HZhuizai

博客园 首页 新随笔 联系 订阅 管理

从0到1学会logstash的玩法（ELK）

目录

本篇文章采用的采用的是logstash-7.7.0版本，主要从如下几个方面介绍

1、logstash是什么，可以用来干啥

2、logstash的基本原理是什么

3、怎么去玩这个elk的组件logstash

回到顶部

一、logstash是什么，有哪些作用

1.1、概念

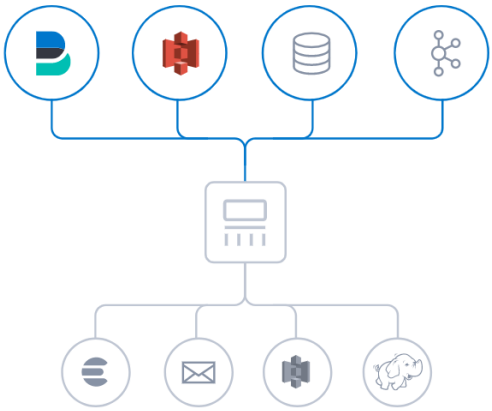
官方概念：Logstash是免费且开放的服务器端数据处理管道，能够从多个来源采集数据，转换数据，然后将数据发送到您最喜欢的"存储库"中。

1.2、功能

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。

采集数据：

能够采集各种样式、大小和来源的数据往往以各种各样的形式，比如log日志，收集redis、kafka等热门分布式技术的数据，并且还可以收集实现了java的JMS规范的消息中心的数据，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，轻松地从日志、指标、Web应用、数据存储以及各种AWS服务采集数据

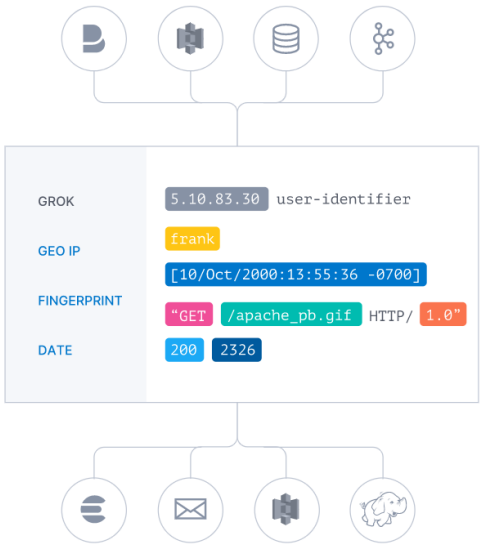


解析数据和转换：

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：

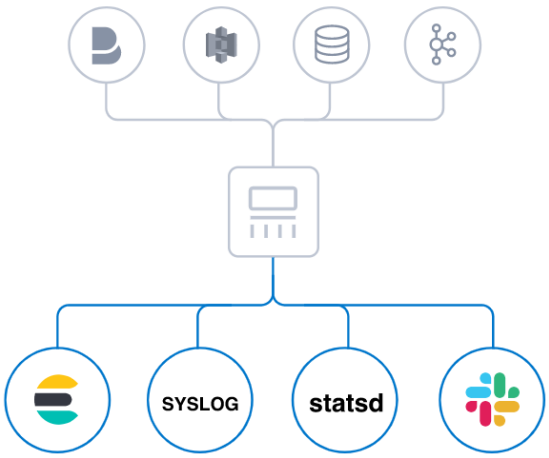
- 利用Grok从非结构化数据中解析出结构数据
- 从IP地址破译出地理坐标
- 将PII数据匿名化，完全排除敏感字段
- 简化整体处理，不受数据源、格式或架构的影响

数据输入端从各种数据源收集到的数据可能会有很多不是我们想要的，这时我们可以给Logstash定义过滤器，过滤器可以定义多个，它们依次执行，最终把我们想要的数据过滤出来，然后把这些数据解析成目标数据库，如elasticsearch等能支持的数据格式存储数据。



数据转存：

选择好存储库，导出数据到存储库进行存储，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，可以将数据发送到您要指定的地方，比如redis、kafka等



1. Re:java多线程与并发（基础篇）
学习了..

--小支

2. Re:java多线程与并发（基础篇）
并发和多线程时等价的吗？

--Parrallel

3. Re:想写一篇jvm的工具入门
good job.

--拥剑公子

4. Re:java之面向对象详解
讲的太好了，我拿走了 ，下次复习用 嘿嘿

--最终、尽头

5. Re:论logstash的玩法（ELK）
看不懂

--烟花散尽13141

阅读排行榜

1. linux后台运行的几种方式(52633)
2. java多线程与并发（基础篇）(49161)
3. java之面向对象详解(17079)
4. hadoop之hdfs命令详解(6479)
5. 一篇文章搞懂filebeat（ELK）(4573)

评论排行榜

1. 解决！！-- krb5-libs.x86_64被卸载，yum不能使用，ssh不能连接(5)
2. 2019年年终总结(2)
3. java多线程与并发（基础篇）(2)
4. 想写一篇jvm的工具入门(1)
5. 从0到1学会logstash的玩法（ELK）(1)

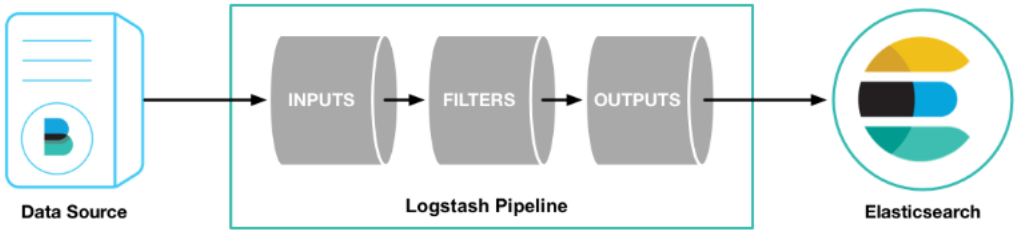
推荐排行榜

1. java多线程与并发（基础篇）(11)
2. 想写一篇jvm的工具入门(6)
3. elasticsearch跨集群数据迁移(5)
4. 一篇文章搞懂filebeat（ELK）(3)
5. 2019年年终总结(2)

回到顶部

二、logstash的基本原理

logstash分为三个步骤：inputs（必须的）→ filters（可选的）→ outputs（必须的），inputs生成时间，filters对其事件进行过滤和处理，outputs输出到输出端或者决定其存储在哪些组件里。inputs和outputs支持编码和解码。



