Logstash原理介绍及应用

logstash是一种分布式日志收集框架，开发语言是JRuby，当然是为了与Java平台对接，不过与Ruby语法兼容良好，非常简洁强大，经常与ElasticSearch，Kibana配置，组成著名的ELK技术栈，非常适合用来做日志数据的分析。

当然它可以单独出现，作为日志收集软件，你可以收集日志到多种存储系统或临时中转系统，如MySQL，redis，kakfa，HDFS, lucene，solr等，并不一定是ElasticSearch。

1、Logstash安装

Logstash安装非常简单，下载解压即可！（JRuby语言开发，需要先安装JDK）

```
cd /usr/local
wget https://artifacts.elastic.co/downloads/logstash/logstash-7.8.0.zip
unzip logstash-7.8.0.zip

#测试
cd /logstash-7.8.0/bin
./logstash -e 'input { stdin { } } output { stdout {} }'
```

```
[root@ydt1 bin]# ./logstash -e "input{stdin{}} output{stdout{}}"
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increase from one, then you should configure the
number of parallel GC threads appropriately using -XX:ParallelGCThreads=N
Sending Logstash logs to /usr/local/logstash-7.8.0/logs which is now configured via log4j2.properties
[2020-09-03T14:18:19,882][WARN ][logstash.config.source.multilocal] Ignoring the 'pipelines.yml' file because modules or comma
nd line options are specified
[2020-09-03T14:18:20,068][INFO ][logstash.runner          ] Starting Logstash {"logstash.version"=>"7.8.0", "jruby.vers
jruby 9.2.11.1 (2.5.7) 2020-03-25 b1f55b1a40 OpenJDK 64-Bit Server VM 25.232-b09 on 1.8.0_232-b09 +indy +jit [linux-x86_64]"}
[2020-09-03T14:18:22,317][INFO ][org.reflections.Reflections] Reflections took 85 ms to scan 1 urls, producing 21 keys and 41
values
[2020-09-03T14:18:23,753][INFO ][logstash.javapipeline    ][main] Starting pipeline {:pipeline_id=>"main", "pipeline.workers"=
>1, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>50, "pipeline.max_inflight"=>125, "pipeline.sources"=>["config string"
], :thread=>"#<Thread:0x1d03bc9 run>"}
[2020-09-03T14:18:25,099][INFO ][logstash.javapipeline    ][main] Pipeline started {"pipeline.id"=>"main"}
The stdin plugin is now waiting for input:
[2020-09-03T14:18:25,190][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_ru
nning_pipelines=>[]}
[2020-09-03T14:18:25,678][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
hello logstash!
/usr/local/logstash-7.8.0/vendor/bundle/jruby/2.5.0/gems/awesome_print-1.7.0/lib/awesome_print/formatters/base_formatter.rb:31
: warning: constant ::Fixnum is deprecated
{
      "@version" => "1",
    "@timestamp" => 2020-09-03T06:18:45.692Z,
          "host" => "ydt1",
       "message" => "hello logstash!"
}
```

2、Logstash原理

2.1 输入、过滤器和输出

Logstash 能够动态地采集、转换和传输数据，不受格式或复杂度的影响。利用 Grok 从非结构化数据中派生出结构，从 IP 地址解码出地理坐标，匿名化或排除敏感字段，并简化整体处理过程。

2.2 采集各种样式、大小和来源的数据

数据往往以各种各样的形式，或分散或集中地存在于很多系统中。Logstash 支持各种输入选择，可以同时从众多常用来源捕捉事件。能够以连续的流式传输方式，轻松地从您的日志、指标、Web 应用、数据存储以及各种 AWS 服务采集数据

2.3 实时解析和转换数据

数据从源传输到存储库的过程中，Logstash 过滤器能够解析各个事件，识别已命名的字段以构建结构，并将它们转换成通用格式，以便进行更强大的分析和实现商业价值。

Logstash 能够动态地转换和解析数据，不受格式或复杂度的影响：

- 利用 Grok 从非结构化数据中派生出结构
- 从 IP 地址破译出地理坐标
- 将 PII 数据匿名化，完全排除敏感字段
- 简化整体处理，不受数据源、格式或架构的影响
- 使用我们丰富的过滤器库和功能多样的 Elastic Common Schema，您可以实现无限丰富的可能。

