**ELK学习总结**

本月主要学习了ELK。ELK是 Elasticsearch、Logstash、Kibana 三个开源软件组合，是一种能够从任意数据源抽取数据，并实时对数据进行搜索、分析和可视化展现的数据分析框架。通过学习，理解到三个软件在其中各自起到的作用，以及它们之间的相互关联和互补。从作用，到安装，到配置文件的分析，到启动，到关闭，以及使用各个方面学习ELK，并通过查看官方文档查漏补缺。最后通过真实运用，部署一整套系统，并成功实时对日志数据进行搜索、分析和可视化展现，更加清晰理解ELK。

下面是学习中的一些记录：

**一、ElasticSearch**

**1、什么是Elasticsearch**

ElasticSearch是一个基于 Lucene 的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于 RESTful web 接口。Elasticsearch 是用 Java 开发的，并作为 Apache 许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

**2、Elasticsearch: 权威指南**<https://www.elastic.co/guide/cn/elasticsearch/guide/current/index.html>

**3、安装**

下载地址：

<https://www.elastic.co/downloads/elasticsearch>

wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.3.0.tar.gz

tar -xzf elasticsearch-6.3.0.tar.gz

cd elasticsearch-6.3.0/

**4、修改配置文件config/elasticSearch.yml**

访问端口号：http.port: 9084

开启远程访问功能：network.host: 0.0.0.0

**5、启动程序**

bin/elasticsearch 或 bin/elasticsearch -d # -d 表示后台启动

查看是否成功：http://192.168.243.19:9084/

**6、关闭程序**

使用ps aux|grep elasticsearch查看，或者使用jps查看进程号

在用kill 命令强制停止即可

**7、使用**

常用命令：

1）查看Index：curl -X GET 'http://localhost:9084/\_cat/indices?v'

2）新建 Index：curl -X PUT 'localhost:9200/weather'

curl -X PUT 'localhost:9084/accounted' -d '

{

 "mappings": {

   "person": {

     "properties": {

       "user": {

         "type": "text"

      },

       "title": {

         "type": "text"

      },

       "desc": {

         "type": "text"

      }

    }

  }

}

}'

3）删除 Index：curl -X DELETE 'localhost:9084/accounts'

4）新增记录（文档）：

curl -X POST 'localhost:9084/accounts/person' -d '

{

 "user": "李四",

 "title": "工程师",

 "desc": "系统管理"

}'

5）查看记录：curl 'localhost:9084/accounts/person/1?pretty=true'

6）删除记录：curl -X DELETE 'localhost:9084/accounts/person/1'

7）修改记录：

curl -X PUT 'localhost:9084/accounts/person/1' -d '

{

   "user" : "张三",

   "title" : "工程师",

   "desc" : "数据库管理，软件开发"

}'

8）查询所有记录：

curl 'localhost:9084/accounts/person/\_search'

<http://192.168.243.19:9084/bank/_search>

9）条件查询：

<http://192.168.243.19:9084/bank/_search?q=age:23>

10）表达式查询：

curl 'localhost:9084/bank/\_search' -d '

{

 "query" : { "match" : { "age" : "23" }}

}'

