codec => multiline {
12             pattern => "^\\s"
13             what => "previous"
14         }
15     }
16 }
17
18 filter {}
19
20 output {
21     stdout { codec => rubydebug }
22 }
23
24 #执行管道
25 bin/logstash -f multiline-exception.conf

```

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

```

1 # pipelines.yml
2 queue.type: persisted (默认是memory)
3 queue.max_bytes: 4gb

```


实践练习：同步mysql数据到Elasticsearch

需求分析

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

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;
- 官方文档：[Jdbc input plugin](#)

拓展：[如何保证Mysql数据库到ES的数据一致性](#)

JDBC Input Plugin实现步骤

- 1) 拷贝jdbc依赖到logstash-8.14.3/drivers（自定义的）目录下
- 2) 准备mysql-demo.conf配置文件

```

1 input {
2   jdbc {
3     jdbc_driver_library => "/home/fox/logstash-8.14.3/driver/mysql-connector-java-
4       5.1.49.jar"
5     jdbc_driver_class => "com.mysql.jdbc.Driver"
6     jdbc_connection_string => "jdbc:mysql://localhost:3306/test?useSSL=false"
7     jdbc_user => "root"
8     jdbc_password => "123456"
9     #启用追踪，如果为true，则需要指定tracking_column
10    use_column_value => true
11    #指定追踪的字段，
12    tracking_column => "last_updated"
13    #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
14    tracking_column_type => "numeric"
15    #记录最后一次运行的结果
16    record_last_run => true
17    #上面运行结果的保存位置
18    last_run_metadata_path => "jdbc-position.txt"
19    statement => "SELECT * FROM user where last_updated >:sql_last_value;"
20    schedule => " * * * * * *"
21  }
22 }
23 output {
24   elasticsearch {
25     document_id => "%{id}"
26     document_type => "_doc"
27     index => "users"
28     hosts => ["http://localhost:9200"]
29     username: "elastic"
30     password: "123456"
31   }
32   stdout{
33     codec => rubydebug
34   }
35 }

```

3) 运行logstash

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

测试

```
1 #user表
2 CREATE TABLE `user` (
3   `id` int NOT NULL AUTO_INCREMENT,
4   `name` varchar(50) DEFAULT NULL,
5   `address` varchar(50) DEFAULT NULL,
6   `last_updated` bigint DEFAULT NULL,
7   `is_deleted` int DEFAULT NULL,
8   PRIMARY KEY (`id`)
9 ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 ;
10 #插入数据
11 INSERT INTO user(name,address,last_updated,is_deleted) VALUES("张三","广州天河",unix_timestamp(NOW()),0);
```

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

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

```
1 #ES中查询
2 # 创建 alias，只显示没有被标记 deleted的用户
3 POST /_aliases
4 {
5   "actions": [
6     {
7       "add": {
8         "index": "users",
9         "alias": "view_users",
10        "filter" : { "term" : { "is_deleted" : 0} }
11      }
12    }
13  ]
14 }
15
16 # 通过 Alias查询，查不到被标记成 deleted的用户
17 POST view_users/_search
18
19 POST view_users/_search
20 {
21   "query": {
22     "term": {
23       "name.keyword": {
24         "value": "张三"
25       }
26     }
27   }
28 }
```

4. 轻量级采集器FileBeat详解

FileBeat的概述

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的工作原理分析

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的安装与配置

<https://www.elastic.co/guide/en/beats/filebeat/8.14/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：8.14.3

```
1 #windows
2 https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-8.14.3-windows-x86_64.zip
3 # linux
4 curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-8.14.3-linux-x86_64.tar.gz
5 tar xzvf filebeat-8.14.3-linux-x86_64.tar.gz
```

2) 编辑配置

修改 filebeat.yml 以设置连接信息：

```
1 output.elasticsearch:
2   hosts: ["192.168.65.174:9200", "192.168.65.192:9200", "192.168.65.204:9200"]
3   username: "elastic"
4   password: "123456"
5 setup.kibana:
6   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：

```
1 # 查看可以模块列表
2 ./filebeat modules list
3
4 #启用nginx模块
5 ./filebeat modules enable nginx
6 #如果需要更改nginx日志路径,修改modules.d/nginx.yml
7 - module: nginx
8   access:
9     enabled: true
10    var.paths: ["/var/log/nginx/access.log*"]
11
12 #启用 Logstash 模块
13 ./filebeat modules enable logstash
14 #在 modules.d/logstash.yml 文件中修改设置
15 - module: logstash
16   log:
17     enabled: true
18    var.paths: ["/home/fox/logstash-8.14.3/logs/*.log"]
19
```