2.4 导出数据

尽管 Elasticsearch 是我们的首选输出方向，能够为我们的搜索和分析带来无限可能，但它并非唯一选择。Logstash 提供众多输出选择，您可以将数据发送到您要指定的地方，并且能够灵活地解锁众多下游用例。

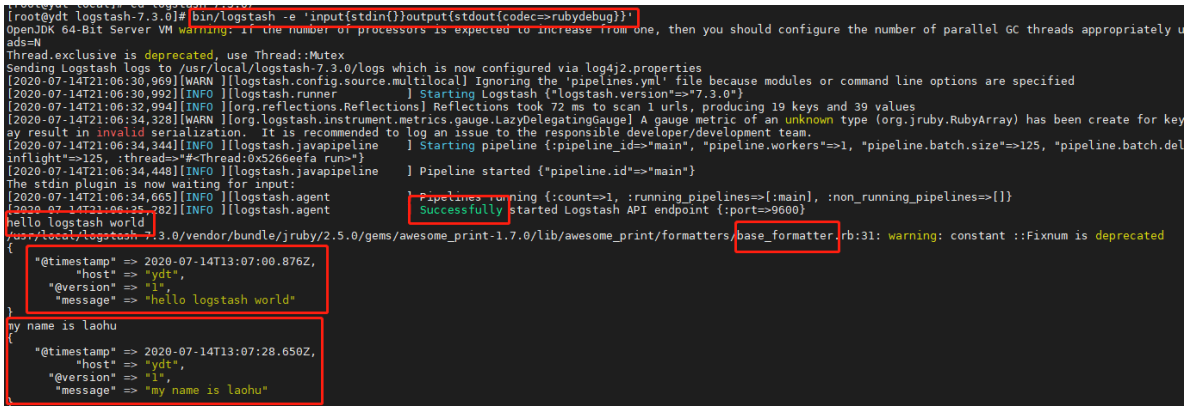
总结：logstash就是一个具备实时数据传输能力的*管道*，负责将数据信息从管道的输入端传输到管道的输出端；与此同时这根管道还可以根据需求在中间加上滤网；是一个input | filter | output 的数据流

3、 LogStash入门使用

3.1 Input插件

3.1.1 stdin标准输入和stdout标准输出

```
cd logstash-7.8.0/
bin/logstash -e 'input{stdin{}}output{stdout{codec=>rubydebug}}' #使用input插件的stdin/stdout输入输出流的形式启动
```



3.1.2 监控日志文件变化

Logstash 使用一个名叫 **FileWatch** 的 Ruby Gem 库来监听文件变化。这个库支持 glob 展开文件路径，而且会记录一个叫 **.sincedb** 的数据库文件来跟踪被监听的日志文件的当前读取位置。

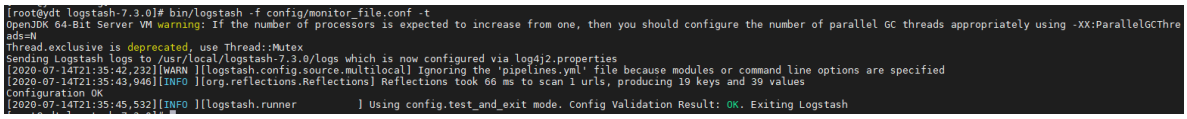
3.1.2.1 编写脚本

```
cd logstash-7.8.0/config/
vim monitor_file.conf  #编辑一个检测脚本文件，输入以下配置
-----------------------------------------------------------------------
input{
        file{
                path => "/usr/local/logstash-7.8.0/config/test.log" #随便找一个文本路径，也可以使用“*”进行模糊匹配
                type => "log" #收集日志类型
                start_position => "beginning" #从文本起始位置开始收集
```