执行模型：

Logstash是协调inputs、filters和outputs执行事件处理的管道。

Logstash管道中的每个input阶段都在自己的线程中运行。将写事件输入到内存（默认）或磁盘上的中心队列。每个管道工作线程从该队列中取出一批事件，通过配置的filter处理该批事件，然后通过output输出到指定的组件存储。管道处理数据量的大小和管道工作线程的数量是可配置的

默认情况下，Logstash使用管道阶段（input→filter和filter→output）之间的内存限制队列来缓冲事件。如果Logstash不安全地终止，存储在内存中的所有事件都将丢失。为了帮助防止数据丢失，可以启用Logstash将飞行中的事件持久化到磁盘。有关详细信息，请参阅持久队列 https://www.elastic.co/guide/en/logstash/current/persistent-queues.html

如下是目前logstash7.7.0支持的inputs、outputs、filters

inputs：

```
azure_event_hubs,beats,cloudwatch,couchdb_changes,dead_letter_queue,elasticsearch,exec,file,ganglia,gelf,generator,gi
```

outputs：

```
boundary, circonus, cloudwatch, csv, datadog, datadog_metrics, elastic_app_search, elasticsearch, email, exec, file,
```

filters：

```
aggregate, alter, bytes, cidr, cipher, clone, csv, date, de_dot, dissect, dns, drop, elapsed, elasticsearch, environm
```

回到顶部

三、玩一玩logstash

3.1、压缩包方式安装

下载地址1：https://www.elastic.co/cn/downloads/logstash

下载地址2：https://elasticsearch.cn/download/

这里需要安装jdk，我使用的是elasticsearch7.7.0自带的jdk：

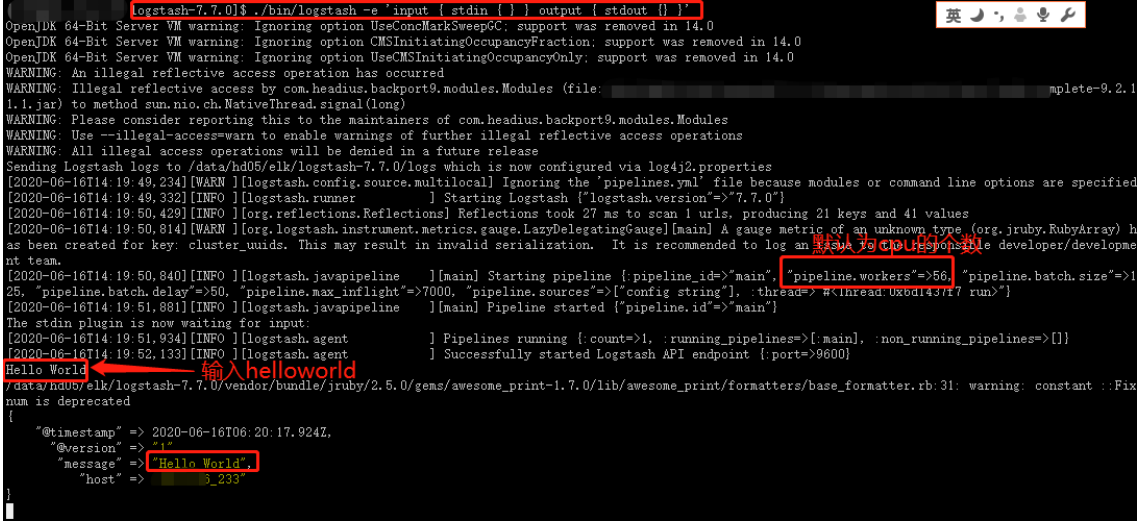
```
[elk@gxt_126_233 ~]$ java -version
openjdk version "14" 2020-03-17
OpenJDK Runtime Environment AdoptOpenJDK (build 14+36)
OpenJDK 64-Bit Server VM AdoptOpenJDK (build 14+36, mixed mode, sharing)
[elk@gxt_126_233 ~]$
```

解压即安装：

```
tar -zxvf logstash-7.7.0.tar.gz
```

来个logstash版本的HelloWorld：

```
./bin/logstash -e 'input { stdin { } } output { stdout {} }'
```



3.2、logstash配置文件

logstash.yml：包含Logstash配置标志。您可以在此文件中设置标志，而不是在命令行中传递标志。在命令行上设置的任何标志都会覆盖logstash中的相应设置