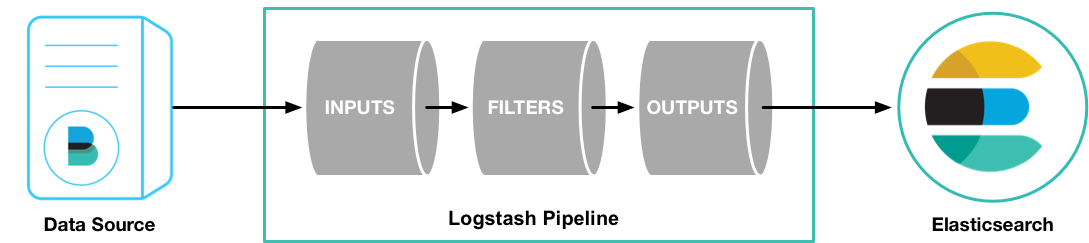
**二、Logstash**

**1、什么是Logstash**

Logstash 是一个开源的数据收集引擎，它具备实时数据传输能力。它可以统一过滤来自不同源的数据，并按照开发者的制定的规范输出到目的地。

顾名思义，Logstash 收集数据对象就是日志文件。由于日志文件来源多（如：系统日志、服务器 日志等），且内容杂乱，不便于人类进行观察。因此，我们可以使用 Logstash 对日志文件进行收集和统一过滤，变成可读性高的内容，方便开发者或运维人员观察，从而有效的分析系统/项目运行的性能，做好监控和预警的准备工作等。

Logstash 是一款强大的数据处理工具，它可以实现数据传输，格式处理，格式化输出，还有强大的插件功能，常用于日志处理。



**2、安装**

Logstash 依赖 JDK1.8 ，因此在安装之前请确保机器已经安装和配置好 JDK1.8。

Logstash 下载 https://www.elastic.co/cn/downloads/logstash

tar -xzf logstash-6.3.0.tar.gz

cd logstash-5.6.3

**3、修改配置文件config/logstash.yml**

访问端口号：http.port: 9086-9086

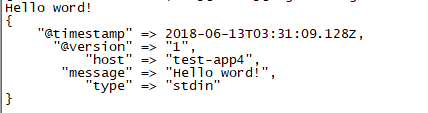
**4、使用**

通过命令行，进入到logstash/bin目录，执行下面的命令：

logstash -e ""

C:\Users\78200\Documents\My Knowledge\temp\cf267c04-17cb-4427-98c8-79d279ba61cd\128\index_files\7936ff04-6baf-46fa-b867-0b1d230fa4a7.png

可以看到提示下面信息（这个命令稍后介绍），输入hello world!

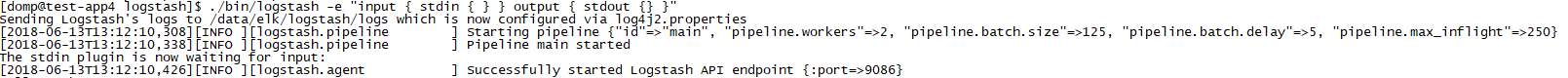


可以看到logstash尾我们自动添加了几个字段，时间戳@timestamp，版本@version，输入的类型type，以及主机名host。

再来看另外一个例子：

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

启动 Logstash 后，再键入 Hello World，结果如下：



HelloWorld

C:\Users\78200\Documents\My Knowledge\temp\cf267c04-17cb-4427-98c8-79d279ba61cd\128\index_files\ada62d33-1303-4bef-95f6-e99726992487.png

Hello World（输入）经过 Logstash 管道（过滤）变成：2018-06-13T05:12:50.410Z test-app4 HelloWorld! （输出）。

细心的会发现，这个日志输出与上面的logstash -e "" 并不相同，这是因为上面的命令默认指定了返回的格式是json形式。

在生产环境中，Logstash 的管道要复杂很多，可能需要配置多个输入、过滤器和输出插件。

因此，需要一个配置文件管理输入、过滤器和输出相关的配置。配置文件内容格式如下：

#　输入

input {

...

}

# 过滤器

filter {

...

}

# 输出

output {

...

}

根据自己的需求在对应的位置配置 输入插件、过滤器插件、输出插件 和 编码解码插件 即可。

命令行中常用的命令

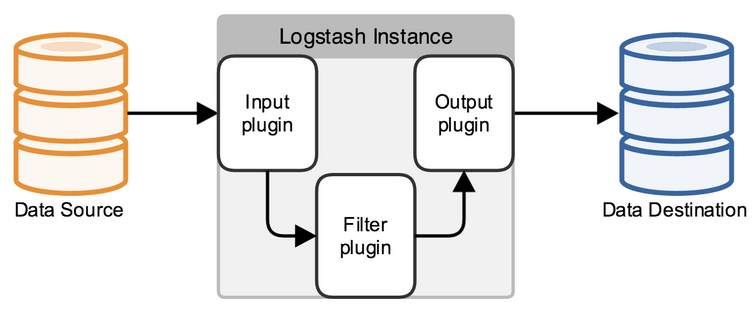
　　-f：通过这个命令可以指定Logstash的配置文件，根据配置文件配置logstash