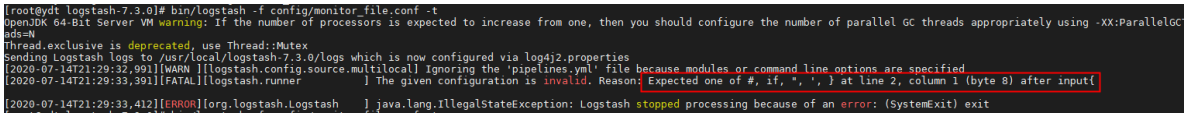
```
        }
}
output{
        stdout{
                codec=>rubydebug #解析转换类型：ruby
        }
}
```

3.1.2.2 启动服务

```
#启动服务
cd logstash-7.8.0/
bin/logstash -f config/monitor_file.conf -t #-t检测脚本是否正确,启动时不要带-t，因为需要查看日志采集情况
```



logstash 监测脚本必须严格的使用tab缩进，否则会有如下报错：

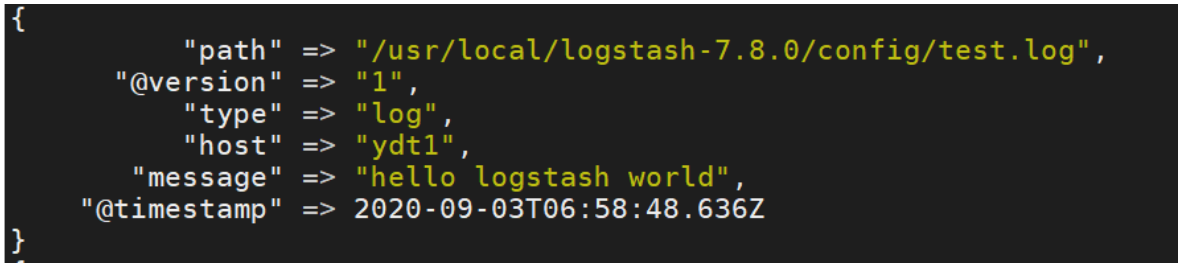


3.1.2.3 测试

另起一个窗口，向test.log文件中追加内容，观察控制台变化。

```
echo "hello logstash world" >> /usr/local/logstash-7.8.0/config/test.log
```

logstash会将文本类容采集，解析和转换输出



PS：一些配置解释

path=>表示监控的文件路径
type=>给类型打标记，用来区分不同的文件类型，**跟采集的文件类型无关。**
start_position=>从哪里开始记录文件，默认是从结尾开始标记，要是你从头导入一个文件就把改成”beginning”.
discover_interval=>多久去监听path下是否有文件，默认是15s
exclude=>排除什么文件
close_older=>一个已经监听中的文件，如果超过这个值的时间内没有更新内容，就关闭监听它的文件句柄。默认是3600秒，即一个小时。
sincedb_path=>监控库存放位置(默认的读取文件信息记录在哪个文件中)。默认在：/data/plugins/inputs/file。

sincedb_write_interval=> logstash 每隔多久写一次 sincedb 文件，默认是 15 秒。
stat_interval=>logstash 每隔多久检查一次被监听文件状态（是否有更新），默认是 1 秒。

4、Logstash高级使用

学习官网：https://www.elastic.co/

4.1 jdbc插件

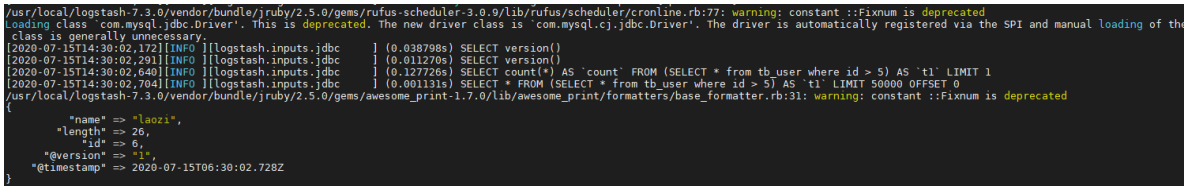
4.1.1编写脚本

```
cd logstash-7.8.0/config/
vim jdbc.conf  #编辑一个检测脚本文件，输入以下配置
----------------------------------------------------------------------
input{
        jdbc{
                jdbc_driver_library => "/usr/local/logstash-7.8.0/config/mysql-connector-java-8.0.16.jar"
                jdbc_driver_class => "com.mysql.jdbc.Driver"
                jdbc_connection_string => "jdbc:mysql://192.168.223.128/test"
                jdbc_user => "root"
                jdbc_password => "root"
                use_column_value => true
                tracking_column => id #追踪字段
                schedule => "* * * * *" #最小采集频率，logstash不支持秒级更新，最小时间单位是1分钟
                jdbc_paging_enabled => "true"
                jdbc_page_size => "50000"
                statement => "SELECT * from tb_user where id > :sql_last_value" #最新数据，可以通过删除 ./root/.logstash_jdbc_last_run 文件重新定位，查询位置命令：find /root -name *.logstash_jdbc_last_run
        }
}
output{
        stdout{
                codec=>rubydebug
        }
}
```