pipelines.yml：包含在单个Logstash实例中运行多个管道的框架和指令。

jvm.options：包含JVM配置标志。使用此文件设置总堆空间的初始值和最大值。您还可以使用此文件为Logsta设置语言环境

log4j2.properties：包含log4j 2库的默认设置

start.options (Linux)：用于配置启动服务脚本

logstash.yml文件详解：

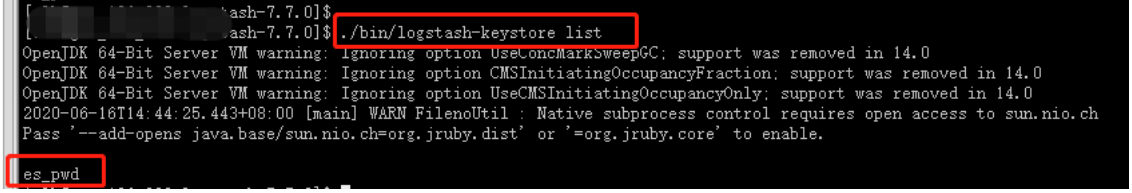
```
node.name  #默认主机名，该节点的描述名字
path.data  #LOGSTASH_HOME/data ，Logstash及其插件用于任何持久需求的目录
pipeline.id #默认main，pipeline的id
pipeline.java_execution #默认true，使用java执行引擎
pipeline.workers #默认为主机cpu的个数，表示并行执行管道的过滤和输出阶段的worker的数量
pipeline.batch.size #默认125 表示单个工作线程在尝试执行过滤器和输出之前从输入中收集的最大事件数
pipeline.batch.delay #默认50 在创建管道事件时，在将一个小批分派给管道工作者之前，每个事件需要等待多长时间(毫秒)
pipeline.unsafe_shutdown  #默认false，当设置为true时，即使内存中仍有运行的事件，强制Logstash在关闭期间将会退出。默认情况下，Logsta
pipeline.ordered #默认auto，设置管道事件顺序。true将强制对管道进行排序，如果有多个worker，则阻止logstash启动。如果为false，将禁用维
path.config #默认LOGSTASH_HOME/config  管道的Logstash配置的路径
config.test_and_exit #默认false，设置为true时，检查配置是否有效，然后退出。请注意，使用此设置不会检查grok模式的正确性
config.reload.automatic #默认false，当设置为true时，定期检查配置是否已更改，并在更改时重新加载配置。这也可以通过SIGHUP信号手动触发
config.reload.interval  #默认3s ，检查配置文件频率
config.debug #默认false 当设置为true时，将完全编译的配置显示为调试日志消息
queue.type #默认memory ，用于事件缓冲的内部排队模型。为基于内存中的遗留队列指定内存，或为基于磁盘的脱机队列(持久队列)指定持久内存
path.queue #默认path.data/queue  ，在启用持久队列时存储数据文件的目录路径
queue.page_capacity #默认64mb ，启用持久队列时(队列)，使用的页面数据文件的大小。队列数据由分隔为页面的仅追加数据文件组成
queue.max_events #默认0，表示无限。启用持久队列时，队列中未读事件的最大数量
queue.max_bytes  #默认1024mb，队列的总容量，以字节为单位。确保磁盘驱动器的容量大于这里指定的值
queue.checkpoint.acks #默认1024，当启用持久队列(队列)时，在强制执行检查点之前被隔离的事件的最大数量
queue.checkpoint.writes #默认1024，当启用持久队列(队列)时，强制执行检查点之前的最大写入事件数
queue.checkpoint.retry #默认false，启用后，对于任何失败的检查点写，Logstash将对每个尝试的检查点写重试一次。任何后续错误都不会重试。并
queue.drain #默认false，启用后，Logstash将等待，直到持久队列耗尽，然后关闭
path.dead_letter_queue#默认path.data/dead_letter_queue，存储dead-letter队列的目录
http.host #默认"127.0.0.1" 表示endpoint REST端点的绑定地址。
http.port #默认9600 表示endpoint REST端点的绑定端口。
log.level #默认info，日志级别fatal，error，warn，info，debug，trace，
log.format #默认plain 日志格式
path.logs  #默认LOGSTASH_HOME/logs 日志目录
```

3.3、keystore

keystore可以保护一些敏感的信息，使用变量的方式替代，比如使用ES_PWD代替elasticsearch的密码，可以通过${ES_PWD}来获取elasticsearch的密码，这样就是的密码不再是明文密码。

```
./bin/logstash-keystore create   #创建一个keyword
 ./bin/logstash-keystore add ES_PWD #创建一个elastic的passwd，然后通过${ES_PWD}使用该密码
 ./bin/logstash-keystore list  #查看已经设置好的键值对
 ./bin/logstash-keystore remove ES_PWD #删除在keyword中的key
```

例如：



3.4、logstash命令

注意：参数和logstash.yml配置文件对应（这里不详解，请查看3.2节）

```
-n, --node.name NAME
-f, --path.config CONFIG_PATH
-e, --config.string CONFIG_STRING
--field-reference-parser MODE
--modules MODULES
-M, --modules.variable MODULES_VARIABLE
--setup
--cloud.id CLOUD_ID
--cloud.auth CLOUD_AUTH
--pipeline.id ID
-w, --pipeline.workers COUNT
--pipeline.ordered ORDERED
--java-execution
--plugin-classloaders
-b, --pipeline.batch.size SIZE
-u, --pipeline.batch.delay DELAY_IN_MS
--pipeline.unsafe_shutdown
--path.data PATH
-p, --path.plugins PATH
-l, --path.logs PATH
--log.level LEVEL
--config.debug
-i, --interactive SHELL
-V, --version
-t, --config.test_and_exit
-r, --config.reload.automatic
--config.reload.interval
--http.host HTTP_HOST
--http.port HTTP_PORT
--log.format FORMAT
--path.settings SETTINGS_DIR
```

3.5、logstash配置文件的格式和value type

1、logstash配置文件的格式如下：

输入，解析过滤，输出，其中filter不是必须的，其他两个是必须的。

```
input {
  ...
}

filter {
  ...
}

output {
  ...
}
```

2、value types（logstash支持的数据类型）

array：数组可以是单个或者多个字符串值。

```
users => [ {id => 1, name => bob}, {id => 2, name => jane} ]
```

Lists：集合

```
path => [ "/var/log/messages", "/var/log/*.log" ]
uris => [ "http://elastic.co", "http://example.net" ]
```

Boolean：true 或者false

```
ssl_enable => true
```

Bytes：字节类型

```
my_bytes => "1113"   # 1113 bytes
my_bytes => "10MiB"  # 10485760 bytes
my_bytes => "100kib" # 102400 bytes
my_bytes => "180 mb" # 180000000 bytes
```

Codec：编码类型

```
codec => "json"
```

Hash：哈希（散列）

```
match => {
  "field1" => "value1"
  "field2" => "value2"
  ...
}
# or as a single line. No commas between entries:
match => { "field1" => "value1" "field2" => "value2" }
```