　　-e：后面跟着字符串，该字符串可以被当做logstash的配置（如果是“” 则默认使用stdin作为输入，stdout作为输出）

　　-l：日志输出的地址（默认就是stdout直接在控制台中输出）

　　-t：测试配置文件是否正确，然后退出。

**配置文件说明**



　　在每个部分中，也可以指定多个访问方式，例如我想要指定两个日志来源文件，则可以这样写：

input {

file { path =>"/var/log/messages" type =>"syslog"}

file { path =>"/var/log/apache/access.log" type =>"apache"}

}

　　类似的，如果在filter中添加了多种处理规则，则按照它的顺序一一处理，但是有一些插件并不是线程安全的。

　　比如在filter中指定了两个一样的的插件，这两个任务并不能保证准确的按顺序执行，因此官方也推荐避免在filter中重复使用插件。

　　说完这些，简单的创建一个配置文件的小例子看看：

复制代码

input {

file {  
　　　　 #指定监听的文件路径，注意必须是绝对路径

path => "E:/software/logstash-1.5.4/logstash-1.5.4/data/test.log"

start\_position => beginning

}

}

filter {

}

output {

stdout {}

}

复制代码

　　日志大致如下：

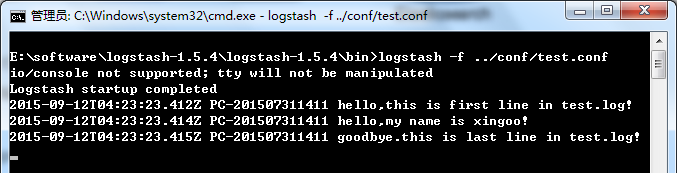
1 hello,this is first line in test.log!

2 hello,my name is xingoo!

3 goodbye.this is last line in test.log!  
4

　　注意最后有一个空行。

　　执行命令得到如下信息：



**四、插件用法**

在使用插件之前，我们先了解一个概念：事件。

Logstash 每读取一次数据的行为叫做事件。

在 Logstach 目录中创建一个配置文件，名为 logstash.conf（名字任意）。

4.1 输入插件

输入插件允许一个特定的事件源可以读取到 Logstash 管道中，配置在 input {} 中，且可以设置多个。

新建配置文件：

vim logstash2.config

input {

   file {

       path => "/data/elk/info.log.2018-04-02"

       type => "error"

       start\_position => "beginning"

  }

}

filter {

​

}

output {

   elasticsearch {

       hosts => ["localhost:9084"]

       index => "error-%{+YYYY.MM.dd}"

  }

}

其中error.log就是随便找一些错误的日志文件即可,

error.log 的内容格式如下：

2017-08-04 13:57:30.378 [http-nio-8080-exec-1] ERROR c.g.a.global.ResponseResultAdvice -设备数据为空

com.light.pay.common.exceptions.ValidationException: 设备数据为空

   at com.light.pay.common.validate.Check.isTrue(Check.java:31)

   at com.light.attendance.controllers.cloudApi.DevicePushController.deviceInfoPush(DevicePushController.java:44)

   at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

   at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

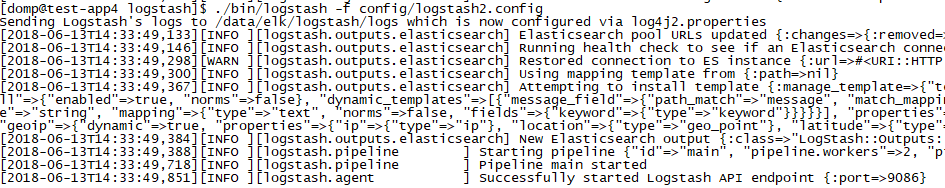
   at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)

   at java.lang.Thread.run(Thread.java:745)