4.1.2测试

```
#启动服务
cd logstash-7.8.0/
bin/logstash -f config/jdbc.conf
```

在test数据库tb_user表中添加数据，可以看到控制台打印如下：



4.2 syslog插件

syslog机制负责记录内核和应用程序产生的日志信息，管理员可以通过查看日志记录，来掌握系统状况。默认系统已经安装了rsyslog直接启动即可。

4.2.1 编写脚本

```
cd logstash-7.8.0/config/
vim syslog.conf  #编辑一个检测脚本文件，输入以下配置
----------------------------------------------------------------------
input{
        syslog{
                type => "system-syslog"
                port => 514
        }
}
output{
        stdout{
                codec=> rubydebug
        }
}
```

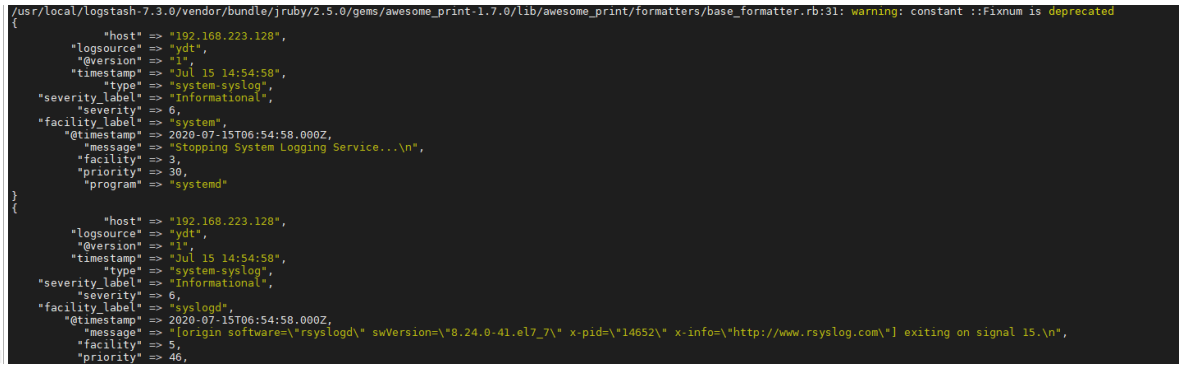
4.2.2 测试

```
#启动服务
cd logstash-7.8.0/
bin/logstash -f config/syslog.conf
```

发送数据

新起一个窗口

```
#修改系统日志配置文件
vim /etc/rsyslog.conf
#添加一行以下配置
*.* @@192.168.223.128:514
#重启系统日志服务使之生效
systemctl restart rsyslog
```