Number：数字类型

```
port => 33
```

Password：密码类型

```
my_password => "password"
```

URI：uri类型

```
my_uri => "http://foo:bar@example.net"
```

Path： 路径类型

```
my_path => "/tmp/logstash"
```

String：字符串类型，字符串必须是单个字符序列。注意，字符串值被括在双引号或单引号中

3.6、logstash的权限配置

配置账号，一个种是role，一种是user，配置方式有两种，一种是通过elasticsearch的API配置，一种是通过kibana配置：

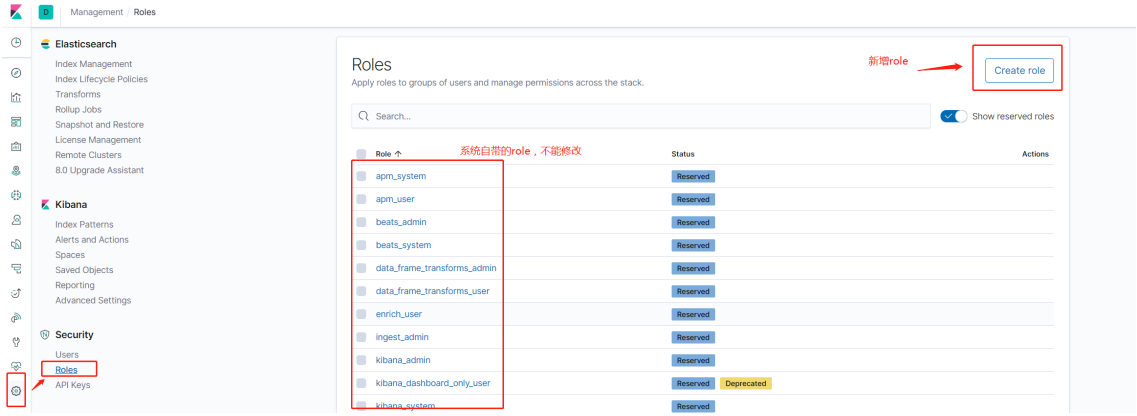
第一种方式：通过elasticsearch的API配置：

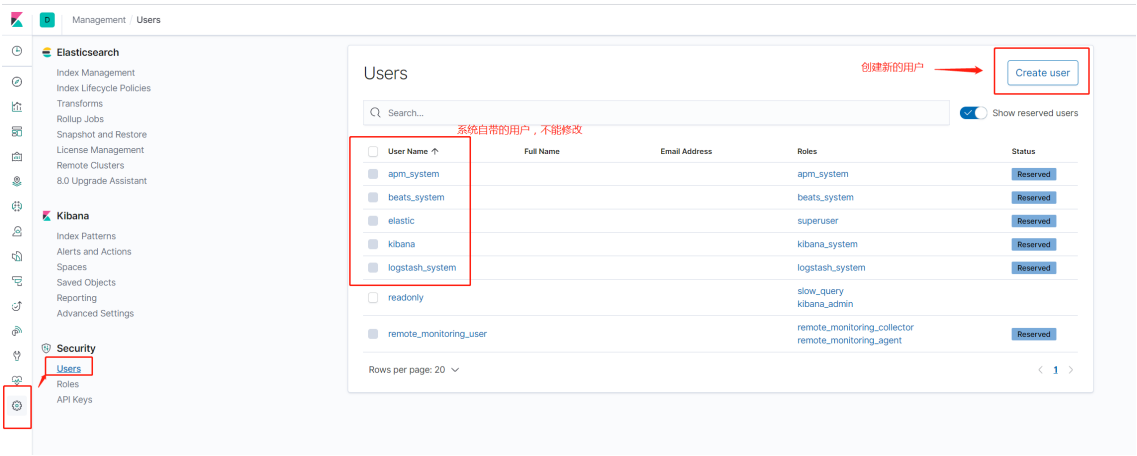
```
#添加一个logstash_writer的角色
POST _xpack/security/role/logstash_writer
{
  "cluster": ["manage_index_templates", "monitor", "manage_ilm"],
  "indices": [
    {
      "names": [ "logstash-*" ],  #索引的模式匹配
      "privileges": ["write","create","delete","create_index","manage","manage_ilm"]   #权限内容
    }
  ]
}

#添加一个有logstash_writer角色权限的用户：logstash_internal
POST _xpack/security/user/logstash_internal
{
  "password" : "x-pack-test-password",
  "roles" : [ "logstash_writer"],  #分配角色
  "full_name" : "Internal Logstash User"
}

#添加一个logstash_reader角色，只有read权限
POST _xpack/security/role/logstash_reader
{
  "indices": [
    {
      "names": [ "logstash-*" ],
      "privileges": ["read","view_index_metadata"]
    }
  ]
}

#添加一个有logstash_reader角色权限的用户：logstash_user
POST _xpack/security/user/logstash_user
```

```
{
  "password" : "x-pack-test-password",
  "roles" : [ "logstash_reader", "logstash_admin"],
  "full_name" : "Kibana User for Logstash"
}
```

第二种：通过kibana的界面配置

Management > Roles
Management > Users





权限选择见elasticsearch官网：

https://www.elastic.co/guide/en/elasticsearch/reference/current/authorization.html

https://www.elastic.co/guide/en/elasticsearch/reference/current/security-privileges.html

3.7、多管道配置

如果需要在同一个进程中运行多个管道，Logstash提供了一种通过名为pipelines.yml的配置文件来实现此目的的方法。

例如：

```
- pipeline.id: my-pipeline_1
  path.config: "/etc/path/to/p1.config"
  pipeline.workers: 3
- pipeline.id: my-other-pipeline
  path.config: "/etc/different/path/p2.cfg"
  queue.type: persisted
```

该文件在YAML文件格式中，并包含一个字典列表，其中每个字典描述一个管道，每个键/值对指定该管道的设置。该示例展示了通过id和配置路径描述的两个不同管道。对于第一个管道，为pipeline.workers的值设置为3，而在另一个中，持久队列特性被启用。未在pipelines.yml显式设置的值。yml文件将使用到logstash中指定的默认值。

当启动Logstash不带参数时，它将读取管道pipelines.yml。yml文件并实例化文件中指定的所有管道。另一方面，当使用-e或-f时，Logstash会忽略管道。

注意：

- 如果当前配置的事件流不共享相同的输入/过滤器和输出，并且使用标记和条件将它们彼此分离，这显得多个管道尤其重要
- 在单个实例中拥有多个管道还允许这些事件流具有不同的性能和持久性参数(例如，pipeline.workers和persistent queues的不同设置)。这种分离意味着一条管道中的阻塞输出不会对另一条管道产生反压力。
- 考虑管道之间的资源竞争是很重要的，因为默认值是针对单个管道调优的。因此，例如，考虑减少每个管道使用pipeline.worker的数量，因为每个管道在默认情况下每个CPU核使用一个worker。
- 每个管道都隔离持久队列和死信队列，它们的位置命名空间为pipeline.id的值

各管道之间的通信原理：https://www.elastic.co/guide/en/logstash/current/pipeline-to-pipeline.html，有兴趣的可以了解下。

3.8、配置的重新加载

在我们运行logstash的过程，不想停掉logstash进程，但是又想修改配置，就可以使用到配置的重新加载了，有两种方式。

第一种是在启动的时候指定参数：

```
bin/logstash -f apache.config --config.reload.automatic
```

Logstash每3秒检查一次配置更改。要更改此间隔，请使用--config.reload.interval <interval>选项，其中interval指定Logstash检查配置文件更改的频率（以秒为单位）,请注意，必须使用单位限定符（s）