4) 启动 Filebeat

```
1 # setup命令加载Kibana仪表盘。 如果仪表盘已经设置，则忽略此命令。
2 ./filebeat setup
3 # 启动Filebeat
4 ./filebeat -e
```

启动成功后，在kibana中可以查看到logstash的日志

实践练习1：FileBeat采集tomcat服务器日志并发送到Logstash

Tomcat服务器运行过程中产生很多日志信息，通过filebeat采集tomcat日志并发送到Logstash

1) 配置FileBeats采集tomcat日志并将日志发送到Logstash

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

```
1 #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。
2 # 不以ip地址开头的行追加到上一行
3 filebeat.inputs:
4 - type: log
5   enabled: true
6   paths:
7     - /home/fox/apache-tomcat-9.0.93/logs/*access*.
8   multiline.pattern: '^\\d+\\.\\.\\d+\\.\\.\\d+\\.\\.\\d+ '
9   multiline.negate: true
10  multiline.match: after
11
12 output.logstash:
13   enabled: true
14   hosts: ["localhost:5044"]
15
```

- 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

```
1 ./filebeat -e -c filebeat-tomcat.yml
```

可能出现的异常:

异常1: **Exiting: error loading config file: config file ("filebeat-tomcat.yml") can only be writable by the owner but the permissions are "-rw-rw-r--" (to fix the permissions use: 'chmod go-w /home/fox/filebeat-8.14.3-linux-x86_64/filebeat-tomcat.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

```
1 chmod 644 filebeat-tomcat.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。

2) 配置Logstash接收FileBeat收集的数据并打印

```
1 vim config/logstash-tomcat.conf
2 # 配置从FileBeat接收数据
3 input {
4     beats {
5         port => 5044
6     }
7 }
8
9 output {
10     stdout {
11         codec => rubydebug
12     }
13 }
```

测试logstash配置是否正确


```
1 bin/logstash -f config/logstash-tomcat.conf --config.test_and_exit
```

启动logstash

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

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

实践练习2： 整合ELK采集与分析tomcat日志

1) Logstash输出数据到Elasticsearch

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

```
1 vim config/logstash-tomcat.conf
2 input {
3     beats {
4         port => 5044
5     }
6 }
7
8 output {
9     elasticsearch {
10         hosts => ["http://localhost:9200"]
11         index => "tomcat-logs"
12         user => "elastic"
13         password => "123456"
14     }
15     stdout{
16         codec => rubydebug
17     }
18 }
```

启动logstash

```
1 bin/logstash -f config/logstash-tomcat.conf --config.reload.automatic
```

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。

2) 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。

```
1 查看Logstash已经安装的插件
2 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/8.14/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：

```
1 %{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：

```
1 %{NUMBER:duration} %{IP:client}
2 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

```
1 filter {
2   grok {
3     match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }
4   }
5 }
```

比如，tomacat日志

```
1 192.168.65.103 - - [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

```
1 %{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

```
1 vim config/logstash-console.conf
2
3 input {
4     beats {
5         port => 5044
6     }
7 }
8
9 filter {
10    grok {
11        match => {
12            "message" => "%{IP:ip} - - \[%{HTTPDATE:date}\] \[%{WORD:method} %{PATH:uri} %
13            {DATA:protocol}\ " %{INT:status:int} %{INT:length:int}"
14        }
15    }
16
17    output {
18        stdout {
19            codec => rubydebug
20        }
21    }
```

启动logstash测试

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

mutate插件

使用mutate插件过滤掉不需要的字段

```
1 mutate {
2     enable_metric => "false"
3     remove_field => ["message", "log", "tags", "input", "agent", "host", "ecs",
4         "@version"]
5 }
```

Date插件

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/8.14/plugins-filters-date.html>