4.3 filter插件

Logstash之所以强悍的主要原因是filter插件；通过过滤器的各种组合可以得到我们想要的结构化数据。

4.3.1 grok插件

4.3.1.1 grok语法

grok正则表达式是logstash非常重要的一个环节；可以通过grok非常方便的将数据拆分和索引。

grok插件：能匹配一切数据，但是性能和对资源的损耗也很大，但是对于时间来说非常便利

语法格式：%{语法：语义}

默认grok调用的是：/logstash-7.8.0/vendor/bundle/jruby/2.5.0/gems/logstash-patterns-core-4.1.2/patterns/grok-patterns 这个目录下的正则，当然，你也可以定义自己的正则表达式！

```
USERNAME [a-zA-Z0-9._-]+
USER %{USERNAME}
EMAILLOCALPART [a-zA-Z][a-zA-Z0-9_.+-=:]+
EMAILADDRESS %{EMAILLOCALPART}@%{HOSTNAME}
INT (?:[+-]?(?:[0-9]+))
BASE10NUM (?<![0-9.+-])(?>[+-]?(?:(?:[0-9]+(?:\.[0-9]+)?)|(?:\.[0-9]+)))
NUMBER (?:%{BASE10NUM})
BASE16NUM (?<![0-9A-Fa-f])(?:[+-]?(?:0x)?(?:[0-9A-Fa-f]+))
BASE16FLOAT \b(?<![0-9A-Fa-f.])(?:[+-]?(?:0x)?(?:(?:[0-9A-Fa-f]+(?:\.[0-9A-Fa-f]*)?)|(?:\.[0-9A-Fa-f]+)))

POSINT \b(?:[1-9][0-9]*)\b
NONNEGINT \b(?:[0-9]+)\b
WORD \b\w+\b
NOTSPACE \S+
SPACE \s*
DATA .*?
GREEDYDATA .*
QUOTEDSTRING (?>(?<!\\)(?>"(?>\\.|[^\\"]+)+"|""|(?>'(?>\\.|[^\\']+)+')|''|(?>`(?>\\.|[^\\`]+)+`)|``))
UUID [A-Fa-f0-9]{8}-(?:[A-Fa-f0-9]{4}-){3}[A-Fa-f0-9]{12}
# URN, allowing use of RFC 2141 section 2.3 reserved characters
URN urn:[0-9A-Za-z][0-9A-Za-z-]{0,31}:(?:%[0-9a-fA-F]{2}|[0-9A-Za-z()+,.:=@;$_!*'/?#-])+

# Networking
MAC (?:%{CISCOMAC}|%{WINDOWSMAC}|%{COMMONMAC})
CISCOMAC (?:(?:[A-Fa-f0-9]{4}\.){2}[A-Fa-f0-9]{4})
WINDOWSMAC (?:(?:[A-Fa-f0-9]{2}-){5}[A-Fa-f0-9]{2})
COMMONMAC (?:(?:[A-Fa-f0-9]{2}:){5}[A-Fa-f0-9]{2})
IPV6 ((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){6}(:[0-9A-Fa-f]{1,4}|((25[0-5]|2[
)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]{1,4}){1,2})|:((25[0
-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]{1,4}:){4}(((:[0-9A-Fa-f]{1,4}){1,3})|(
```

4.3.1.2 收集控制台输入数据，采集IPV4

```
cd logstash-7.8.0/config/
vim filter-grok.conf  #编辑一个检测脚本文件，输入以下配置
-----------------------------------------------------------------------
input{
        stdin{

        }
}

filter{
        grok{
                match => {"message" => "%{IPV4:ip}"}
        }
}

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

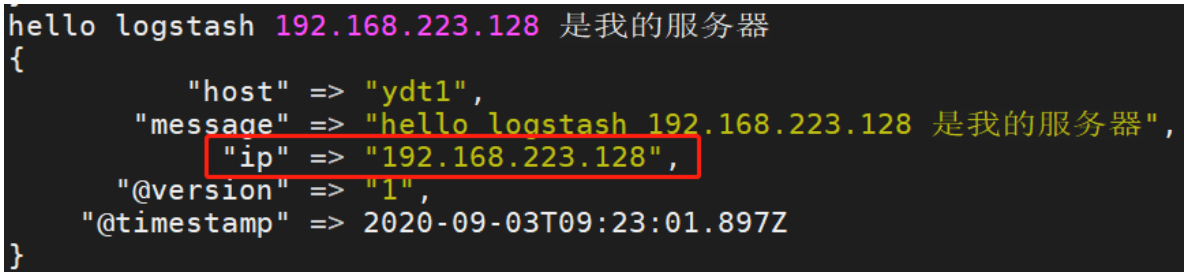
测试：