第二种就是强制加载配置文件

```
kill -SIGHUP pid  #pid为logstash的pid
```

自动配置重新加载配置注意点：

- 当Logstash检测到配置文件中的更改时，它将通过停止所有输入来停止当前管道，并尝试创建使用更新后的配置的新管道。验证新配置的语法后，Logstash验证所有输入和输出都可以初始化（例如，所有必需的端口都已打开）。如果检查成功，则Logstash会将现有管道与新管道交换。如果检查失败，旧管道将继续运行，并且错误将传播到控制台。
- 在自动重新加载配置期间，不会重新启动JVM。管道的创建和交换都在同一过程中进行。
- 对grok模式文件的更改也将重新加载，但仅在配置文件中的更改触发重新加载（或重新启动管道）时。

3.9、常用filter介绍

1、grok

注：grok插件是一个十分耗费资源的插件

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

Grok是将非结构化日志数据解析为结构化和可查询内容的好方法，非常适合syslog日志，apache和其他Web服务器日志，mysql日志等等

首先官方提供了120中的匹配模式（但是我一直都没打开这个网址）：https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns

还有一个用来验证自定义的解析是否正确的一个网址：http://grokdebug.herokuapp.com/

既然我没能打开官方提供的120个的模式匹配，从一片博客中找到了一部分的匹配模式（如下）：
https://blog.csdn.net/cui929434/article/details/94390617

⊞ grok内置的一些匹配模式

基础语法，**一种是自带的模式，一种是自定义的模式：**

自带的模式语法：%{SYNTAX:SEMANTIC}

SYNTAX是将匹配文本模式的名称，grok自带的那些匹配模式名
SEMANTIC是你给一段文字的标识相匹配该匹配模式匹配到的内容，相当于一个字段名

例如：

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

比如要解析如下的日志：

```
55.3.244.1 GET /index.html 15824 0.043
```

就可以使用该匹配模式去匹配：

```
%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}
```

得出的结果如下：

```
client: 55.3.244.1
method: GET
request: /index.html
bytes: 15824
duration: 0.043
```

自定义的模式语法：(?<field_name>the pattern here)

例如：

```
(?<queue_id>[0-9A-F]{10,11})  #表示10-11个字符的16进制
```

我们可以创建一个目录pattters，把我自定义的模式添加进去，在使用的时候就可以使用grok自带的匹配模式的语法，例如：

```
我们在./patterns/postfix文件中添加如下内容
POSTFIX_QUEUEID [0-9A-F]{10,11}
```

如上我们就定义了一个匹配模式了，可以使用如下的方式使用。假如我们有如下的日志格式：

```
Jan  1 06:25:43 mailserver14 postfix/cleanup[21403]: BEF25A72965: message-id=<20130101142543.5828399CCAF@mailserver14
```

然后我们对其就行解析：

```
filter {
      grok {
        patterns_dir => ["./patterns"]  #指定自定义的匹配模式路径
        match => { "message" => "%{SYSLOGBASE} %{POSTFIX_QUEUEID:queue_id}: %{GREEDYDATA:syslog_message}" }
      }
    }
```

解析出的结果如下：

```
timestamp: Jan  1 06:25:43
logsource: mailserver14
program: postfix/cleanup
pid: 21403
queue_id: BEF25A72965
syslog_message: message-id=<20130101142543.5828399CCAF@mailserver14.example.com>
```

grok的配置选项：

```
break_on_match
值类型是布尔值
默认是true
描述：match可以一次设定多组，预设会依照顺序设定处理，如果日志满足设定条件，则会终止向下处理。但有的时候我们会希望让Logstash跑完所有的设定，

keep_empty_captures
值类型是布尔值
默认值是 false
描述：如果为true，捕获失败的字段将设置为空值

match
值类型是数组
默认值是 {}
描述：字段值的模式匹配
例如：
filter {
  grok { match => { "message" => "Duration: %{NUMBER:duration}" } }
}
#如果你需要针对单个字段匹配多个模式，则该值可以是一组，例如：
filter {
  grok { match => { "message" => [ "Duration: %{NUMBER:duration}", "Speed: %{NUMBER:speed}" ] } }
}

named_captures_only
值类型是布尔值
默认值是 true
描述：如果设置为true，则仅存储来自grok的命名捕获

overwrite
值类型是 array
默认是[]
描述：覆盖已经存在的字段内容
例如：

filter {
  grok {
    match => { "message" => "%{SYSLOGBASE} %{DATA:message}" }
    overwrite => [ "message" ]
  }
}
如果日志是May 29 16:37:11 sadness logger: hello world经过match属性match => { “message” => “%{SYSLOGBASE} %{DATA:message}

pattern_definitions
值类型是 数组
默认值是 {}
描述：模式名称和模式正则表达式，也是用于定义当前过滤器要使用的自定义模式。匹配现有名称的模式将覆盖预先存在的定义。可以将此视为仅适用于grok定
例如：
filter {
    grok {
        patterns_dir => "/usr/local/elk/logstash/patterns"
        pattern_definitions => {"MYSELFTIMESTAMP" => "20%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{HOUR}:?%{MINUTE}(?::?%{SECO
        match => {"message" => ["%{MYSELFTIMESTAMP:timestamp} %{JAVACLASS:message}","%{MYSELF:content}"]}
    }
}
```

patterns_dir
值类型是数组
默认值是 []
描述：一些复杂的正则表达式，不适合直接写到filter中，可以指定一个文件夹，用来专门保存正则表达式的文件，需要注意的是该文件夹中的所有文件中的正

patterns_dir => ["/opt/logstash/patterns", "/opt/logstash/extra_patterns"]
正则文件以文本格式描述：

patterns_file_glob
属性值的类型：string
默认值："*"
描述：针对patterns_dir属性中指定的文件夹里哪些正则文件，可以在这个filter中生效，需要本属性来指定。默认值"*"是指所有正则文件都生效。

tag_on_failure
值类型是数组
默认值是 ["_grokparsefailure"]
描述：没有成功匹配时，将值附加到字段到tags

tag_on_timeout
值类型是字符串
默认值是 "_groktimeout"
描述：如果Grok正则表达式超时，则应用标记。

timeout_millis
值类型是数字
默认值是 30000
描述： 尝试在这段时间后终止正则表达式。如果应用了多个模式，则这适用于每个模式。这将永远不会提前超时，但超时可能需要一些时间。实际的超时时间是