2017-08-04 13:57:44.495 [http-nio-8080-exec-2] ERROR c.g.a.global.ResponseResultAdvice -Failed to invoke remote method: pushData, provider: dubbo://192.168.2.100:20880/com.light.attendance.api.DevicePushApi?application=salary-custom&default.check=false&default.timeout=30000&dubbo=2.8.4&interface=com.light.attendance.api.DevicePushApi&methods=getAllDevices,getDeviceById,pushData&organization=com.light.attendance&ow

保存文件。键入：

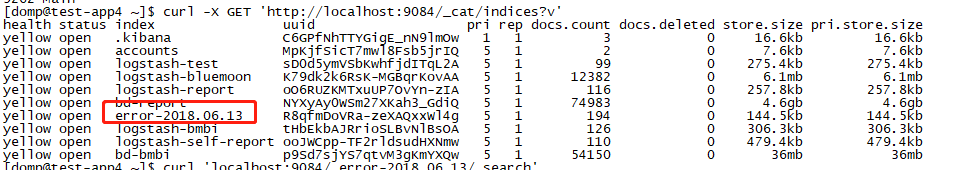
./bin/logstash -f logstash2.config



4.2 输出插件

输出插件将事件数据发送到特定的目的地，配置在 output {} 中，且可以设置多个。

配置文件中使用 elasticsearch 输出插件。输出的日志信息将被保存到 Elasticsearch 中，索引名称为 index 参数设置的格式。



curl 'localhost:9084/error-2018.06.13/\_search?q=\*&pretty'



数据运行成功。

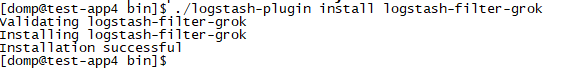
**4、3 过滤器插件**

过滤器插件位于 Logstash 管道的中间位置，对事件执行过滤处理，配置在 filter {}，且可以配置多个。

本次测试使用 grok 插件演示，grok 插件用于过滤杂乱的内容，将其结构化，增加可读性。

安装：

bin/logstash-plugin install logstash-filter-grok



修改配置文件：

input {

    stdin {}

}

filter {

    grok {

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

:duration}" }

    }

}

output {

    stdout {

       codec => "rubydebug"

    }

}

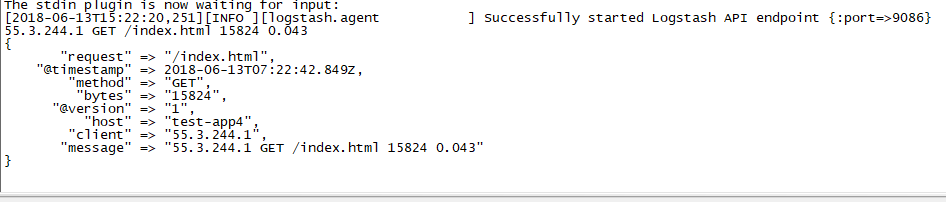
保存文件。键入：

./bin/logstash -f config/logstash2.config

启动成功后，我们输入：

55.3.244.1 GET /index.html 15824 0.043

控制台返回：



输入的内容被匹配到相应的名字中。

最后的配置如下：

input {

   file {

       # 要导入的文件的位置，可以使用\*，例如/var/log/nginx/\*.log

       path => "/data/syslogs/bd\_eureka\_server/info.log"

        #type是给结果增加了一个属性叫type值为"<xxx>"的条目。这里的type，对应了ES中index中的type，即如果输入ES时，没有指定type，那么这里的type将作为ES中index的type。

       type => "info"