```
#启动服务
cd logstash-7.8.0/
bin/logstash -f config/filter-grok.conf
```

控制台输入文字：**我们的事业，在希望的田野上**,可以看到产生了一个tags索引：[0]

没看到是报错吗？为什么？因为不匹配呗！

我们再次输入：**hello logstash 192.168.223.128 是我的服务器**，匹配成功!



4.3.1.3 grok收集服务请求日志数据

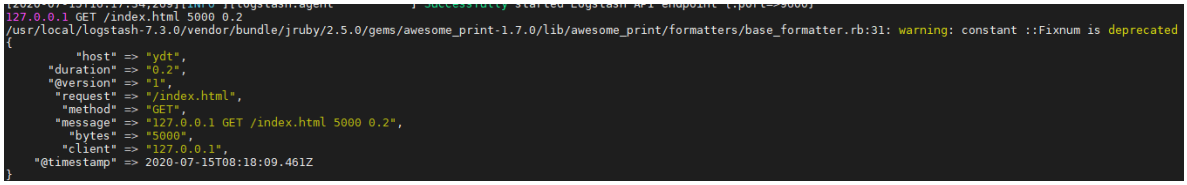
```
cd logstash-7.8.0/config/
vim monitor-server.conf  #编辑一个检测脚本文件，输入以下配置
----------------------------------------------------------------------
input{
        stdin{
        }
}

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

测试：

```
#启动服务
cd logstash-7.8.0/
bin/logstash -f config/monitor-server.conf
```

从控制台输入server日志文件数据

测试输入数据：**127.0.0.1 GET /index.html 5000 0.2**



4.3.2 Date插件

4.3.2.1 采集配置

从字段解析日期以用作事件的Logstash时间戳，以下配置解析名为`logdate`的字段以设置Logstash时间戳，获取的是日志时间，而不是系统时间:@timestamp

```
cd logstash-7.3.0/config/
vim filter-date.conf  #编辑一个检测脚本文件，输入以下配置
---------------------------------------------------------------------
input{
        stdin{

        }
}

filter{
        grok{
                match => {"message" => "%{HTTPDATE:timestamp}"}
        }
        date{
                match => ["timestamp","dd/MMM/yyyy:HH:mm:ss Z"] #时区偏移量需要用一个字母Z来转换
        }
}

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

4.3.2.2 匹配字段格式化

4.3.3 geoip地址查询插件

4.3.3.1 采集配置

geoip是常见的免费的IP地址归类查询库，geoip可以根据IP地址提供对应的地域信息，包括国别，省市，经纬度等等，此插件对于可视化地图和区域统计非常有用。

```
cd logstash-7.3.0/config/
vim filter-geoip.conf  #编辑一个检测脚本文件，输入以下配置
```

```
# ------------------------------------------------------------------------
input{
        stdin{
        }
}

filter{
        grok {
                match => {
                        "message" => "%{IP:ip}"
                }
                remove_field => ["message"]
        }
        geoip {
                source => "ip"
        }
}

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

4.3.3.2 测试

```
#启动服务
cd logstash-7.8.0/
bin/logstash -f config/filter-geoip.conf
```

```
ip地址为220.202.225.63
/usr/local/logstash-7.8.0/vendor/bundle/jruby/2.5.0/gems/awesome_print-1.7.0/lib/awesome_print/formatters/
: warning: constant ::Fixnum is deprecated
{
    "@timestamp" => 2020-09-03T10:08:28.795Z,
          "host" => "ydt1",
      "@version" => "1",
            "ip" => "220.202.225.63",
         "geoip" => {
             "city_name" => "Shenzhen",
             "longitude" => 114.1333,
              "location" => {
            "lon" => 114.1333,
            "lat" => 22.5333
        },
          "country_name" => "China",
         "country_code3" => "CN",
        "continent_code" => "AS",
           "region_name" => "Guangdong",
                    "ip" => "220.202.225.63",
              "latitude" => 22.5333,
           "region_code" => "GD",
              "timezone" => "Asia/Shanghai",
         "country_code2" => "CN"
    }
}
```