各组件的公共配置选项：

break_on_match
值类型是布尔值
默认是true
描述：match可以一次设定多组，预设会依照顺序设定处理，如果日志满足设定条件，则会终止向下处理。但有的时候我们会希望让Logstash跑完所有的设定，

keep_empty_captures
值类型是布尔值
默认值是 false
描述：如果为true，捕获失败的字段将设置为空值

match
值类型是数组
默认值是 {}
描述：字段值的模式匹配
例如：

```
filter {
  grok { match => { "message" => "Duration: %{NUMBER:duration}" } }
}

#如果你需要针对单个字段匹配多个模式，则该值可以是一组，例如：
filter {
  grok { match => { "message" => [ "Duration: %{NUMBER:duration}", "Speed: %{NUMBER:speed}" ] } }
}
```

named_captures_only
值类型是布尔值
默认值是 true
描述：如果设置为true，则仅存储来自grok的命名捕获

overwrite
值类型是 array
默认是[]
描述：覆盖已经存在的字段内容
例如：

```
filter {
  grok {
    match => { "message" => "%{SYSLOGBASE} %{DATA:message}" }
    overwrite => [ "message" ]
  }
}
```

如果日志是May 29 16:37:11 sadness logger: hello world经过match属性match => { "message" => "%{SYSLOGBASE} %{DATA:message}

pattern_definitions
值类型是 数组
默认值是 {}
描述：模式名称和模式正则表达式，也是用于定义当前过滤器要使用的自定义模式。匹配现有名称的模式将覆盖预先存在的定义。可以将此视为仅适用于grok定
例如：

```
filter {
    grok {
        patterns_dir => "/usr/local/elk/logstash/patterns"
        pattern_definitions => {"MYSELFTIMESTAMP" => "20%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{HOUR}:?%{MINUTE}(?::?%{SECO
        match => {"message" => ["%{MYSELFTIMESTAMP:timestamp} %{JAVACLASS:message}","%{MYSELF:content}"]}
    }
}
```

patterns_dir
值类型是数组
默认值是 []
描述：一些复杂的正则表达式，不适合直接写到filter中，可以指定一个文件夹，用来专门保存正则表达式的文件，需要注意的是该文件夹中的所有文件中的正

```
patterns_dir => ["/opt/logstash/patterns", "/opt/logstash/extra_patterns"]
```

正则文件以文本格式描述：

patterns_file_glob
属性值的类型：string
默认值："*"
描述：针对patterns_dir属性中指定的文件夹里哪些正则文件，可以在这个filter中生效，需要本属性来指定。默认值"*"是指所有正则文件都生效。

tag_on_failure
值类型是数组
默认值是 ["_grokparsefailure"]
描述：没有成功匹配时，将值附加到字段到tags

tag_on_timeout
值类型是字符串
默认值是 "_groktimeout"
描述：如果Grok正则表达式超时，则应用标记。

timeout_millis
值类型是数字
默认值是 30000
描述： 尝试在这段时间后终止正则表达式。如果应用了多个模式，则这适用于每个模式。这将永远不会提前超时，但超时可能需要一些时间。实际的超时时间是

常用选项
所有过滤器插件都支持以下配置选项：

dd_field
值类型是散列
默认值是 {}
描述：如果匹配成功，向此事件添加任意字段。字段名可以是动态的，并使用%{Field}包含事件的一部分

```
filter {
      grok {
        add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
      }
    }

# 你也可以一次添加多个字段
filter {
  grok {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
  }
}
```

add_tag
值类型是数组
默认值是 []
描述：如果此过滤器成功，请向该事件添加任意标签。标签可以是动态的，并使用%{field} 语法包含事件的一部分。
例如：

```
filter {
  grok {
    add_tag => [ "foo_%{somefield}" ]
  }
}

# 你也可以一次添加多个标签
```

```
filter {
  grok {
    add_tag => [ "foo_%{somefield}", "taggedy_tag"]
  }
}
```

enable_metric
值类型是布尔值
默认值是 true
描述：禁用或启用度量标准

id
值类型是字符串
此值没有默认值。
描述：向插件实例添加唯一ID，此ID用于跟踪插件特定配置的信息。
例如：

```
filter {
  grok {
    id => "ABC"
  }
}
```

periodic_flush
值类型是布尔值
默认值是 false
描述：如果设置为ture，会定时的调用filter的更新函数（flush method）

remove_field
值的类型：array
默认值：[]
描述：删除当前文档中的指定filted

```
filter {
  grok {
    remove_field => [ "foo_%{somefield}" ]
  }
}
# 你也可以一次移除多个字段：
filter {
  grok {
    remove_field => [ "foo_%{somefield}", "my_extraneous_field" ]
  }
}
```

remove_tag
值类型是数组
默认值是 []
描述：如果此过滤器成功，请从该事件中移除任意标签。标签可以是动态的，并使用%{field} 语法包括事件的一部分。
例如：

```
filter {
  grok {
    remove_tag => [ "foo_%{somefield}" ]
  }
}
# 你也可以一次删除多个标签
filter {
  grok {
    remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag"]
  }
}
```

2、mutate

官网网址：https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html

mutate允许对字段执行常规改变。可以重命名、删除、替换和修改事件中的字段。

二话不说，来个例子先：

```
filter {
    mutate {
        split => ["hostname", "."]  #切分
        add_field => { "shortHostname" => "%{hostname[0]}" } #获取切分后的第一个字段作为添加字段
    }
    mutate {
        rename => ["shortHostname", "hostname" ] #重命名
```

```
    }
}
```

mutate的配置选项：

常用的一些操作：

convert：转换数据类型，数据类型hash

可以装换的数据类型有：integer，string，integer_eu（和integer相同，显示格式为1.000），float，float_eu，boolean