       #start\_position可以设置为beginning或者end，beginning表示从头开始读取文件，end表示读取最新的，这个也要和ignore\_older一起使用。

       start\_position => end

        #sincedb\_path表示文件读取进度的记录，每行表示一个文件，每行有两个数字，第一个表示文件的inode，第二个表示文件读取到的位置（byteoffset）。默认为$HOME/.sincedb\*

 sincedb\_path => "/data/elk/log/log\_progress"

 #ignore\_older表示了针对多久的文件进行监控，默认一天，单位为秒，可以自己定制，比如默认只读取一天内被修改的文件。

 ignore\_older => 604800

 #add\_field增加属性

 add\_field => {"project"=>"bd\_eureka\_server"}

   }

file {

       path => "/data/elk/error.log"

       type => "error"

       start\_position => end

 sincedb\_path => "/data/elk/log/log\_progress"

 ignore\_older => 604800

 add\_field => {"project"=>"bd\_eureka\_server"}

 #表示将多行合并为一条数据，默认是一行一条数据的

 codec => multiline {

   pattern => "^%{TIMESTAMP\_ISO8601}"

   negate => true

   #what有两个值可选 previous和next，将数据合并到前一条数据还是后一条数据中

   what => "previous"

   #5秒内如果没有新的数据，就将数据提交

   auto\_flush\_interval => 5

   #总体的意思就是，如果日志不是以时间日期开头的话，就默认为是上一条的数据

 }

   }

}

filter {

grok {

    #grok 插件用于过滤杂乱的内容，将其结构化，以方便es的查询

 match=>{"message"=>"%{TIMESTAMP\_ISO8601:time}\-\[%{LOGLEVEL:level}\]\-\((?<classname>([\s\S]\*))\:%{NUMBER:lineNumber}\)(?<msg>([\s\S]\*))"}

}

}

output {

   elasticsearch {

   #输出到ES中

 codec => "json"

       hosts => ["localhost:9084"]

 index => "%{project}\_%{type}\_%{+YYYY.MM.dd}"

   }

}

文档见：

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

**三、Kibana**

**1、什么是Kibana**

Kibana 是一款开源的数据分析和可视化平台，它是 Elastic Stack 成员之一，设计用于和 Elasticsearch 协作。您可以使用 Kibana 对 Elasticsearch 索引中的数据进行搜索、查看、交互操作。您可以很方便的利用图表、表格及地图对数据进行多元化的分析和呈现。

Kibana 可以使大数据通俗易懂。它很简单，基于浏览器的界面便于您快速创建和分享动态数据仪表板来追踪 Elasticsearch 的实时数据变化。

搭建 Kibana 非常简单。您可以分分钟完成 Kibana 的安装并开始探索 Elasticsearch 的索引数据 — 没有代码、不需要额外的基础设施。

**2、Kibana用户手册**

https://www.elastic.co/guide/cn/kibana/current/index.html

**3、 安装 Kibana**

<https://www.elastic.co/guide/cn/kibana/current/targz.html>

wget https://artifacts.elastic.co/downloads/kibana/kibana-6.3.0-linux-x86\_64.tar.gz

tar -xzf kibana-6.3.0-linux-x86\_64.tar.gz

cd kibana-6.3.0-linux-x86\_64/

mv kibana-5.2.2-linux-x86\_64  /data/elk/

mv kibana-5.2.2-linux-x86\_64 kibana

**4、 修改配置文件**

cd /data/elk/kibana/config

vi kibana.yml

==修改内容如下==

server.port: 9085

server.host: "192.168.243.19"

elasticsearch.url: "[http://127.0.0.1:9084"](http://127.0.0.1:9084%22)

**5、 启动**

./kibana //不能关闭终端

nohup ./kibana > /nohub.out & //可关闭终端，在nohup.out中查看log

**6、关闭程序**

使用ps aux|grep kibana查看

在用kill 命令强制停止即可

**7、使用**

1）ES中加载数据   
2）定义自己的索引模式

3）探索数据

line\_id:<100 AND speech\_number:<100

或者：account\_number:<100 AND balance:>47500

4）可视化数据

5）使用仪表板汇总数据