4.3.4 mutate插件

mutate插件是logstash另一个非常重要的插件，它提供了丰富的基础类型数据处理能力，包括重命名、删除、替换、修改日志事件中的字段。我们这里使用一个常用的mutate插件：正则表达式替换字段功能gsub

PS:gsub可以通过正则表达式替换字段中匹配到的值，但是这本身只对字符串字段有效。

4.3.4.1 采集配置

```
cd logstash-7.3.0/config/
vim filter-mutate.conf  #编辑一个检测脚本文件，输入以下配置
---------------------------------------------------------------------
input{
        stdin{
        }
}

filter{
        grok{
                match => {"message" => "%{QS:referrer}"}
                remove_field => ["message"]
        }
        mutate{
                gsub => ["referrer","/","-"]
        }
}

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

4.3.4.2 测试

```
#启动服务
cd logstash-7.8.0/
bin/logstash -f config/filter-mutate.conf
```

```
[07/Feb/2018:16:24:9 +0800] "GET /HTTP/1.1" 403 5039
{
      "referrer" => "\"GET -HTTP-1.1\"",
    "@timestamp" => 2020-09-03T10:19:04.417Z,
          "host" => "ydt1",
      "@version" => "1"
}
```

4.4 Output插件

刚刚上面演示的全部都是标准的控制台输出，其实logstash还有很多更高级的应用输出

4.4.1 采集数据保存到file文件中

```
cd logstash-7.3.0/config/
vim output_file.conf  #编辑一个检测脚本文件，输入以下配置
---------------------------------------------------------------------
input{
        stdin{
        }
}

output{
        file{
```

```
                path => "/usr/local/logstash-7.8.0/config/datas/%{+YYYY-MM-dd}-%
{host}.txt"
                codec => line {
                        format => "%{message}"
                }
                flush_interval => 0
        }
}
```

测试：

```
#启动服务
cd logstash-7.8.0/
bin/logstash -f config/output_file.conf
```

控制台输入一些字符串：

```
老王是一个非常热心的人，经常跑到邻居家帮忙修水管，一般都会 男主人不在家的时候
[2020-07-15T16:35:00,489][INFO ][logstash.outputs.file     ] Opening file {:path=>"/usr/local/logstash-7.3.0/config/datas/2020-07-15-ydt.txt"}
^C[2020-07-15T16:35:08,671][WARN ][logstash.runner          ] SIGINT received. Shutting down.
[2020-07-15T16:35:08,906][INFO ][logstash.javapipeline    ] Pipeline terminated {"pipeline.id"=>"main"}
[2020-07-15T16:35:08,987][INFO ][logstash.runner          ] Logstash shut down.
[root@ydt logstash-7.3.0]# vim config/datas/2020-07-15-ydt.txt
```

查看输出的文件内容：