实例：

```
filter {
      mutate {
        convert => {
          "fieldname" => "integer"
          "booleanfield" => "boolean"
        }
      }
    }
```

copy：类型hash，将现有字段复制到另一个字段。现有的目标域将被覆盖

实例：

```
 filter {
      mutate {
         copy => { "source_field" => "dest_field" }
      }
    }
```

gsub：类型array，根据字段值匹配正则表达式，并用替换字符串替换所有匹配项。只支持字符串或字符串数组的字段。对于其他类型的字段，将不采取任何操

实例：

```
filter {
      mutate {
        gsub => [
          # replace all forward slashes with underscore
          "fieldname", "/", "_",
          # replace backslashes, question marks, hashes, and minuses
          # with a dot "."
          "fieldname2", "[\\?#-]", "."
        ]
      }
    }
```

join：类型hash，用分隔符连接数组。对非数组字段不执行任何操作

实例：

```
filter {
     mutate {
       join => { "fieldname" => "," }
     }
   }
```

lowercase：类型array，转为小写

实例：

```
filter {
      mutate {
        lowercase => [ "fieldname" ]
      }
    }
```

merge：类型hash，合并数组或散列的两个字段。字符串字段将自动转换为数组

实例：

```
filter {
     mutate {
        merge => { "dest_field" => "added_field" }
     }
   }
```

coerce：类型hash，设置存在但为空的字段的默认值

实例：

```
filter {
      mutate {
        # Sets the default value of the 'field1' field to 'default_value'
        coerce => { "field1" => "default_value" }
      }
    }
```

rename：类型hash，重命名

实例：

```
filter {
      mutate {
```

```
        # Renames the 'HOSTORIP' field to 'client_ip'
        rename => { "HOSTORIP" => "client_ip" }
      }
    }
```

replace：类型hash，用新值替换字段的值
实例：

```
filter {
      mutate {
        replace => { "message" => "%{source_host}: My new message" }
      }
    }
```

split：类型hash，使用分隔符将字段分割为数组。只对字符串字段有效
实例：

```
filter {
      mutate {
         split => { "fieldname" => "," }
      }
    }
```

strip：类型array，字段中删除空白。注意:这只对前导和后导空格有效。
实例：

```
filter {
      mutate {
         strip => ["field1", "field2"]
      }
    }
```

update：类型hash，使用新值更新现有字段。如果该字段不存在，则不采取任何操作。
实例：

```
filter {
      mutate {
        update => { "sample" => "My new message" }
      }
    }
```

uppercase：类型array，转为大写
实例：

```
filter {
      mutate {
        uppercase => [ "fieldname" ]
      }
    }
```

capitalize：类型array，将字符串转换为等效的大写字母。
实例：

```
filter {
      mutate {
        capitalize => [ "fieldname" ]
      }
    }
```

tag_on_failure：类型string，如果在应用此变异筛选器期间发生故障，则终止其余操作，默认值：_mutate_error

公共配置见：grok公共配置

3、date

date filter用于从字段解析日期，然后使用该日期或时间戳作为事件的logstash时间戳

date常用的配置选项：

match：类型array，字段名在前，格式模式在后的数组[field,formats...]，表示该字段能够匹配到的时间模式，时间模式可以有多种
实例：

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

tag_on_failure：类型array，默认值["_dateparsefailure"]，当没有成功匹配时，将值追加到tags字段

target：类型String，默认值"@timestamp"，将匹配的时间戳存储到给定的目标字段中。如果没有提供，默认更新事件到@timestamp字段。

timezone：类型String，表示时区，可以在该网址查看：http://joda-time.sourceforge.net/timezones.html

公共配置见：grok公共配置

这里只对如上三种filter说明，具体其他的filter请见官网：https://www.elastic.co/guide/en/logstash/current/filter-plugins.html

3.10、案例一、apache日志解析

这个例子属官网的一个例子：https://www.elastic.co/guide/en/logstash/current/advanced-pipeline.html

但是我这里不弄这么负责，我们不使用filebeat，直接使用logstash，apache日志的数据集下载：
https://download.elastic.co/demos/logstash/gettingstarted/logstash-tutorial.log.gz

我这里不打算安装apach，所以直接使用官方提供的数据集。

下载数据集，然后解压文件，就可以得到我们的一个日志文件：logstash-tutorial.log

首先我们看一下apache日志的格式：

```
[elk@lgh ~]$ tail -3 logstash-tutorial.log
86.1.76.62 - - [04/Jan/2015:05:30:37 +0000] "GET /projects/xdotool/ HTTP/1.1" 200 12292 "http://www.haskell.org/haske
86.1.76.62 - - [04/Jan/2015:05:30:37 +0000] "GET /reset.css HTTP/1.1" 200 1015 "http://www.semicomplete.com/projects/
86.1.76.62 - - [04/Jan/2015:05:30:37 +0000] "GET /style2.css HTTP/1.1" 200 4877 "http://www.semicomplete.com/projects
```

然后开始配置：

```
cd logstash-7.7.0/ && mkdir conf.d
cd conf.d/
vim  apache.conf
#############apache.conf的内容如下###################
input {
        file {
            path => "/home/elk/logstash-tutorial.log"
            type => "log"
            start_position => "beginning"
            }
    }
filter {
    grok {
        match => { "message" => "%{COMBINEDAPACHELOG}"}
    }
}

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

然后启动命令（可以选择用nohup后台启动）：

```
 cd logstash-7.7.0/ && ./bin/logstash -f conf.d/apache.conf
```

执行结果如下（部分结果）：

执行结果

2.11、案例二、nginx日志解析

首先安装nginx：nginx功能介绍和基本安装

这里我们也不使用filebeat，因为这篇文章只是介绍logstash

配置：

```
cd conf.d/
vim  nginx.conf
############nginx.conf配置如下#####################
input {
        file {
            path => "/usr/local/nginx/logs/access.log"
            type => "log"
            start_position => "beginning"
            }
    }

filter {
grok {
        match => { "message" => ["(?<RemoteIP>(\d*.\d*.\d*.\d*)) - %{DATA:[nginx][access][user_name]} \[%{HTTPDATE:[n
        add_field => {
                "Device" => "Charles Desktop"
        }
        remove_field => "message"
        remove_field => "beat.version"
        remove_field => "beat.name"
```

```
    }
}
    output {
             elasticsearch {
                    hosts => ["192.168.110.130:9200"]
                    index => "nginx-log-%{+YYYY.MM.dd}"
                }
    }
```

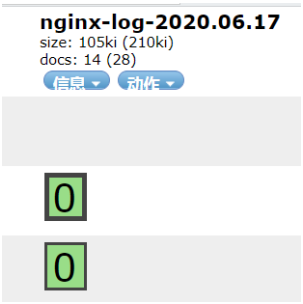
如上的配置中输出到elasticsearch中，这里没有设置密码，所以不需要用户和密码，还有就是这里使用的默认模板，如果想要修改的话可以使用，可以添加如下配置：

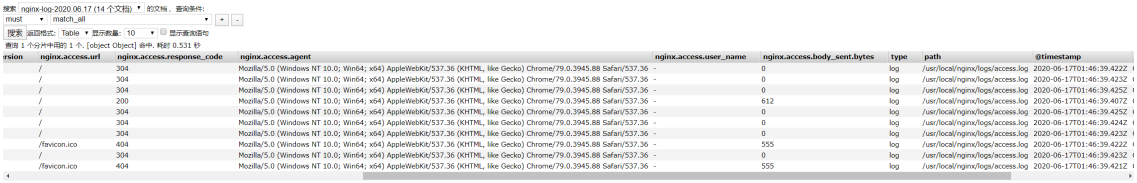
```
user => "elastic"  #用户
password => "${ES_PWD}" #通过keystore存储的密码
manage_template => false #关闭默认的模板
template_name => "elastic-slowquery" #指定自定义的模板
```

执行命令启动logstash