用法如下：

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。

```
1 date {
2     match => ["date", "dd/MMM/yyyy:HH:mm:ss Z", "yyyy-MM-dd HH:mm:ss"]
3     target => "date"
4 }
```

filter完整的配置

测试效果

3) 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。

```
1 output {
2   elasticsearch {
3     index => "tomcat_web_log_%{+YYYY-MM}"
4     hosts => ["http://localhost:9200"]
5     user => "elastic"
6     password => "123456"
7   }
8   stdout{
9     codec => rubydebug
10  }
11 }
```

注意：index名称中，不能出现大写字符

完整的Logstash配置文件

```
1 vim config/logstash-tomcat-es.conf
2
3 input {
4     beats {
5         port => 5044
6     }
7 }
8
9 filter {
10     grok {
11         match => {
12             "message" => "%{IP:ip} - - \[%{HTTPDATE:date}\] \"%{WORD:method} %{PATH:uri} %
13             %{DATA:protocol}\" %{INT:status:int} %{INT:length:int}"
14         }
15     }
16     mutate {
17         enable_metric => "false"
18         remove_field => ["message", "log", "tags", "input", "agent", "host", "ecs",
19             "@version"]
20     }
21     date {
22         match => ["date", "dd/MMM/yyyy:HH:mm:ss Z", "yyyy-MM-dd HH:mm:ss"]
23         target => "date"
24     }
25 }
26
27 output {
28     stdout {
29         codec => rubydebug
30     }
31     elasticsearch {
32         index => "tomcat_web_log_%{+YYYY-MM}"
33         hosts => ["http://localhost:9200"]
34         user => "elastic"
35         password => "123456"
36     }
37 }
```

启动logstash

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

查询es中是否有数据

4) 通过Kibana分析微服务日志

在kibana中，创建一个数据视图，创建完成后可以看到索引的相关详细信息

点击Discover，选择刚刚创建的数据视图

筛选出status为403的日志

5. 微服务整合ELK实现日志采集与分析实战

实现思路分析

Spring Boot应用输出日志到ELK的流程如下图所示：

实现步骤：

1. Spring Boot应用产生日志数据，使用Logback日志框架记录日志。
2. Logstash作为日志收集器，接收Spring Boot应用发送的日志数据。
3. Logstash解析和过滤日志数据，可能会对其进行格式化和处理。
4. 处理后的日志数据被发送到Elasticsearch，Elasticsearch将日志数据存储到分布式索引中。
5. Kibana连接到Elasticsearch，可以查看存储在Elasticsearch中的日志数据。

微服务整合Logstash实现日志采集

1) 使用logstash日志插件

引入依赖

```
1 <dependency>
2   <groupId>net.logstash.logback</groupId>
3   <artifactId>logstash-logback-encoder</artifactId>
4   <version>6.3</version>
5 </dependency>
```


2) logback-spring.xml中添加logstash配置

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <configuration debug="false">
3      <property name="LOG_HOME" value="logs/elk-demo.log" />
4      <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
5          <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
6              <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} -
                %msg%n</pattern>
7          </encoder>
8      </appender>
9
10     <appender name="logstash"
11         class="net.logstash.logback.appender.LogstashTcpSocketAppender">
12         <destination>192.168.65.211:4560</destination>
13         <encoder class="net.logstash.logback.encoder.LogstashEncoder" >
14             <customFields>{"appname": "elk-demo"}</customFields>
15         </encoder>
16     </appender>
17     <!-- 日志输出级别 -->
18     <root level="INFO">
19         <appender-ref ref="STDOUT" />
20         <appender-ref ref="logstash" />
21     </root>
22 </configuration>
```

3) 添加elk-demo.conf配置，启动logstash

```
1 vim config/elk-demo.conf
2
3 input {
4   tcp {
5     host => "0.0.0.0"
6     port => "4560"
7     mode => "server"
8     codec => json_lines
9   }
10  stdin {}
11 }
12 filter {
13
14 }
15 output {
16   stdout {
17     codec => rubydebug
18   }
19   elasticsearch {
20     hosts => ["127.0.0.1:9200"]
21     index => "%{[appname]}-%{+YYYY.MM.dd}"
22   }
23 }
24
```

启动logstash

```
1 # 后台启动
2 bin/logstash -f config/elk-demo.conf
```

4) 测试

调用springboot应用提供的接口，logstash控制台是否正常打印日志

在kibana中查看elk-demo开头的索引是否存在

通过Kibana分析微服务日志

创建demo-elk-*的数据视图

在Discover中查看日志数据