```
more 2018-11-08-node01.hadoop.com.txt
老王是一个非常热心的人，经常跑到邻居帮忙修水管，一般男主人不在家的时候
~
```

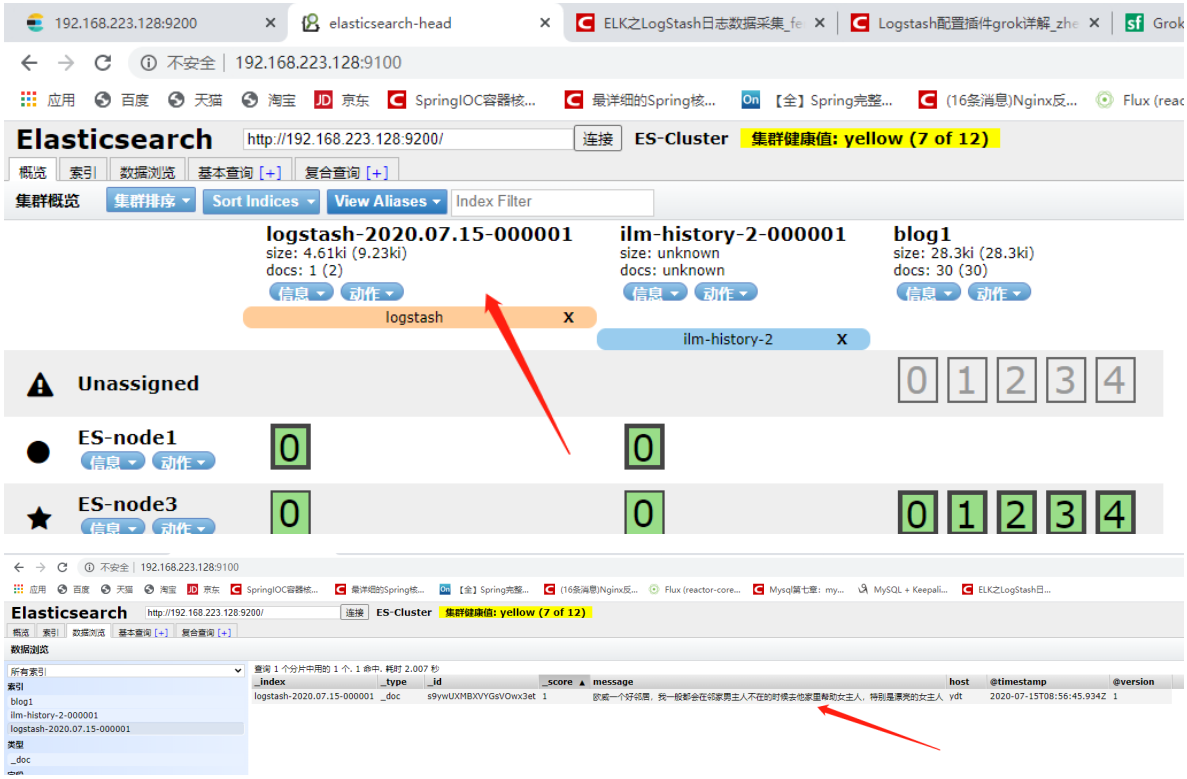
4.4.2 采集数据保存到elasticsearch

```
cd logstash-7.8.0/config/
vim output_es.conf  #编辑一个检测脚本文件，输入以下配置
-----------------------------------------------------------------
input {stdin{}}
output {
    elasticsearch {
        hosts => ["node01:9200"]
        index => "logstash-%{+YYYY.MM.dd}"
    }
}
```

测试：

```
#启动服务
cd logstash-7.8.0/
bin/logstash -f config/output_es.conf
```

控制台输入一些数据，然后通过elasticsearch-head查看是否保存成功：


192.168.223.128:9200
elasticsearch-head
ELK之LogStash日志数据采集_fe
Logstash配置插件grok详解_zhe
Grok
不安全 | 192.168.223.128:9100
应用
百度
天猫
淘宝
京东
SpringIOC容器核...
最详细的Spring核...
【全】Spring完整...
(16条消息)Nginx反...
Flux (reac
Elasticsearch
http://192.168.223.128:9200/
连接
ES-Cluster
集群健康值: yellow (7 of 12)
概览
索引
数据浏览
基本查询 [+]
复合查询 [+]
集群概览
集群排序
Sort Indices
View Aliases
Index Filter
logstash-2020.07.15-000001
size: 4.61ki (9.23ki)
docs: 1 (2)
信息
动作
logstash
X
ilm-history-2-000001
size: unknown
docs: unknown
ilm-history-2
blog1
size: 28.3ki (28.3ki)
docs: 30 (30)
Unassigned
0 1 2 3 4
ES-node1
ES-node3

数据浏览
所有索引
索引
blog1
ilm-history-2-000001
logstash-2020.07.15-000001
类型
_doc
查询 1 个分片中用的 1 个，1 命中，耗时 2.007 秒
_index
_type
_id
_score
message
host
@timestamp
@version
logstash-2020.07.15-000001
_doc
s9ywUXMBXVYGsVOwx3et
1
欧威一个好邻居，我一般都会在邻家男主人不在的时候去他家里帮助女主人，特别是漂亮的女主人
ydt
2020-07-15T08:56:45.934Z
1