```
cd logstash-7.7.0/ && ./bin/logstash -f conf.d/nginx.conf
```

执行的结果（查看elasticsearch集群）：





rsion	nginx.access.url	nginx.access.response_code	nginx.access.agent	nginx.access.user_name	nginx.access.body_sent.bytes	type	path	@timestamp
	/	304	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	0	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.422Z
	/	304	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	0	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.423Z
	/	304	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	0	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.425Z
	/	200	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	612	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.407Z
	/	304	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	0	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.425Z
	/	304	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	0	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.424Z
	/	304	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	0	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.423Z
	/favicon.ico	404	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	555	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.422Z
	/	304	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	0	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.423Z
	/favicon.ico	404	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36	-	555	log	/usr/local/nginx/logs/access.log	2020-06-17T01:46:39.421Z

从上面的两个图看，一个创建了我们在nginx.conf中指定的一个索引，然后索引的内容都是解析出来的一些字段内容。

3.12、案例三、elasticsearch慢日志解析

这是实例我们采用filebeat+logstash+elasticsearch，还有权限验证进行试验：

这里主要是对elasticsearch的慢日志查询做解析，虽然我在一篇文章搞懂filebeat（ELK）中篇文章中直接通过filebeat的elasticsearch（beat版本）的模块对其做过解析，但是解析的还是不够特别完善，这里引入logstash对其解析，过滤。

首先配置filebeat文件（这里只配置了一个输入和一个输出，没有做多余的处理，只是用来收集日志）：

```
#=========================== Filebeat inputs =============================
filebeat.inputs:

# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input specific configurations.

- type: log

  # Change to true to enable this input configuration.
  enabled: true

  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - /var/logs/es_aaa_index_search_slowlog.log
    - /var/logs/es_bbb_index_search_slowlog.log
    #- c:\programdata\elasticsearch\logs\*

#================================ Outputs =====================================

# Configure what output to use when sending the data collected by the beat.

#-------------------------- Logstash output -------------------------------
output.logstash:
  # The Logstash hosts
```

```
  hosts: ["192.168.110.130:5044","192.168.110.131:5044","192.168.110.132:5044"]
  loadbalance: true   #这里采用负载均衡机制，

  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"
```

然后启动filebeat：

```
cd filebeat-7.7.0-linux-x86_64 && ./filebeat -e
```

然后配置logstash的配置文件：

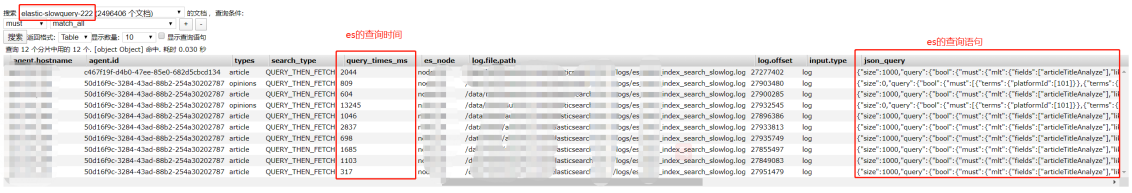
```
cd conf.d/
vim  es.conf
#############es.conf配置如下###############
input {
        beats{
                port => 5044
        }
    }

filter {
grok {
        match => {"message" => "\[%{TIMESTAMP_ISO8601:query_time},%{NUMBER:number1}\]\s*\[%{DATA:log_type}\]\s*\[%{DA
        remove_field => ["message","@version","status","times_s","@timestamp","number1"]
    }
  mutate{
        convert => {
        "query_times_ms" => "integer"
}
}
}
    output {
            elasticsearch {
                  hosts => ["192.168.110.130:9200","192.168.110.131:9200","192.168.110.132:9200"]
                  index => "elastic-slowquery-222"
                  user => "elastic"
                  password => "${ES_PWD}"
                  manage_template => false
                  template_name => "elastic-slowquery"
                }
    }
```

如上配置使用了用户的权限验证，以及elasticsearch的自定义模板

启动logstash

```
cd logstash-7.7.0/ && ./bin/logstash -f conf.d/es.conf
```

登录es查看结果：



logstash就介绍到这里了，如果有疑问多看官网比较好

参考：

logstash官网：https://www.elastic.co/guide/en/logstash/current/index.html

每个阶段有每个阶段的使命，愿自己的努力，能够更好的完成自己的使命，谨记：别忘了自己是谁！！！

分类: ELK

一寸HUI
关注 - 3
粉丝 - 27
+加关注

1 0

« 上一篇： 一篇文章搞懂filebeat（ELK）
» 下一篇： 搞懂ELK并不是一件特别难的事（ELK）

posted @ 2020-06-17 10:35 一寸HUI 阅读(759) 评论(1) 编辑 收藏

评论列表

#1楼 2020-06-17 10:39 烟花散尽13141

看不懂

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。

相关博文：
· ELK(Elasticsearch + Logstash + Kibana) 日志收集
· CentOS7下安装ELK（nginx 、elasticsearch-5.1.1、logstash-5.1.1、kibana-5.1.1）
· ELK学习实验012：Logstash的安装和使用
· elk分布式+ logstash日志监控+kibana监控
· Logstash
» 更多推荐...

最新 IT 新闻:
· 快递价格战压缩快递员收入：一单仅赚2毛5 用户体验直线下降
· 茅台被严重低估
· 字节跳动回应关于TikTok传言：方案不涉及任何算法和技术的转让
· Facebook押宝VR和AR硬件的背后原因是什么？
· 广州城璞长租公寓"爆雷"：此前房租骤降 有人刚搬进去
» 更多新闻...